

Iso Virtual  
VTB Application for IsoNs  
[www.promax.it](http://www.promax.it)



The contained information in this handbook are only informative and they can being change without warning and they must not being understandings with some engagement from Promax srl. Promax srl does not assume responsibility or obligates for errors or inaccuracies that can be found in this handbook. Except how much granted from the license, no part of this publication can be reproduced, saved in a recording system or transmitted in whatever form or with any means, electronic, mechanical or recording system or otherwise without Promax srl authorization.

Any reference to names of society or products have only demonstrative scope and it does not allude to some real organization.

Rev. 2.0.2

## 1 PREFACE

This manual describes the VTB application for IsoNs software.

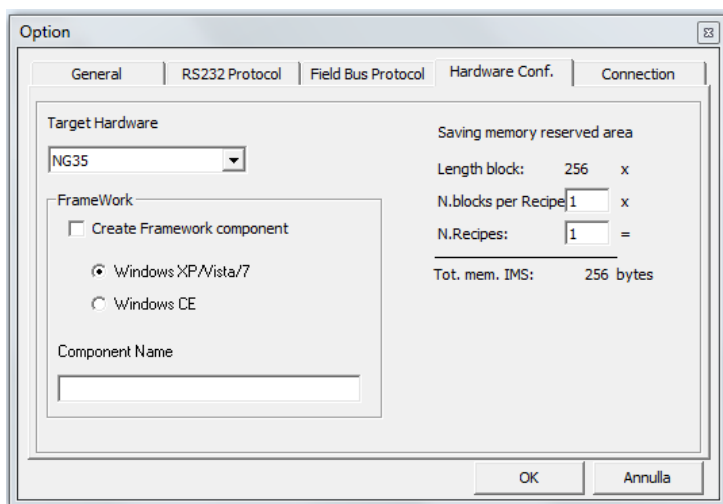
The VTB application is the same for the IsoNs Windows Xp,7,8 and IsoNs Windows CE

## 2 BASE COMPONENTS

To begin the application, is necessary to insert the following components in a new VTB Application:

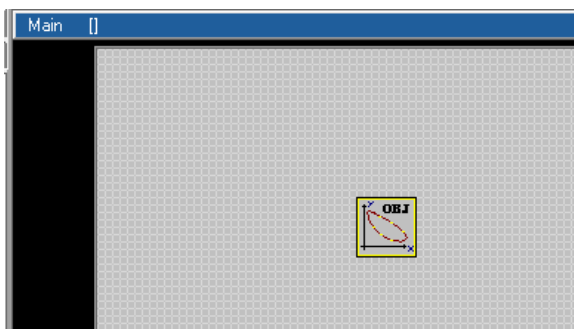
### 2.1 Select Hardware

*Tools → Options → Hardware conf. (Ex NG35)*



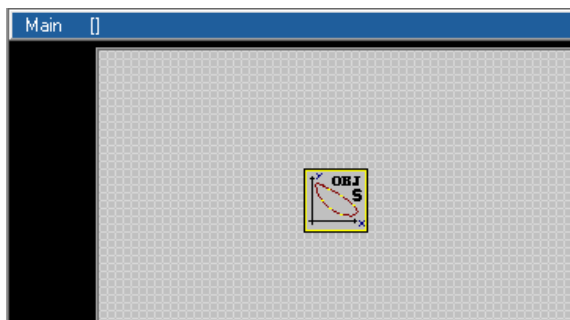
### 2.2 Insert Object *cobjinterpola*

*Objects → Motor Control → cobjinterpola.vco*



You can also insert the Obj interpola Rampe S

The difference is that in this object you can use the ISONS Parameter **JERK**, sinus acceleration ramps.



## 2.3 Properties setting of objinterpolata

nome (name)	→	objinterpolata1 (default)
N. Assi (Number of Axes)	→	Number of interpolated axes for this process
N.tratti (depth look ahead )	→	Look Ahead depth Default 16
Vper	→	Not change
Div. Vper	→	Not change
Abilita Arcto	→	Not change

Interpolata1	
Property	Events
Property	Value
Nome	Interpolata1
Left	130
Top	75
N.assi	3
N.tratti	16
Vper	1024
Div. Vper	1024
Abilita arcto	1

Max DEPTH look ahead (this value must be multiple of 2 ex: 8 – 16 – 32 etc.)

### NGQuark

- 2 Axes → 32
- 3 Axes → 16
- 4 Axes → 8

### NGM EVO

- 2 Axes → 128
- 3 Axes → 128
- 4 Axes → 128

### NG35

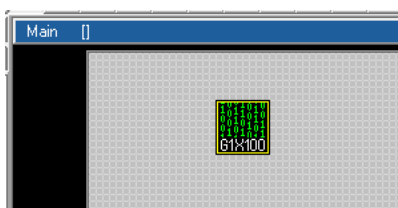
- All → 2048

## 2.4 Insert Object Iso Virtual

**Objects → IsoNs → - ISOVirtual - \$Rev 2.3.0 (ISOVirtual - \$Rev 1.1.9 is Obsolete)**

The ISOVirtual light is used for **NGQ or NGQx** system only

This component uses little memory



## 2.5 Properties setting of IsoVirtual

nome (name)	→	ISOV1 (default)
Indice processo	→	Not change
Parametri Custom (Curtom Parameter)	→	See below

ISOV1	
Property	Events
Property	Value
Nome	ISOV1
Left	95
Top	35
Indice processo	1
Parametri custom	0

## 2.6 Insert a long fixed variable name fixed0

This variable is used to synchronized the IsoNs PC process

### Objects → Fixed Var

Internal VAR	Bit VAR	Define	Static VAR	VSD VAR	Fixed VAR
fixed0		LONG	EXP <input type="checkbox"/>		
Addr	Variable	Type			

### Select the first address (click in the 0 position)

Addr	Variable	Type
0		
1		

### Press Add Button

Addr	Variable	Type
0	fixed0	LONG
1	*****	*****
2	*****	*****
3	*****	*****

## 3 Insert the Axes type

After insert the base components, is possible insert the axes

### Choose the axes type

Objects → IsoNs → IsoCanOpen

#### Axes Type

##### *IsoCanOpen*

This contain all the CanOpen Driver

Typical set

AX1	
Property	Events
Property	Value
Nome	AX1
Left	90
Top	175
Nodo	1
Indice asse ISO	0
Nome processo	ISOV1
Nome quota pdo	qi
Mask Allarmi	0xFFFFFFFF
Mask Parametri	0x00

Nome(Name)

→ Axis Name

Nodo (Node)

→ CanOpen Node

Indice Asse Iso (Index ISO Axis)

→ Indicates the ISONS Axis (0 -X , 1-Y , 2 – Z etc.)

Nome Processo (process Name)

→ Process Name with is associate ISOVIRTUAL OBJECT

Nome quota pdo (PDO CanOpen Name)

→ PDO CanOpen Name for interpolation mode

**IsoPid.vco**

This contain all the Analog driver +/- 10V with encoder loop Driver,closed loop for Stepper motor and CanOpen (with external encoder)

**IsoPid- NG35 Filtro Digitale**



Used for analog motor with +/-10 V and encoder for closed loop only for NG35 and NGIO expansion

**Properties**

- Nome(Name) → Object IsoPid name
- Indice Asse Iso (Index ISO Axis) → Indicates the ISO Axis (0 -X , 1-Y , 2 – Z etc.)
- Nome Processo (process Name) → Process Name with is associate ISOVIRTUAL OBJECT
- Indice Asse NGIO (Index NGIO Axis) → Indicates the channel of NGIO with is connect the driver
  - 0 → Ch0 of First NGIO
  - 1 → Ch1 of First NGIO
  - 2 → Ch0 of Second NGIO
  - 3 → Ch1 of second NGIO etc.
- Kp,Ki,Kv,Err Saturazione,Divisore,Dir → PID Parameters set by IsoNs PC application
- Enable Kp,Ki,Kd → Not change (True)
- T0 Level → Level of encoder index
  - 0 Low
  - 1 High
- Soglia ServoErr,Ritardo ServoErr → Parameters set by IsoNs PC application

**NGPP- NG-PP + PID**



Used for closed loop in STEPPER MOTOR. Normally this object is used for high precision axes



**Properties**

Nome(Name)	→ Object IsoPid name
Indice Asse Iso (Index ISO Axis)	→ Indicates the ISO Axis (0 -X , 1-Y , 2 – Z etc.)
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
Indice Asse NGPP (Index NGPP Axis)	→ Indicates the channel of NGPP with is connect the driver
	0 → Ch0 of First NGPP
	1 → Ch1 of First NGPP
	2 → Ch2 of First NGPP
	3 → Ch3 of First NGPP
	4 → Ch0 of Second NGPP
	5 → Ch1 of Second NGPP Etc
Indice Asse NGIO (Index NGIO Axis)	→ Indicates the encoder channel input relative to axis
	0 → Ch0 of First NGIO
	1 → Ch1 of First NGIO
	2 → Ch0 of Second NGIO
	3 → Ch1 of second NGIO etc.
Kp,Ki,Kv,Err Saturazione,Divisore,Dir	→ PID Parameters set by IsoNs PC application
Enable Kp,Ki,Kd	→ not change (True)
T0 Level	→ Level of encoder index
	0 Low
	1 High
Soglia ServoErr,Ritardo ServoErr	→ Parameters set by IsoNs PC application
Max Freq,Scalav,Enable Out	→ Reserved

**IsoDouble\_Enc.vco**

This contain all the Iso CanOpen with external encoder closed loop

**CanOpen + PID**

Used for external closed loop in CanOpen ESTUN DRIVES with interpolation mode. Normally this object is used for high precision axes



**Properties**

Nome(Name)	→ Object IsoPid name
Nodo(Node)	→ Driver Canopen Node
Indice Asse Iso (Index ISO NS Axis)	→ Indicates the ISO NS Axis (0 -X , 1-Y , 2 – Z etc.)
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
Nome quota pdo (PDO CanOpen Name)	→ PDO CanOpen Name for interpolation mode
Indice Asse NGIO (Index NGIO Axis)	→ Indicates the encoder channel input relative to axis
	0 → Ch0 of First NGIO
	1 → Ch1 of First NGIO
	2 → Ch0 of Second NGIO
	3 → Ch1 of second NGIO etc.
Kp,Ki,Kv,Err Saturazione,Divisore,Dir	→ PID Parameters set by IsoNs PC application
Enable Kp,Ki,Kd	→ Not change (True)
T0 Level	→ Level of encoder index
	0 Low
	1 High
Soglia ServoErr,Ritardo ServoErr	→ Parameters set by IsoNs PC application
Home Delay	→ Delay in Millisecond for home commando (typically 1000)



**IsoPP.vco**

This contain all the Axes STEP DIR open loop

**IsoPP NGM EVO Passo-Passo**

Used for the NGM EVO board and STEP DIR axes

**Properties**

Nome(Name)	→ Object IsoPid name
Indice Asse Iso (Index ISONS Axis)	→ Indicates the ISONS Axis (0 -X , 1-Y , 2 – Z etc.)
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
NGM EVO Channel	→ Indicates the channel of NGM EVO with is connect the driver
	0 → Ch0
	1 → Ch1
	2 → Ch2
	3 → Ch3

**IsoPP NG35 (NG-PP) Passo-Passo**

Used for the NG35+NGPP board and STEP DIR axes

**Properties**

Nome(Name)	→ Object IsoPid name
Indice Asse Iso (Index ISONS Axis)	→ Indicates the ISONS Axis (0 -X , 1-Y , 2 – Z etc.)
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
NG-PP Channel	→ Indicates the channel of NGPP with is connect the driver
	0 → Ch0 first NGPP
	1 → Ch1 first NGPP
	2 → Ch2 first NGPP
	3 → Ch3 first NGPP
	4 → Ch0 second NGPP
	5 → Ch1 second NGPP etc.

**IsoPP\_slave.vco**

This contain all the Axes STEP DIR open loop slave mode.

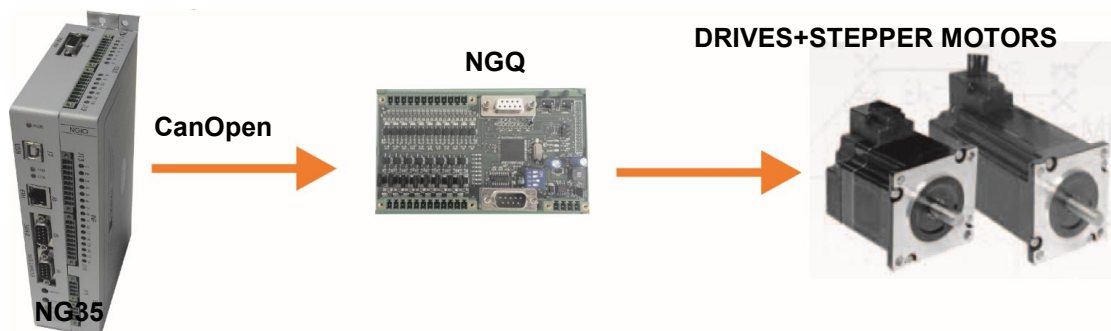
In this mode, the master can control the slave axes step dir connect at a board via CanOpen

Ex:

- NG35 Master
- NGQ Slave CanOpen with step axes

**IsoPP\_slave (Do not Use NGM13)**

Used for the NGQ board and STEP DIR axes in canopen



**Properties**

Nome(Name)	→ Object IsoPid name
Indice Asse Iso (Index ISONS Axis)	→ Indicates the ISONS Axis (0 -X , 1-Y , 2 – Z etc.)
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
NGM-13 Channel	→ Indicates the channel of NGQ with is connect the driver
	0 → Ch0
	1 → Ch1
	2 → Ch2
	3 → Ch3
Nodo(Node)	→ NGQ CanOpen node
Nome quota pdo (PDO CanOpen Name)	→ PDO CanOpen Name for interpolation mode

**4 Insert the I/O**

When the all axes are inserted, is possible insert the additional I/O module

**Choose the I/O type**

Objects → IsoNs → IsoCanOpen

**Iso-IO.vco**

This contain all the I/O module

**Iso-IO – NG-IO - NGM-IO**

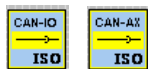
I/O on the NGIO -NGMIO (NG35 -NGM EVO expansion)

**Properties**

Nome(Name)	→ Object IsoPid name
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
Indice ISO-IO (16 bit) (ISO-IO INDEX)	→ Number of block I/O from 0 to 15
	0 → Group 1 - I/O from 0 to 15
	1 → Group 2 - I/O from 16 to 31
	3 → Group 3 – I/O from 32 to 47 etc.
Indice NG-IO (NG-IO Index)	→ Indicates the number NGIO oer NGMIO expansion
	0 → First NGIO-NGMIO on local bus
	1 → Second NGIO-NGMIO on local bus
	2 → Third NGIO-NGMIO on local bus Etc.
Hardware enable	→ Do not change (true)

**Iso-IO - CAN-IO – CAN-AX - NGM-13 SLAVE**

I/O on the NGQ or NGQx slave CanOpen

**Properties**

Nome(Name)	→ Object IsoPid name
Nodo(Node)	→ NGQ - NGQx -NGM EVOs CanOpen node
Allarme cfg (CFG Alarm)	→ Number of alarm generated when the board is in error default 41 If you have more boards NGQ,NGQx,NGM EVOs , increase this value (42,43 etc)
Nome Processo (process Name)	→ Process Name with is associate ISOVIRTUAL OBJECT
Indice ISO-IO (16 bit) (ISO-IO INDEX)	→ Number of block I/O from 0 to 15
	0 → Group 1 - I/O from 0 to 15
	1 → Group 2 - I/O from 16 to 31
	3 → Group 3 – I/O from 32 to 47 etc.
Hardware enable	→ Do not change (true)
Variabile Inp (Variable digital inputs PDO)	→ Name for PDO digital inputs
Variabile Out (Variable digital outputs PDO)	→ Name for PDO digital outputs

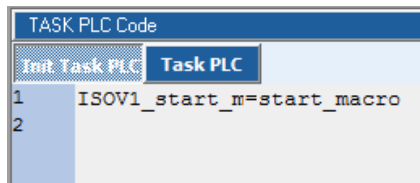
## CAN-PPN,- EMC-IO Do not use obsolete

### 5 USE the M functions to internal the CNC

In this example the ISO VIRTUAL OBJECT NAME is ISOV1

1) Set the VTB pointer at function **ISOV1\_start\_m**

Insert the code in **INIT TASK PLC**



Start\_macro is a function located in Main → Page Functions

This function is called automatically when the **M** function is invoked by PC

start macro Function

```
function start_macro() as void
ISOV1_M_ack=1 ' set acknoleged for ISONS PC
select ISOV1_M_cmd
  case 1
    ' M Function 1
  case 2
    ' M Function 2
  case 3
    ' M Function 3
  case else
    ISOV1_M_ack=0 ' M not found
endselect
endfunction
```

## 5.1 Read the M parameters

IsoNs can write the M parameters from code:

### *IsoNs Code*

```
$_PARAM1=100
$_PARAM2=130
M5
```

You can read in the VTB application this parameters in the following variables:

```
ISOV1_M_1 → $_PARAM1 ISOV1_M_2 → $_PARAM2
ISOV1_M_3 → $_PARAM3 ISOV1_M_4 → $_PARAM4
ISOV1_M_5 → $_PARAM5 ISOV1_M_6 → $_PARAM6
ISOV1_M_7 → $_PARAM7 ISOV1_M_8 → $_PARAM8
ISOV1_M_9 → $_PARAM9 ISOV1_M_10 → $_PARAM10
```

### *VTB Code*

```
function start_macro() as void
    ISOV1_M_ack=1 ' set acknoleged for ISONS PC
    select ISOV1_M_cmd
        case 5
            ' M Function 5
            if ISOV1_M_1=100 && ISOV1_M_2=130
                ....
            endif
            ISOV1_status_m_run=0 ' Free the IsoNs application
        case else
            ISOV1_M_ack=0 ' M not found
    endselect
endfunction
```

## 5.2 Write the M parameters

The VTB application can be write the M parameters for IsoNs application:

### *IsoNs Code*

```
M5
IF $_PARAM1=100
    ...
    ...
END_IF
```

### *VTB Code*

```
function start_macro() as void
    ISOV1_M_ack=1 ' set acknoleged for ISONS PC
    select ISOV1_M_cmd
        case 5
            ' M Function 5
            ISOV1_M_1=100
            ISOV1_status_m_run=0 ' Free the IsoNs application
        case else
            ISOV1_M_ack=0 ' M not found
    endselect
endfunction
```

### 5.3 M flags

In the VTB application, the M functions have the following flags:

<b>ISOV1_M_ack</b>	If set <b>True</b> (1) indicate to IsoNs PC application the M is performed If set <b>False</b> (0) the M is not found – Error
<b>ISOV1_M_cmd</b>	Contains the M number invoked by IsoNs PC application
<b>ISOV1_status_m_run</b>	The IsoNs Part Program PC applications, wait for this flag to value <b>False</b> (0) The value <b>True</b> (1) lock the part program When the time out is reached, an error is performed
<b>ISOV1_status_m_stop</b>	This flag is set <b>True</b> (1) from IsoNs when the <b>STOP REQUEST</b> is performed The VTB application must stop all internal M cycles in run and reset this flag

## 5.4 Example M3 M4 M5 start/stop Spindle

The IsoNs application can be write the spindle speed (S function) automatically in the VTB variable

**ISOV1\_generic(9)** (only if the IsoNs PC parameter **WR\_SPD9=1**)

When in the Gcode is present the Function S **value**, automatically value is written in the **ISOV1\_generic(9)** VTB array

The following sample refers to NG35 with NGIO for analog Spindle output

Note:

**MAXDAC** is a VTB define 2047 (Dac division)

**MAXSPEEDSPINDLE** is a VTB define the Rpm max of Spindle ex: 24000

**CwOut** is a bit of digital output for set the spindle in Cw mode

**CCwOut** is a bit of digital output for set the spindle in CCw mode

**SpindleStart** is a bit of digital output for Start Spindle

**VelSpindle** is a VTB long variable

### IsoNs Code

**S1200** // set to 1200 rpm

**M3** // start spindle in CW mode

### VTB Code

```
function Start_Macro() as char
  ISOV1_m_ACK=1
  select ISOV1_M_cmd
    case 3 ' start spindle in CW
      Cw=true ' set Cw mode
      Ccw=false
      ' calculate the Speed
      VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
      ng_dac(0,VelSpindle) ' Update the Speed on Dac out
      SpindleStart=true ' Start spindle
      ISOV1_status_m_run=0
    case 4 ' start spindle in CCW
      Cw=false
      Ccw=true ' set CCw mode
      ' calculate the Speed
      VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
      ng_dac(0,VelSpindle) ' Update the Speed on Dac out
      SpindleStart=true ' Start spindle
      ISOV1_status_m_run=0
    case 5 ' Stop spindle
      SpindleStart=false ' Stop spindle
      VelSpindle=0 ' set Speed to 0
      ng_dac(0,VelSpindle) ' Update the Speed on Dac out
      ISOV1_status_m_run=0
  case else
    ISOV1_m_ACK=0
  endselect
endfunction
```

## 6 Standard I/O

The following describes all virtual I/O exchange between VTB and application PC (the process is named ISOV1)

I/O	VTB VARIABLE	DESCRIPTION
External RUN request	ISOV1_ext_run	<p>This variable is set True (1) the PLC cycle request the <b>RUN</b> to IsoNs application. Ex: when external <b>RUN</b> button is pressed</p> <pre> <b>if</b> input_ext_run = 1 &amp;&amp; ISOV1_status_run=0 ' test external input run and not in run ' request of run IsoNs PC application     ISOV1_ext_run=1 ' the PC read this flag and RUN the part program <b>endif</b> </pre>
External STOP request	ISOV1_ext_stop	<p>This variable is set True (1) the PLC cycle request the <b>STOP</b> to IsoNs application. Ex: when external <b>STOP</b> button is pressed</p> <pre> <b>if</b> input_ext_stop = 1 &amp;&amp; ISOV1_status_run=1 ' test external input stop and in run ' request of stop IsoNs PC application     ISOV1_ext_stop=1 ' the PC read this flag and STOP the part program <b>endif</b> </pre>
External PAUSE request	ISOV1_ext_pausa	<p>This variable is set True (1) the PLC cycle request the <b>PAUSE</b> to IsoNs application. Ex: when external <b>PAUSE</b> button is pressed</p> <pre> <b>if</b> input_ext_pausa = 1 &amp;&amp; ISOV1_status_pausa=0 ' test external input pause and in not in pause ' request of pause IsoNs PC application     ISOV1_ext_pausa=1 ' the PC read this flag and PAUSE the part program <b>endif</b> </pre>
Negative limit switch for X axis	ISOV1_ext_fcm_x	<p>Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for X axis.</p> <pre>ISOV1_ext_fcm_x=InputLimtXneg</pre> <p>Automatically the VTB application stops the PartProgram</p>
Positive limit switch for X axis	ISOV1_ext_fcp_x	<p>Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for X axis.</p> <pre>ISOV1_ext_fcp_x=InputLimtXpos</pre> <p>Automatically the VTB application stops the PartProgram</p>
Negative limit switch for Y axis	ISOV1_ext_fcm_y	<p>Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for Y axis.</p> <pre>ISOV1_ext_fcm_y=InputLimtYneg</pre> <p>Automatically the VTB application stops the PartProgram</p>

Positive limit switch for Y axis	ISOV1_ext_fcp_y	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for Y axis. <b>ISOV1_ext_fcp_y=InputLimtYpos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for Z axis	ISOV1_ext_fcm_z	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for Z axis. <b>ISOV1_ext_fcm_z=InputLimtzneg</b>  Automatically the VTB application stops the PartProgram
Positive limit switch for Z axis	ISOV1_ext_fcp_z	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for Z axis. <b>ISOV1_ext_fcp_z=InputLimtZpos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for A axis	ISOV1_ext_fcm_a	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for A axis. <b>ISOV1_ext_fcm_a=InputLimtAneg</b>  Automatically the VTB application stops the PartProgram
Positive limit switch for A axis	ISOV1_ext_fcp_a	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for A axis. <b>ISOV1_ext_fcp_a=InputLimtApos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for B axis	ISOV1_ext_fcm_b	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for B axis. <b>ISOV1_ext_fcm_b=InputLimtBneg</b>  Automatically the VTB application stops the PartProgram
Positive limit switch for B axis	ISOV1_ext_fcp_b	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for B axis. <b>ISOV1_ext_fcp_b=InputLimtBpos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for C axis	ISOV1_ext_fcm_c	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for C axis. <b>ISOV1_ext_fcm_c=InputLimtCneg</b>  Automatically the VTB application stops the PartProgram



\Positive limit switch for C axis	ISOV1_ext_fcp_c	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for C axis. <b>ISOV1_ext_fcp_c=InputLimtCpos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for U axis	ISOV1_ext_fcm_u	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for U axis. <b>ISOV1_ext_fcm_u=InputLimtUneg</b>  Automatically the VTB application stops the PartProgram
Positive limit switch for U axis	ISOV1_ext_fcp_u	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for U axis. <b>ISOV1_ext_fcp_u=InputLimtUpos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for V axis	ISOV1_ext_fcm_v	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for V axis. <b>ISOV1_ext_fcm_v=InputLimtVneg</b>  Automatically the VTB application stops the PartProgram
Positive limit switch for V axis	ISOV1_ext_fcp_v	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for V axis. <b>ISOV1_ext_fcp_v=InputLimtVpos</b>  Automatically the VTB application stops the PartProgram
Negative limit switch for W axis	ISOV1_ext_fcm_w	Copy in this bit the digital input where is connected the <b>Negative</b> limit switch for W axis. <b>ISOV1_ext_fcm_w=InputLimtwneg</b>  Automatically the VTB application stops the PartProgram
Positive limit switch for W axis	ISOV1_ext_fcp_w	Copy in this bit the digital input where is connected the <b>Positive</b> limit switch for W axis. <b>ISOV1_ext_fcp_w=InputLimtWpos</b>  Automatically the VTB application stops the PartProgram
Acq Sensor	ISOV1_ext_acq	Copy in this bit the digital input where is connected the sensor for acquisitions <b>G102 function</b> <b>ISOV1_ext_acq=InputAcq</b>  Automatically the VTB application stops the Axes when the input is set

Stop Axes	ISOV1_stop_assi	<p>When this bit is set, the Gcode application is stopped This bit is alternative to <b>ISOV1_ext_stop</b> for example can be used for forcing stop axes by external events</p> <pre>if InputForceStop = 1 &amp;&amp; ISOV1_status_run=1 ' Force a Stop axes     ISOV1_stop_assi=1 ' the PC part program is stopped endif</pre>
Stop Axes with emergency	ISOV1_stop_emcy	<p>When this bit is set, the Gcode application is stopped and ALARM EMERGENCY is invoked All drives are disbled</p> <pre>if InputForceEmcy = 1 &amp;&amp; ISOV1_status_run=1 ' Force a Stop axes with EMCY     ISOV1_stop_EMCY=1 ' the PC part program is stopped with error invoked endif</pre>
Emergency Input	ISOV1_ext_emcy	<p>Copy in this bit the digital input where is connected the <b>Emergency</b> general output (typical N.C.)</p> <pre>ISOV1_ext_emcy=!InputGeneralEmcy</pre> <p>Automatically the VTB application stops the PartProgram and a Alarm is invoked</p>
Manual JOG X -	ISOV1_ext_jogm_x	<p>Copy in this bit the digital input where is connected the <b>External Jog X - Button</b></p> <pre>ISOV1_ext_jogm_x=JogExtXm</pre> <p>The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application</p>
Manual JOG X +	ISOV1_ext_jogp_x	<p>Copy in this bit the digital input where is connected the <b>External Jog X + Button</b></p> <pre>ISOV1_ext_jogp_x=JogExtXp</pre> <p>The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application</p>
Manual JOG Y -	ISOV1_ext_jogm_y	<p>Copy in this bit the digital input where is connected the <b>External Jog Y - Button</b></p> <pre>ISOV1_ext_jogm_y=JogExtYm</pre> <p>The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application</p>
Manual JOG Y +	ISOV1_ext_jogp_y	<p>Copy in this bit the digital input where is connected the <b>External Jog Y + Button</b></p> <pre>ISOV1_ext_jogp_y=JogExtYp</pre> <p>The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application</p>
Manual JOG Z -	ISOV1_ext_jogm_z	<p>Copy in this bit the digital input where is connected the <b>External Jog Z - Button</b></p> <pre>ISOV1_ext_jogm_z=JogExtZm</pre> <p>The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application</p>

Manual JOG Z +	ISOV1_ext_jogp_z	Copy in this bit the digital input where is connected the <b>External Jog Z + Button</b> <b>ISOV1_ext_jogp_z=JogExtZp</b> The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG A -	ISOV1_ext_jogm_a	Copy in this bit the digital input where is connected the <b>External Jog A - Button</b> <b>ISOV1_ext_jogm_a=JogExtAm</b> The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG A +	ISOV1_ext_jogp_a	Copy in this bit the digital input where is connected the <b>External Jog A + Button</b> <b>ISOV1_ext_jogp_a=JogExtAp</b> The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG B -	ISOV1_ext_jogm_b	Copy in this bit the digital input where is connected the <b>External Jog B - Button</b> <b>ISOV1_ext_jogm_b=JogExtBm</b> The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG B +	ISOV1_ext_jogp_b	Copy in this bit the digital input where is connected the <b>External Jog B + Button</b> <b>ISOV1_ext_jogp_b=JogExtBp</b> The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG C -	ISOV1_ext_jogm_c	Copy in this bit the digital input where is connected the <b>External Jog C - Button</b> <b>ISOV1_ext_jogm_c=JogExtCm</b> The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG C +	ISOV1_ext_jogp_c	Copy in this bit the digital input where is connected the <b>External Jog C + Button</b> <b>ISOV1_ext_jogp_c=JogExtCp</b> The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG U -	ISOV1_ext_jogm_u	Copy in this bit the digital input where is connected the <b>External Jog U - Button</b> <b>ISOV1_ext_jogm_u=JogExtUm</b> The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG U +	ISOV1_ext_jogp_u	Copy in this bit the digital input where is connected the <b>External Jog U + Button</b> <b>ISOV1_ext_jogp_u=JogExtUp</b> The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application

Manual JOG V -	ISOV1_ext_jogm_v	Copy in this bit the digital input where is connected the <b>External Jog V - Button</b>  ISOV1_ext_jogm_v=JogExtVm  The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG V +	ISOV1_ext_jogp_v	Copy in this bit the digital input where is connected the <b>External Jog V + Button</b>  ISOV1_ext_jogp_v=JogExtVp  The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG W -	ISOV1_ext_jogm_w	Copy in this bit the digital input where is connected the <b>External Jog W - Button</b>  ISOV1_ext_jogm_w=JogExtWm  The Axis is moved in <b>NEGATIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG W +	ISOV1_ext_jogp_w	Copy in this bit the digital input where is connected the <b>External Jog W + Button</b>  ISOV1_ext_jogp_w=JogExtWp  The Axis is moved in <b>POSITIVE</b> direction at <b>VJOG</b> set by IsoNs application
Manual JOG - Generic Axis	ISOV1_ext_jogm	This bit is used to move in manual mode negative direction a generic axis. The axis is set in the VTB variable <b>ISOV1_asse_man</b> . Copy the input jog button  ISOV1_ext_jogm=JogExtM  ISOV1_asse_man=0 → X axis is set ISOV1_asse_man=1 → Y axis is set ISOV1_asse_man=2 → Z axis is set ISOV1_asse_man=3 → A axis is set ISOV1_asse_man=4 → B axis is set ISOV1_asse_man=5 → C axis is set ISOV1_asse_man=6 → U axis is set ISOV1_asse_man=7 → V axis is set ISOV1_asse_man=8 → W axis is set
Manual JOG + Generic Axis	ISOV1_ext_jogp	This bit is used to move in manual mode positive direction a generic axis. The axis is set in the VTB variable <b>ISOV1_asse_man</b> (see above). Copy the input jog button  ISOV1_ext_jogp=JogExtP
Homing X Switch	ISOV1_ext_fcz_x	Copy in this bit the digital input where is connected the <b>Homing X Switch</b>  ISOV1_ext_fcz_x=InputHoming_X  This bit is used when the command axis home is performed

<b>Homing Y Switch</b>	<b>ISOV1_ext_fcz_y</b>	Copy in this bit the digital input where is connected the <b>Homing Y Switch</b> <b>ISOV1_ext_fcz_y=InputHoming_Y</b> This bit is used when the command axis home is performed
<b>Homing Z Switch</b>	<b>ISOV1_ext_fcz_z</b>	Copy in this bit the digital input where is connected the <b>Homing Z Switch</b> <b>ISOV1_ext_fcz_z=InputHoming_Z</b> This bit is used when the command axis home is performed
<b>Homing A Switch</b>	<b>ISOV1_ext_fcz_a</b>	Copy in this bit the digital input where is connected the <b>Homing A Switch</b> <b>ISOV1_ext_fcz_a=InputHoming_A</b> This bit is used when the command axis home is performed
<b>Homing B Switch</b>	<b>ISOV1_ext_fcz_b</b>	Copy in this bit the digital input where is connected the <b>Homing B Switch</b> <b>ISOV1_ext_fcz_b=InputHoming_B</b> This bit is used when the command axis home is performed
<b>Homing C Switch</b>	<b>ISOV1_ext_fcz_c</b>	Copy in this bit the digital input where is connected the <b>Homing C Switch</b> <b>ISOV1_ext_fcz_c=InputHoming_C</b> This bit is used when the command axis home is performed
<b>Homing U Switch</b>	<b>ISOV1_ext_fcz_u</b>	Copy in this bit the digital input where is connected the <b>Homing U Switch</b> <b>ISOV1_ext_fcz_u=InputHoming_U</b> This bit is used when the command axis home is performed
<b>Homing V Switch</b>	<b>ISOV1_ext_fcz_v</b>	Copy in this bit the digital input where is connected the <b>Homing V Switch</b> <b>ISOV1_ext_fcz_v=InputHoming_V</b> This bit is used when the command axis home is performed
<b>Homing W Switch</b>	<b>ISOV1_ext_fcz_W</b>	Copy in this bit the digital input where is connected the <b>Homing W Switch</b> <b>ISOV1_ext_fcz_w=InputHoming_W</b> This bit is used when the command axis home is performed

## 7 Status Word

The VTB application uses a Status Word for communicate to PC application the status. Normally this status is set by internal VTB functions, but the application can read the status. This Status Word is Bit mapped

VTB Bit Name	DESCRIPTION
ISOV1_status_run	Setted to <b>True</b> (1) when the IsoNs application run the GCODE
ISOV1_status_move	Setted to <b>True</b> (1) when the axes are in movements
ISOV1_status_pausa	Setted to <b>True</b> (1) when the IsoNs application is in PAUSE MODE
ISOV1_status_error	Setted to <b>True</b> (1) when the Alarm is present (EMCY drives, Emergency button etc.)
ISOV1_status_rzero	Setted to <b>True</b> (1) when one axis is during search homing Home command by IsoNs application
ISOV1_status_rsens	Setted to <b>True</b> (1) when the acquisition command is performed set to <b>False</b> (0) when the acquisition is finished G102 function by IsoNs application
ISOV1_status_para_upd	Setted to <b>True</b> (1) when the PC application updates the VTB internal parameters. This flag is not reset automatically, but when read, you can reset. In the some case is necessary when the PC updates the VTB parameters, perform the operations with the new parameters.
ISOV1_status_m_run	Setted to <b>True</b> (1) when the M cycle is in execution. Reset by M functions management, <b>See chapter 5 - "USE the M functions to internal the CNC "</b>
ISOV1_status_m_stop	This flag is Setted <b>True</b> (1) from IsoNs when the <b>STOP REQUEST</b> is performed The VTB application must stop all internal M cycles in run and reset this flag
ISOV1_status_home_x	This flag is Setted <b>True</b> (1) when the relative axis has performed the homing procedure
ISOV1_status_home_y	
ISOV1_status_home_z	
ISOV1_status_home_a	
ISOV1_status_home_b	
ISOV1_status_home_c	
ISOV1_status_home_u	
ISOV1_status_home_v	
ISOV1_status_home_W	
ISOV1_status_enable_x	This flag is Setted <b>True</b> (1) when the relative axis is enabled or <b>False</b> if is Disabled
ISOV1_status_enable_y	
ISOV1_status_enable_z	
ISOV1_status_enable_a	
ISOV1_status_enable_b	
ISOV1_status_enable_c	
ISOV1_status_enable_u	
ISOV1_status_enable_v	
ISOV1_status_enable_w	

## 8 PLC I/O management

The VTB application can use the PLC I/O in very simple mode.

First, is necessary, load in the application the relative I/O object (**See chapter 4 - "Insert the I/O"**)

The VTB application use e definition Bit to management the I/O.

The max number of I/O is:

**256 Digital Inputs**  
**256 Digital outputs**

The I/O are a blocks of 16 bit

**If is used the NGQ-NGQx module with 11 digital Inputs, you can not use the Inputs 11 to 15 (or relative to block 27 to 31 etc.) because these are not present in the board NGQ-NGQx**

**You can not use the outputs 8 to 15 (or relative to block 24 to 31 etc)**

**If is used the NGIO-NGMIO module with 14 digital Outputs, you can not use the Outputs 14,15 (or relative to block 30,31 etc) because these are not present in the board NGIO-NGMIO**

**You can use always the outputs 14,15 for internal flag**

### 8.1 Defined bit digital Inputs

ISOV1.inp0 → Digital Inputs 1  
ISOV1.inp1 → Digital Inputs 2  
ISOV1.inp2 → Digital Inputs 3  
.  
ISOV1.inp255 → Digital Inputs 255

### 8.2 Defined bit digital Outputs

ISOV1.out0 → Digital Outputs 1  
ISOV1.out1 → Digital Outputs 2  
ISOV1.out2 → Digital Outputs 3  
.  
ISOV1.out255 → Digital Outputs 255

Ex:

```
if ISOV1.inp0 = 1 && ISOV1.inp1=0
    ISOV1.out0=true 'set out 0
endif
```

## 9 FEED External Override

By VTB application is possible management a potentiometer for Axes feed control.

The potentiometer must be connected at analog input.

The Variable is :

**ISOV1\_vper**

The value range is 0 to 1024, you can copy the analog input directly in this variable

Ex:

The analog input is Number 1

Insert this code in main cycle or task plc cycle

```
ISOV1_vper=ng_adc(0)
```

If is used the NGM EVO,NGQ,NGQx analog inputs (12 bit – range from 0 to 4095)

divide by 4 the **ISOV1\_vper**

```
ISOV1_vper=ng_adc(0)
```

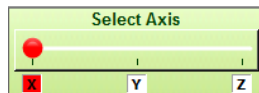
```
ISOV1_vper=ISOV1_vper/4
```

## 10 Electronic HandWheel

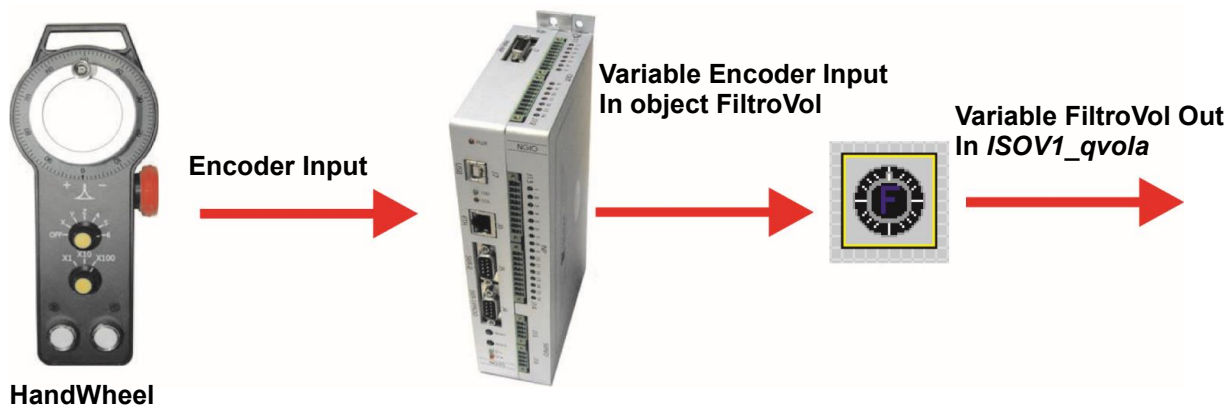
This chapter describe the management Electronic HandWheel.

### VTB Variable

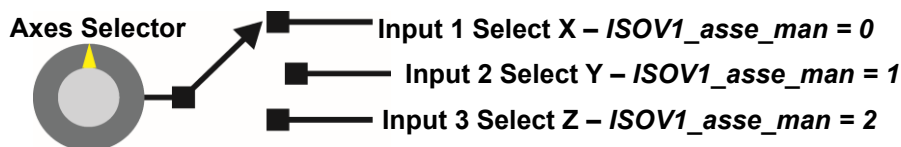
**ISOV1\_soft\_sel\_man** → If Set at value 1 the Axes Virtual Selector on PC is Enabled  
 → If Set at value 0 Virtual Selector on PC is Disabled



**ISOV1\_qvola** → Electronic HandWheel incremental encodr value  
 Generally this value is got by a FiltroVol Object, because this Object smoothing the handwheel pulse.

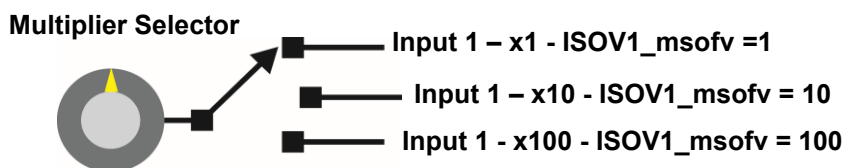


**ISOV1\_asse\_man** → Select the axes to move with Handwheel  
 0 → X            1 → Y            2 → Z  
 3 → A            4 → B            5 → C  
 6 → U            7 → V            8 → W  
 Generally this Variable is got by digital input from selector Axis  
 Only if the variable **ISOV1\_soft\_sel\_man=0**



**ISOV1\_soft\_sel\_man** → If Set to 0, the **ISOV1\_asse\_man** is set by VTB code  
 If Set to 1, The **ISOV1\_asse\_man** is set by IsoNs PC (virtual selector)

**ISOV1\_msofv** → Set the multiplier for pulse (generally x1 x10 x100)  
 If **ISOV1\_soft\_sel\_man=1** This variable is set by IsoNs PC  
 This Variable is combined with parameter IsoNs **DSOFV\_X, \_Y** etc  
 The **DSOFV\_** generally contains the pulse encoder for revolution multiplier 4  
 Ex: i/g Handwheel=100 **DSOFV\_=400**





## VTB Example

The system is NG35+NGIO with following connection:

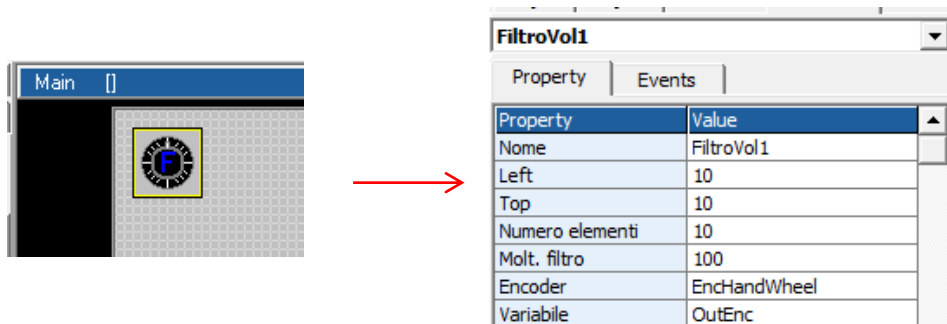
**Ch0 Encoder** → **Encoder HandWheel**  
**Digital Input 0** → **Selector Axis X**  
**Digital Input 1** → **Selector Axis Y**  
**Digital Input 2** → **Selector Axis Z**  
**Digital Input 3** → **Selector Multiplier x1**  
**Digital Input 4** → **Selector Multiplier x10**  
**Digital Input 5** → **Selector Multiplier x100**  
**Digital Input 6** → **Input JOG Positive**  
**Digital Input 7** → **Input JOG Negative**

Variable used:

**EncHandWheel** → **Long**  
**OutEnc** → **Long**  
**Input** → **Long**

Insert the Object FiltroVol in Main and set the parameters :

**Objects**→**Motor Control**→**CFiltrovol.vco**



Insert the following code in **Init Task Plc**:

```
ISOV1_soft_sel_man=0      ' Enable the internal VTB selector
FiltroVol1.enable=1      ' Enable filter
```

Insert the following code in **Task Plc**:

```

-----
' Read the HandWheel Encoder,
' Put in EncHandWheel Variable
' Note: the () set the pointer – Very important
-----
ng_enc(0,EncHandWheel())

-----
'Set the multiplier
-----
if ISOV1.inp3          'multiplier x1
    ISOV1_msofv=1
endif
if ISOV1.inp4          'multiplier x10
    ISOV1_msofv=10
endif
if ISOV1.inp5          'multiplier x100
    ISOV1_msofv=100
endif

```

```
'-----  
'Set the Axis  
'-----  
if ISOV1.inp0           'Axis X  
    ISOV1_asse_man=0  
endif  
if ISOV1.inp1           'Axis Y  
    ISOV1_asse_man=1  
endif  
if ISOV1.inp2           'Axis Z  
    ISOV1_asse_man=2  
endif  
  
'-----  
'Update the Handwheel from FiltroVol Object  
'-----  
if !ISOV1_status_run  
    FiltroVol1.enable=true           'enable the handwheel filter  
    ISOV1_qvola=OutEnc  
else  
    FiltroVol1.enable=false         'disable the handwheel filter  
    ISOV1_qvola=0  
endif  
  
'-----  
'Update the Jog Input  
'-----  
ISOV1_ext_jogp=ISOV1.inp6  
ISOV1_ext_jogm=ISOV1.inp7
```

## 11 Machine Parameters

With VTB application you can read and write the machine Parameters.

Generally this function is not necessary, because the machine parameters are managed from IsoNs VTB application system, but if you want read or write the parameters this is possible.

All parameters are in the array **ISOV1\_PARA**

The array dimension depends of Axes number.

All parameters are type LONG (32 bit signed)

### ISOV1\_PARA array organization

0-49 General Parameters	50-99 Axis X Parameters	100-149 Axis Y Parameters	....
-------------------------	-------------------------	---------------------------	------

#### 11.1 General Parameters

Idx ISOV1_Para	Name	Description
0	VRIPOS	Axes Velocity repositioning after Pause (if in IsoNs is not present the Macro <b>GOPAUSE</b> )
1	SGLP	Threshold Edge
2	SGLR	Threshold of Arc error
3	ACQ_MODE	Acquisition type <b>0 → Speed</b> <b>1 → step by step</b>
4	ACQ_VEL	Feed Acquisition
5	ACQ_STEP	Step length for acquisition type 1
6	ACQ_TIME	Time for acquisition type 1
7	ACC_QSTOP	Acceleration for Quick stop
8	JERK	Jerk acceleration
9	NOSHORT	Remove the short linee
Free 10 to 49	.....	.....

## 11.2 Axes X Parameters

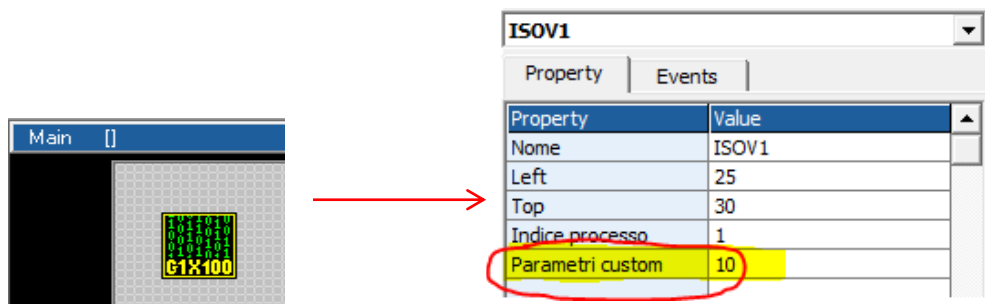
(50 x blocks)

Idx ISOV1_Para	Name	Description
50	VJOG	JOG Feed Axis (mm/min)
51	ACC_JOG	Acceleration JOG Axis (count)
52	LIMITE_N_	Negative Software Limit Axis
53	LIMITE_P_	Positive Software Limit Axis
54	DSOFV	Jog Handwheel Divisor Axis
55	RZERO_MODE	Homing mode
56	RZERO_OFFSET	Homing Offset
57	RZERO_PRESET	Homing Preset
58	RZERO_VEL	Homing High Feed Axis
59	RZERO_VELF	Homing Low Feed Axis
60	RZERO_ACC	Homing Acceleration Axis
61	MSOF	Number Count/Revolution Axis
62	DSOF	Distance for One Revolution Encoder Axis
63	GANTRY	Gantry Axis
64	SGL_3D	Edge Threshold Axis 3D
65	BACKSLASH	Axis Backslash (um)
66	TBCK	Axis Time Backslash (TAU)
67	TSHF	Speed Shift
<b>Free 67 to 79</b>		
<b>The following Parameters are present only for analog +/- 10V drives</b>		
80	PID_KP	(Proportional Costant) Axis
81	PID_KI	(Integral Costant) Axis
82	PID_KV	(Feed Costant) Axis
83	PID_I_LIMIT	IL (Integration Limit) Axis
84	PID_DIV	PID Magnitude Axis
85	PID_SERVO	Servo Error Axis (um)
86	PID_TIME_SERVO	During Time Servo Error Axis
87	PID_DIR	Analog Direction Axis
88	PID_OFFS_ANA	Analog Digital Offset Axis
<b>Free 88 to 99</b>		
<b>100 to 149 Axis Y parameters</b>		
<b>150 to 199 Axis Z parameters (if present)</b>		
<b>200 to 249 Axis A parameters (if present)</b>		
<b>250 to 299 Axis B parameters (if present)</b>		
<b>300 to 349 Axis C parameters (if present)</b>		
<b>350 to 399 Axis U parameters (if present)</b>		
<b>400 to 449 Axis V parameters (if present)</b>		
<b>450 to 499 Axis W parameters (if present)</b>		

### 11.3 Custom Parameters

With IsoNs PC Application is possible declare a Custom machine Parameters.  
This parameters can be read by VTB application.

First is necessary declare in the Object ISOVIRTUAL (ISOV1) the MAX NUMBER OF CUSTOM PARAMETERS in the application.



The Custom parameters are available at the following address:

**ISOV1\_PARA(ISOV1\_P\_CUSTOM)**

Where **ISOV1\_P\_CUSTOM** is a define automatically calculate by VTB compiler

At this address is present the first custom parameter configured:

the second is present in **ISOV1\_PARA(ISOV1\_P\_CUSTOM+1)** etc.

The real value is:

**ISOV1\_P\_CUSTOM = 50 \* Axes Number + 50**

**Ex: 4 Axes configured and 3 custom parameters**

CUSTOM_1	Custom parameter 1	100
CUSTOM_2	Custom parameter 2	100
CUSTOM_3	Custom parameter 3	100

The **CUSTOM\_1** must have Address **250**

The **CUSTOM\_2** must have Address **251**

The **CUSTOM\_3** must have Address **252**

**CustomPar1=ISOV1\_PARA(ISOV1\_P\_CUSTOM)**

**CustomPar2=ISOV1\_PARA(ISOV1\_P\_CUSTOM+1)**

**CustomPar3=ISOV1\_PARA(ISOV1\_P\_CUSTOM+2)**

## 12 Alarms bit mapped

All alarms are bit mapped. If the application, sets the relative bit, the alarm is activated. Generally these alarms are activated by system IsoNs VTB automatically when the event is occurred. The application can read and write the single bit.

Bit Name	Alarm Description
ISOV1.allarm0	X Negative Limit Reached
ISOV1.allarm1	X Positive Limit Reached
ISOV1.allarm2	Y Negative Limit Reached
ISOV1.allarm3	Y Positive Limit Reached
ISOV1.allarm4	Z Negative Limit Reached
ISOV1.allarm5	Z Positive Limit Reached
ISOV1.allarm6	A Negative Limit Reached
ISOV1.allarm7	A Positive Limit Reached
ISOV1.allarm8	B Negative Limit Reached
ISOV1.allarm9	B Positive Limit Reached
ISOV1.allarm10	C Negative Limit Reached
ISOV1.allarm11	C Positive Limit Reached
ISOV1.allarm12	U Negative Limit Reached
ISOV1.allarm13	U Positive Limit Reached
ISOV1.allarm14	V Negative Limit Reached
ISOV1.allarm15	V Positive Limit Reached
ISOV1.allarm16	W Negative Limit Reached
ISOV1.allarm17	W Positive Limit Reached
ISOV1.allarm18	X SERVO-EMERGENCY
ISOV1.allarm19	Y SERVO-EMERGENCY
ISOV1.allarm20	Z SERVO-EMERGENCY
ISOV1.allarm21	A SERVO-EMERGENCY
ISOV1.allarm22	B SERVO-EMERGENCY
ISOV1.allarm23	C SERVO-EMERGENCY
ISOV1.allarm24	U SERVO-EMERGENCY
ISOV1.allarm25	V SERVO-EMERGENCY
ISOV1.allarm26	W SERVO-
ISOV1.allarm27	General EMERGENCY Activated
ISOV1.allarm28	Acquisition Error
ISOV1.allarm29	Short line found (active if the NO_SHORT=2)
ISOV1.allarm30	Free
ISOV1.allarm31	Free
ISOV1.allarm32	X Axis Configuration error (CanOpen or Ethercat)

ISOV1.allarm32	Y Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm34	Z Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm35	A Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm36	B Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm37	C Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm38	U Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm39	V Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm40	W Axis Configuration error (CanOpen or Ethercat)
ISOV1.allarm41	Free
ISOV1.allarm42	Free
....	
ISOV1.allarm255	Free

### 13 CanOpen Alarms

The management of Canopen alarms (EMCY-OBJ), is performed directly by Axis object loaded.  
When the alarm is present, the single bit is set:

**X SERVO-EMERGENCY**

**Y SERVO-EMERGENCY**

**Y SERVO-EMERGENCY**

etc

In the `ISOV1_last_allarm(node axis)` you can find the CanOpen Code error

## 14 Examples

Following VTB application examples

### 14.1 NGQ-NGM EVO 3 Axes Step/Dir

Link RS32 on COM1 NGQ-NGM EVO

#### Digital Inputs

Digital Input 1	→ Switch Home X (N.C.)
Digital Input 2	→ Switch Home Y (N.C.)
Digital Input 3	→ Switch Home Z (N.C.)
Digital Input 4	→ GENERAL EMERGENCY INPUT (N.C.)
Digital Input 5	→ Button JOG X+ (N.O.)
Digital Input 6	→ Button JOG X- (N.O.)
Digital Input 7	→ Button JOG Y+ (N.O.)
Digital Input 8	→ Button JOG Y- (N.O.)
Digital Input 9	→ Button JOG Z+ (N.O.)
Digital Input 10	→ Button JOG Z- (N.O.)

#### Analog Inputs

Analog Inputs 1	→ Feed Potentiometer Override Axes
-----------------	------------------------------------

For enable this Override, you must select the Enable the “Ext OW” from IsoNs Interface”



#### Digital Outputs

Digital outputs 1	→ X axis enabled
Digital outputs 2	→ Y axis enabled
Digital outputs 3	→ Z axis enabled
Digital outputs 4	→ CNC Error

#### Axes Outputs

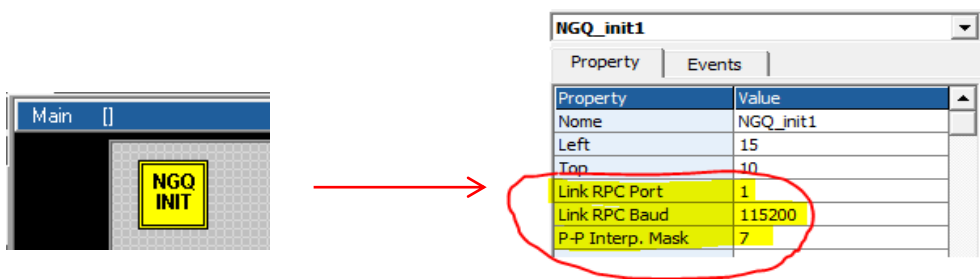
Step/Dir Ch 1	→ X
Step/Dir Ch 2	→ Y
Step/Dir Ch 3	→ Z

1) Open new project VTB and select NGQ hardware or NGM EVO (the example is on NGQ)  
Select 4 Ms sample

Start Page:	<input type="text" value="1"/>
Sample:	<input type="text" value="4"/> mS
Task Time:	<input type="text" value="5"/> x 4 = <input type="text" value="20"/> mS
Screensave:	<input checked="" type="checkbox"/> Enable <input type="text" value="30"/> sec



2) Set link on COM1(or COM2) NGQ-NGM EVO and PP Interp mask on 7 (X Y Z channel enabled)



3) Insert object ISOVIRTUAL and set the default properties

Objects → Iso\_Ns → IsoVirtual.vco

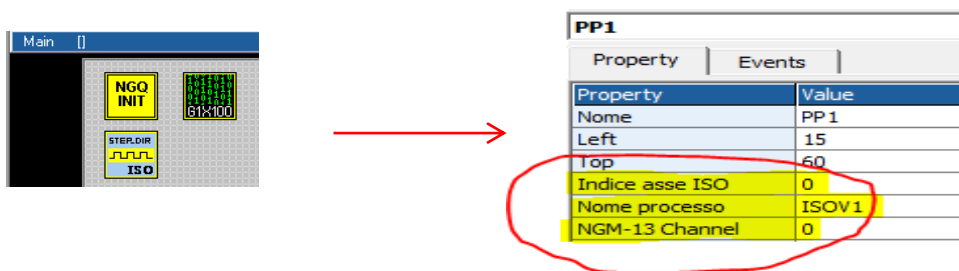


4) Insert Axis X ISOPP (The NGM EVO object is the same of the NGQ)

Objects → Iso\_Ns → IsoPP.vco

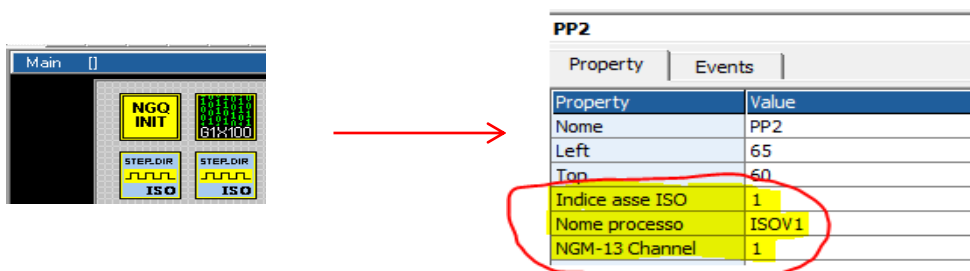


5) Set the following properties



6) Insert Axis Y ISOPP and set the following properties

Objects → Iso\_Ns → IsoPP.vco



7) Insert Axis Z ISOPP and set the following properties

Objects → Iso\_Ns → IsoPP.vco

PP3	
Property	Events
Property	Value
Nome	PP3
Left	115
Top	88
Indice asse ISO	2
Nome processo	ISOV1
NGM-13 Channel	2

8) Insert the ISO I/O

Objects → Iso\_Ns → Iso-IO.vco

NGM-IO  
ISO

- NGM-IO - \$Rev 1.0.1

8) Set the following properties

PP3	
Property	Events
Property	Value
Nome	PP3
Left	115
Top	88
Indice asse ISO	2
Nome processo	ISOV1
NGM-13 Channel	2

9) Insert the object ObjInterpola

Objects → Motor Control → CobjInterpola.vco

Select Components

OBJ

- Interpolatore - \$Rev 1.1.2

10) Set the following properties

Interpola1	
Property	Events
Property	Value
Nome	Interpola1
Left	65
Top	110
N.assi	3
N.tratti	16
Vper	1024
Div. Vper	1024
Abilita arco	1

## 11) Insert in the Task main (or task plc) the code management

```

Init Task PLC | Task PLC
1 | '-----
2 | ' Read The Analog Inputs
3 | ' For override control
4 | '-----
5 | ISOV1_vper=ng_adc(0)
6 | '-----
7 | ' Test Digital Inputs
8 | '-----
9 | ISOV1_ext_fcz_x=!ISOV1.inp0 ' Homing switch X
10 | ISOV1_ext_fcz_y=!ISOV1.inp1 ' Homing switch Y
11 | ISOV1_ext_fcz_z=!ISOV1.inp2 ' Homing switch Z
12 | ISOV1_ext_emcy=!ISOV1.inp3 ' General emergency
13 | ISOV1_ext_jogp_x=ISOV1.inp4 ' JOG X +
14 | ISOV1_ext_jogm_x=ISOV1.inp5 ' JOG X -
15 | ISOV1_ext_jogp_y=ISOV1.inp6 ' JOG Y +
16 | ISOV1_ext_jogm_y=ISOV1.inp7 ' JOG Y -
17 | ISOV1_ext_jogp_z=ISOV1.inp8 ' JOG Z +
18 | ISOV1_ext_jogm_z=ISOV1.inp9 ' JOG Z -
19 | '-----
20 | ' Test Digital Outputs
21 | '-----
22 | ISOV1.out0=ISOV1_status_enable_x ' X enabled
23 | ISOV1.out1=ISOV1_status_enable_y ' Y enabled
24 | ISOV1.out2=ISOV1_status_enable_z ' Z enabled
25 | ISOV1.out2=ISOV1_status_error ' CNC error

```

```

'-----
' Read The Analog Inputs
' For override control
'-----
ISOV1_vper=ng_adc(0)
'-----
' Test Digital Inputs
'-----
ISOV1_ext_fcz_x=!ISOV1.inp0 ' Homing switch X
ISOV1_ext_fcz_y=!ISOV1.inp1 ' Homing switch Y
ISOV1_ext_fcz_z=!ISOV1.inp2 ' Homing switch Z
ISOV1_ext_emcy=!ISOV1.inp3 ' General emergency
ISOV1_ext_jogp_x=ISOV1.inp4 ' JOG X +
ISOV1_ext_jogm_x=ISOV1.inp5 ' JOG X -
ISOV1_ext_jogp_y=ISOV1.inp6 ' JOG Y +
ISOV1_ext_jogm_y=ISOV1.inp7 ' JOG Y -
ISOV1_ext_jogp_z=ISOV1.inp8 ' JOG Z +
ISOV1_ext_jogm_z=ISOV1.inp9 ' JOG Z -
'-----
' Test Digital Outputs
'-----
ISOV1.out0=ISOV1_status_enable_x ' X enabled
ISOV1.out1=ISOV1_status_enable_y ' Y enabled
ISOV1.out2=ISOV1_status_enable_z ' Z enabled
ISOV1.out2=ISOV1_status_error ' CNC error

```

## 14.2 NG35+2xNGIO Axes 3 Analog +/- 10V and handwheel

Link ETHERNET IP: "10.0.0.80" (default)

The following project used a handwell connected to Ch 2 second NGIO encoder inputs , selector for JOG AXES and spindle.

To enable a selector is necessary insert in the init TASK PLC the following code:

ISOV1\_soft\_sel\_man=0 ' Enable the internal VTB selector

### Digital Inputs

Digital Input 1	→ Switch Home X (N.C.)
Digital Input 2	→ Switch Home Y (N.C.)
Digital Input 3	→ Switch Home Z (N.C.)
Digital Input 4	→ GENERAL EMERGENCY INPUT (N.C.)
Digital Input 5	→ Selector JOG X (N.O.)
Digital Input 6	→ Selector JOG Y (N.O.)
Digital Input 7	→ Selector JOG Z (N.O.)
Digital Input 8	→ Button JOG - (N.O.)
Digital Input 9	→ Button JOG + (N.O.)
Digital Input 10	→ Handwheel Speed x1
Digital Input 11	→ Handwheel Speed x10
Digital Input 12	→ Handwheel Speed x100

### Analog Inputs

Analog Inputs 1 → Feed Potentiometer Override Axes

For enable this Override, you must select the Enable the "Ext OW" from IsoNs Interface"



### Digital Outputs

Digital outputs 1	→ X axis enabled
Digital outputs 2	→ Y axis enabled
Digital outputs 3	→ Z axis enabled
Digital outputs 4	→ CNC Error
Digital outputs 5	→ Spindle start/stop
Digital outputs 6	→ Spindle CW (M3)
Digital outputs 7	→ Spindle CCW (M4)

### Axes Inputs

Encoder Ch 1 (first NGIO)	→ X encoder feedback
Encoder Ch 2 (first NGIO)	→ Y encoder feedback
Encoder Ch 1 (second NGIO)	→ Z encoder feedback
Encoder Ch 2 (second NGIO)	→ HandWheel encoder

### Axes Outputs

Analog out 1 (first NGIO)	→ X Speed +/-10V
Analog out 2 (first NGIO)	→ Y Speed +/-10V
Analog out 1 (first NGIO)	→ Z Speed +/-10V
Analog out 2 (first NGIO)	→ SPEED Spindle

- 1) Open new project VTB and select NG35  
Select 1 Ms sample

Start Page:

Sample:  mS

Task Time:  x 1 =  mS

Screensave:  Enable  sec

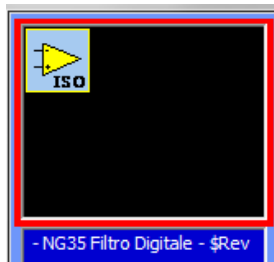
- 2) Insert object ISOVIRTUAL and set the default properties

Objects → Iso\_Ns → IsoVirtual.vco



- 3) Insert Axis X ISOPID

Objects → Iso\_Ns → IsoPid.vco



- 4) Set the following properties



PID1	
Property	Events
Property	Value
Nome	PID1
Left	60
Top	10
Indice asse ISO	0
Nome processo	ISOV1
Indice asse NG-IO	0

- 5) Insert Axis Y ISOPID and set the following properties

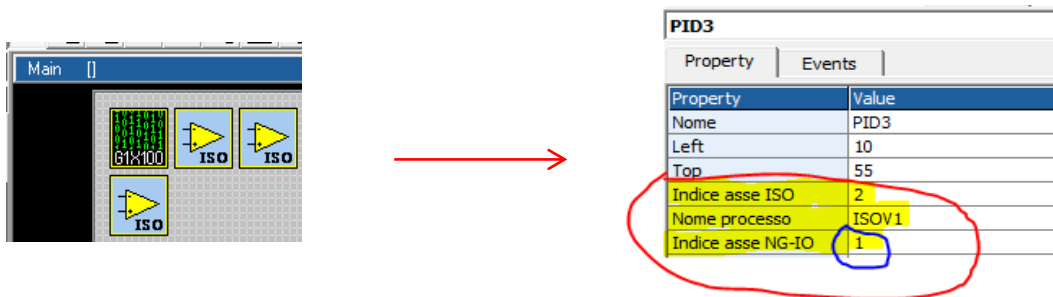
Objects → Iso\_Ns → IsoPid.vco



PID2	
Property	Events
Property	Value
Nome	PID2
Left	100
Top	10
Indice asse ISO	1
Nome processo	ISOV1
Indice asse NG-IO	0

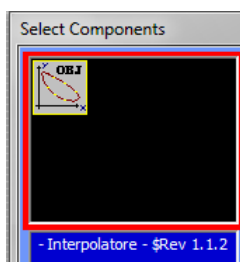
6) Insert Axis Z ISOPID and set the following properties

Objects → Iso\_Ns → IsoPid.vco

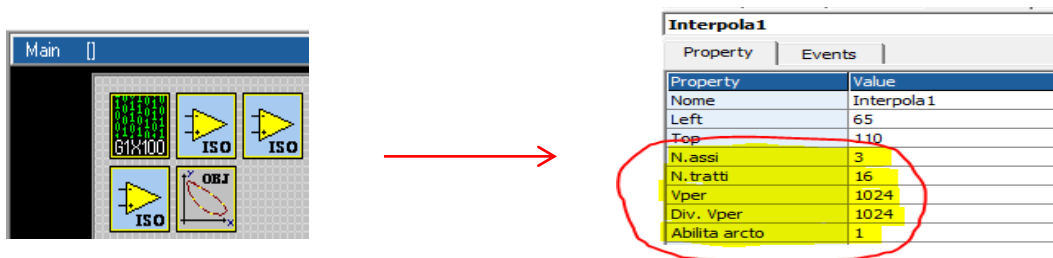


7) Insert the object ObjInterpola

Objects → Motor Control → CobjInterpola.vco



8) Set the following properties

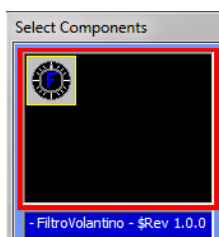


9) Declare a following Global Variables

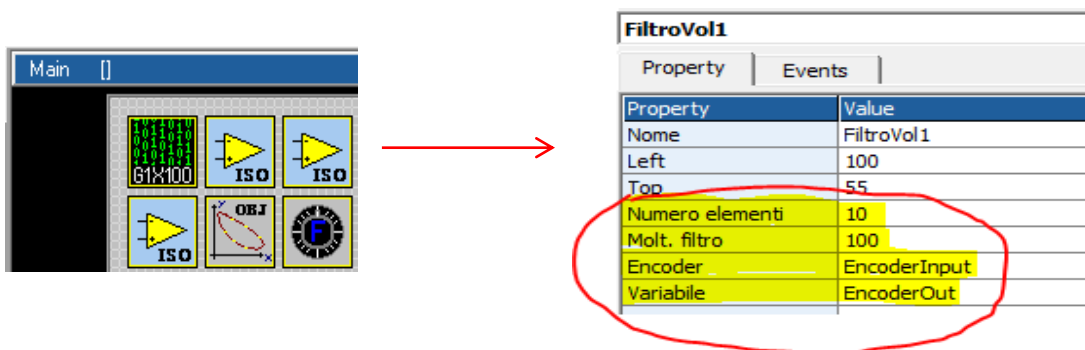
Internal VAR	Bit VAR	Define	Static VAR	VSD VAR	Fixed VA
			No	EXP	
Variable	Type	Shared	Export in Class		
EncoderInput	LONG	No			
EncoderOut	LONG	No			

10) Insert the FiltroVol Object for handwheel

Objects → Motor Control → CfiltroVol.vco



### 11) Set the following properties



FiltroVol1	
Property	Events
Property	Value
Nome	FiltroVol1
Left	100
Top	55
Numero elementi	10
Molt. filtro	100
Encoder	EncoderInput
Variable	EncoderOut

### 12) Insert in "Init task PLC " the entry point for M Functions and the enable external selector

TASK PLC Code	
Init Task PLC	Task PLC
1	ISOV1_soft_sel_man=0 ' Enable the internal VTB selector
2	ISOV1_Start_m=StartMacro ' entry point MACRO management

### 13) Insert 2 Define Constant in Global Variables → Define

The MAXSPEEDSPINDLE depend to MAX rpm at 10 Volt your spindle

Internal VAR	Bit VAR	Define	Static VAR
Variable		Type	
MAXDAC			2047
MAXSPEEDSPINDLE			24000

## 14) Insert in Task Main Functions Page M functions management

Page Init	Master Event	Master Cycle	Page Functions
1			<code>function StartMacro() as char</code>
2			<code>dim VelSpindle as long</code>
3			
4			<code>ISOV1_m_ACK=1</code>
5			<code>select ISOV1_M_cmd</code>
6			<code>case 3</code>
7			<code>ISOV1.out5=true</code> <i>' start spindle in CW</i>
8			<code>ISOV1.out6=false</code> <i>' set CW mode</i>
9			<i>' calculate the Speed</i>
10			<code>VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE</code>
11			<code>ng_dac(3,VelSpindle)</code> <i>' Update the Speed on Dac out</i>
12			<code>ISOV1.out4=true</code> <i>' Start spindle</i>
13			<code>ISOV1_status_m_run=0</code>
14			<code>case 4</code> <i>' start spindle in CCW</i>
15			<code>ISOV1.out5=false</code> <i>' reset CW mode</i>
16			<code>ISOV1.out6=true</code> <i>' set CCW mode</i>
17			<i>' calculate the Speed</i>
18			<code>VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE</code>
19			<code>ng_dac(3,VelSpindle)</code> <i>' Update the Speed on Dac out</i>
20			<code>ISOV1.out4=true</code> <i>' Start spindle</i>
21			<code>ISOV1_status_m_run=0</code>
22			<code>case 5</code> <i>' Stop spindle</i>
23			<code>ISOV1.out4=false</code> <i>' Stop spindle</i>
24			<code>VelSpindle=0</code> <i>' set Speed to 0</i>
25			<code>ng_dac(3,VelSpindle)</code> <i>' Update the Speed on Dac out</i>
26			<code>ISOV1_status_m_run=0</code>
27			<code>case else</code>
28			<code>ISOV1_m_ACK=0</code>
29			<code>endselect</code>
30			<code>endfunction</code>

```
function StartMacro() as char
```

```
dim VelSpindle as long
```

```
ISOV1_m_ACK=1
```

```
select ISOV1_M_cmd
```

```
case 3
```

```
ISOV1.out5=true
```

```
' set CW mode
```

```
ISOV1.out6=false
```

```
' reset CCW mode
```

```
' calculate the Speed
```

```
VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
```

```
ng_dac(3,VelSpindle)
```

```
' Update the Speed on Dac out
```

```
ISOV1.out4=true
```

```
' Start spindle
```

```
ISOV1_status_m_run=0
```

```
case 4
```

```
ISOV1.out5=false
```

```
' reset CW mode
```

```
ISOV1.out6=true
```

```
' set CCW mode
```

```
' calculate the Speed
```

```
VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
```

```
ng_dac(3,VelSpindle)
```

```
' Update the Speed on Dac out
```

```
ISOV1.out4=true
```

```
' Start spindle
```

```
ISOV1_status_m_run=0
```

```
case 5
```

```
ISOV1.out4=false
```

```
' Stop spindle
```

```
VelSpindle=0
```

```
' set Speed to 0
```

```
ng_dac(3,VelSpindle)
```

```
' Update the Speed on Dac out
```

```
ISOV1_status_m_run=0
```

```
case else
```

```
ISOV1_m_ACK=0
```

```
endselect
```

```
endfunction
```



## 15) Insert in Task Main (Master Cycle) Or Task PLC the Call at Functions for I/O Management

Page Init	Master Event	Master Cycle	Page Functions
1	AssiHoming()		' Control homing Switch
2	GetEMCY()		' Get EMERGENCY status
3	AssiManualJog()		' Control Manual JOG
4	SpeedHeWheel()		' Control Speed HandWheel
5	GetOverride()		' Read the Override potentiometer
6	SetOutputs()		' Set Digital outputs

```

AxesHoming()           ' Control homing Switch
GetEMCY()              ' Get EMERGENCY status
AxesManualJog()       ' Control Manuale JOG
SpeedHandWheel()      ' Control Speed HandWheel
GetOverride()         ' Read the Override potentiometer
SetOutputs()          ' Set Digital outputs

```

## 16) Insert the functions in Task Main (Page Functions)

Page Init	Master Event	Master Cycle	Page Functions
1			'*****
2			' Control the switch Axes homing
3			'*****
4			function AxesHoming() as void
5			ISOV1_ext_fcz_x=!ISOV1.inp0 ' Homing switch X
6			ISOV1_ext_fcz_y=!ISOV1.inp1 ' Homing switch Y
7			ISOV1_ext_fcz_z=!ISOV1.inp2 ' Homing switch Z
8			endfunction
9			.
10			.
11			.

```

'*****
' Control the switch Axes homing
'*****
function AxesHoming() as void
    ISOV1_ext_fcz_x=!ISOV1.inp0 ' Homing switch X
    ISOV1_ext_fcz_y=!ISOV1.inp1 ' Homing switch Y
    ISOV1_ext_fcz_z=!ISOV1.inp2 ' Homing switch Z
endfunction
'*****
' Control the General Emergency
'*****
function GetEMCY() as void
    ISOV1_ext_emcy=!ISOV1.inp3 ' General emergency
endfunction
'*****
' Control Manuale JOG
'*****
function AxesManualJog() as void
    if ISOV1.inp4 ' Set Axis X
        ISOV1_asse_man=0
    endif
    if ISOV1.inp5 ' Set Axis Y
        ISOV1_asse_man=1
    endif
    if ISOV1.inp6 ' Set Axis Z
        ISOV1_asse_man=2
    endif
endfunction

```

```

endif
ISOV1_ext_jogp=ISOV1.inp8 'Update the Jog Input +
ISOV1_ext_jogm=ISOV1.inp7 'Update the Jog Input -
endfunction
*****
' Control Manuale JOG
' The update Encoder is in Task PLC
*****
function SpeedHandWheel() as void
    if ISOV1.inp9          'multiplier x1
        ISOV1_msofv=1
    endif
    if ISOV1.inp10       'multiplier x10
        ISOV1_msofv=10
    endif
    if ISOV1.inp11      'multiplier x100
        ISOV1_msofv=100
    endif
endfunction
*****
' Control Override potentiometer
*****
function GetOverride()as void
    ISOV1_vper=ng_adc(0)
endfunction
*****
' Set digital outputs
*****
function SetOutputs()as void
    ISOV1.out0=ISOV1_status_enable_x      ' X enabled
    ISOV1.out1=ISOV1_status_enable_y      ' Y enabled
    ISOV1.out2=ISOV1_status_enable_z      ' Z enabled
    ISOV1.out3=ISOV1_status_error         ' CNC error
endfunction

```

### 17) Insert in theTask PLC the handwheel encoder update

This functions must be sync with task PLC

TASK PLC Code	
Init Task PLC	Task PLC
1	ng_enc(3,EncoderInput()) ' read the HandWheel encoder
2	ISOV1_qvola=EncoderOut 'Update the Handwheel from FiltroVol Object

```

ng_enc(3,EncoderInput())      'read the HandWheel encoder
ISOV1_qvola=EncoderOut        'Update the Handwheel from FiltroVol Object

```

### 14.3 NG35+1xNGIO Axes 3 CanOpen

Link ETHERNET IP: "10.0.0.80" (default)

The follwing project used a axes Canopen type ESTUN

#### Digital Inputs

Digital Input 1 → Switch Home X (N.C.)  
 Digital Input 2 → Switch Home Y (N.C.)  
 Digital Input 3 → Switch Home Z (N.C.)  
 Digital Input 4 → GENERAL EMERGENCY INPUT (N.C.)

Digital Input 5	→ Button JOG X+ (N.O.)
Digital Input 6	→ Button JOG X- (N.O.)
Digital Input 7	→ Button JOG Y+ (N.O.)
Digital Input 8	→ Button JOG Y- (N.O.)
Digital Input 9	→ Button JOG Z+ (N.O.)
Digital Input 10	→ Button JOG Z (N.O.)

### Analog Inputs

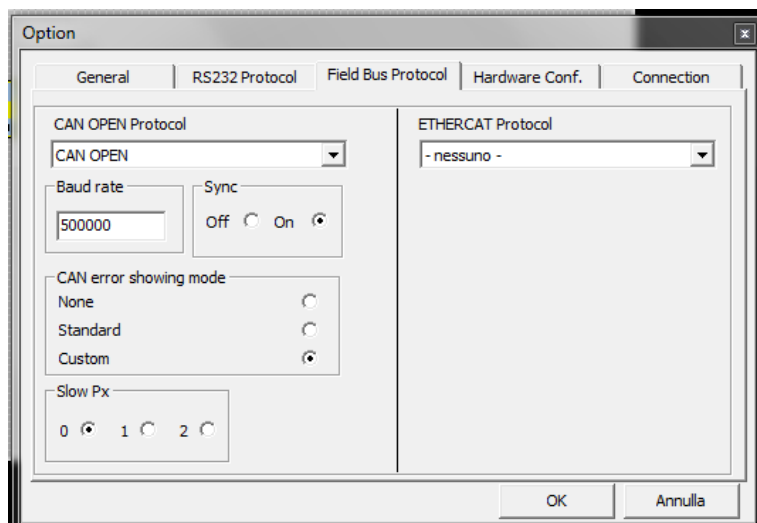
Analog Inputs 1 → Feed Potentiometer Override Axes  
 For enable this Override, you must select the Enable the “Ext OW” from IsoNs Interface”



### Digital Outputs

Digital outputs 1	→ X axis enabled
Digital outputs 2	→ Y axis enabled
Digital outputs 3	→ Z axis enabled
Digital outputs 4	→ CNC Error

- 1) Open new project VTB and select NG35 – Enable the CanOpen Fieldbus – see following

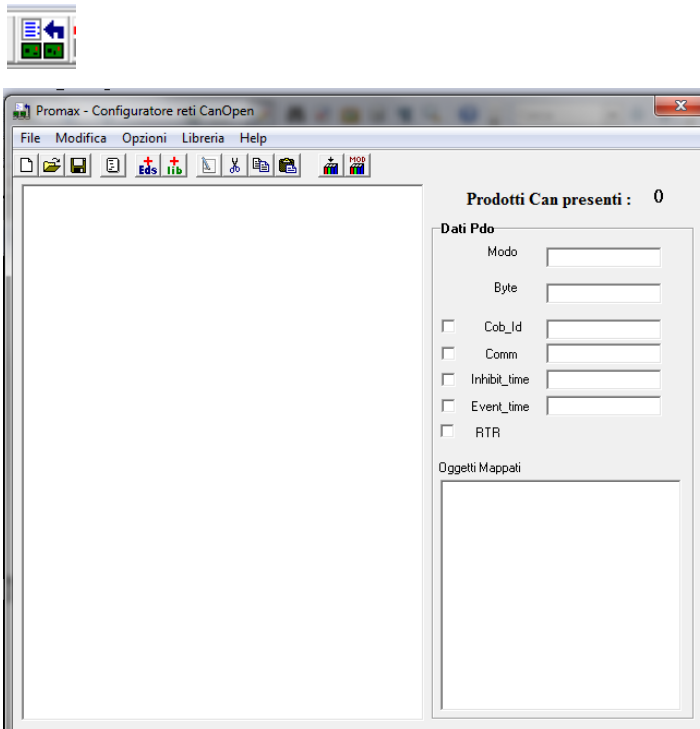


### Select a 2 Ms sample

Start Page:	<input type="text" value="1"/>
Sample:	<input type="text" value="2,0"/> mS
Task Time:	<input type="text" value="5"/> x 2,0 = <input type="text" value="10"/> mS
Screensave:	<input checked="" type="checkbox"/> Enable <input type="text" value="30"/> sec

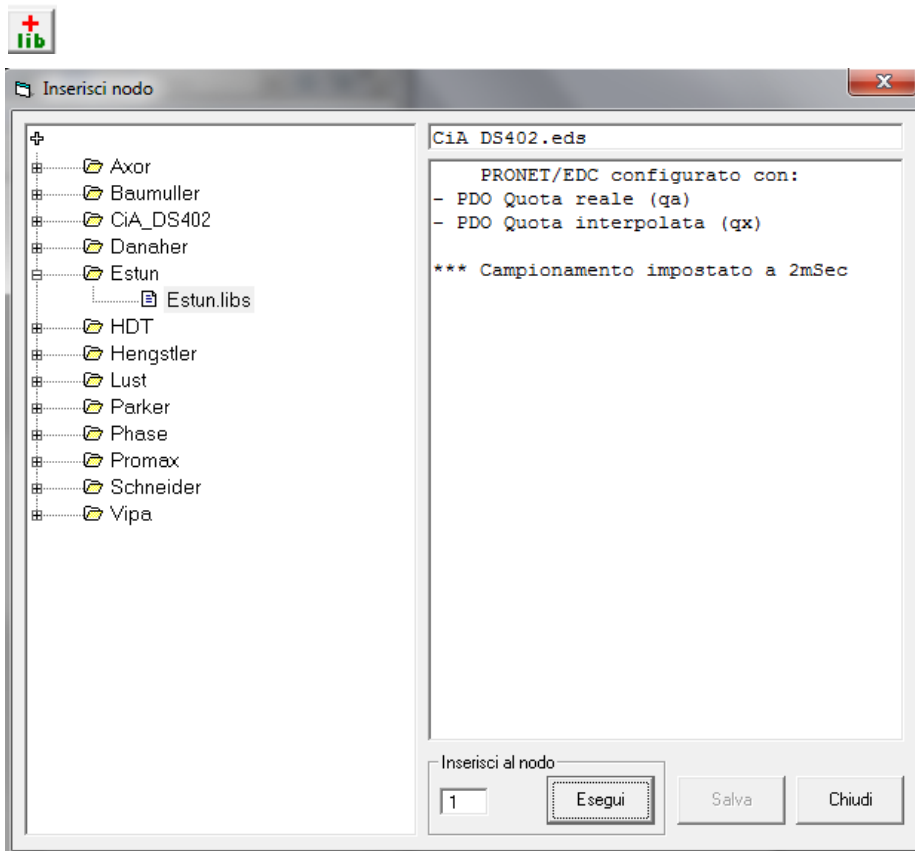
## 2) Prepare the file COP with CanOpen configurator

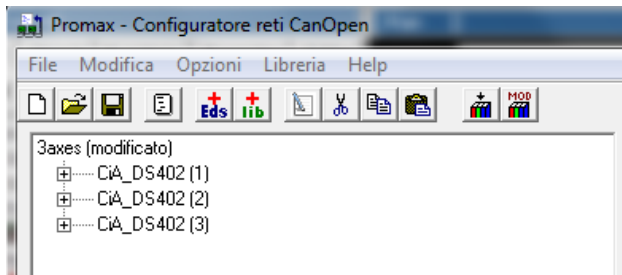
### A) Open the CanOpen configurator



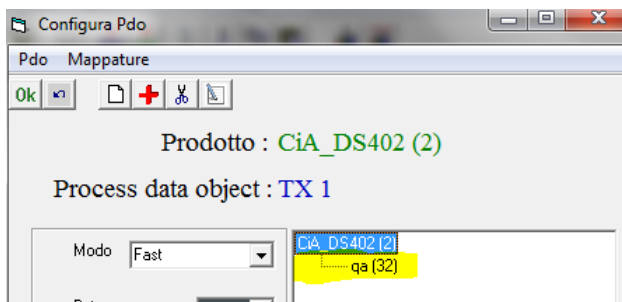
### B) Add from Lib the estun type and insert node 1 – Press button “esegui”

Repeat for Node 2 and Node 3

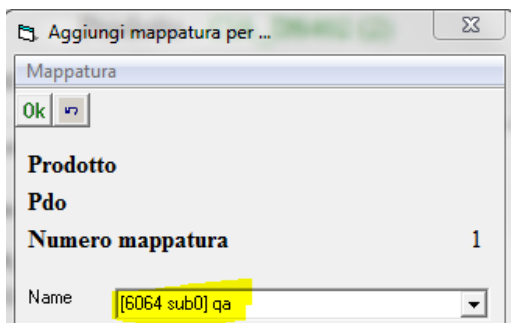




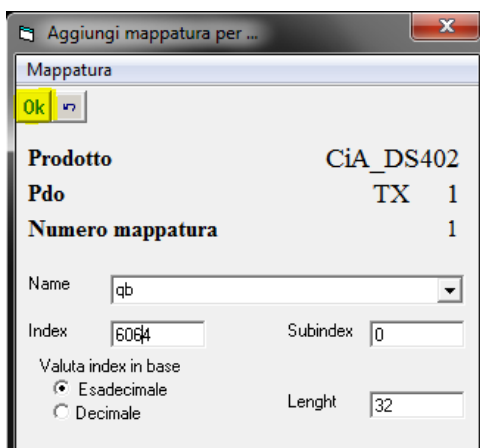
**C) Expand node 2 and double click on pdo\_Tx1 (Fast)**

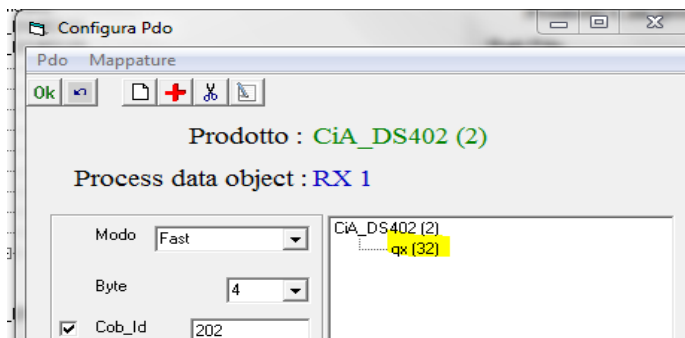
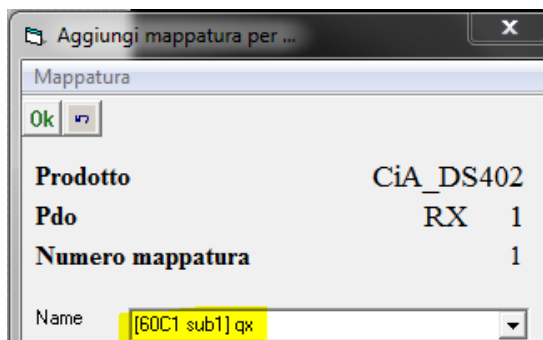


**D) Double click on qa(32)**

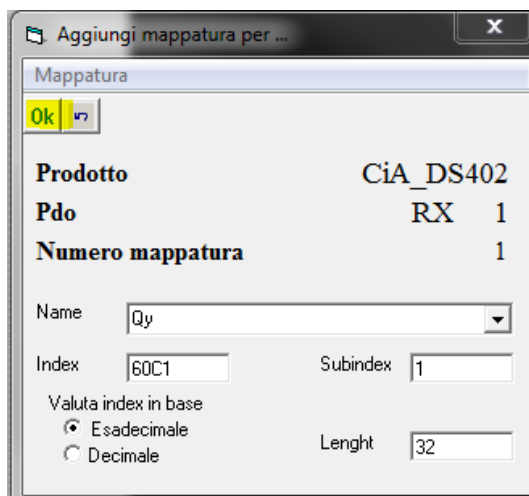
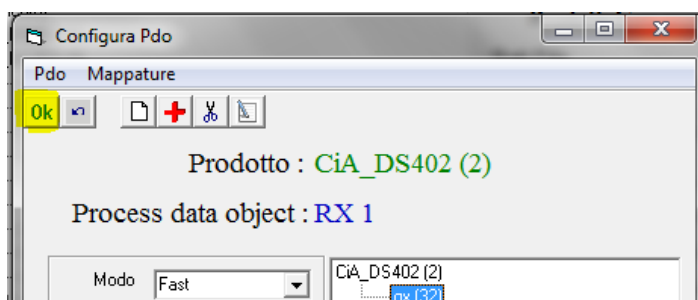


Delete all text Name – (6064 sub0) qa  
And insert only text Qb and press button Ok



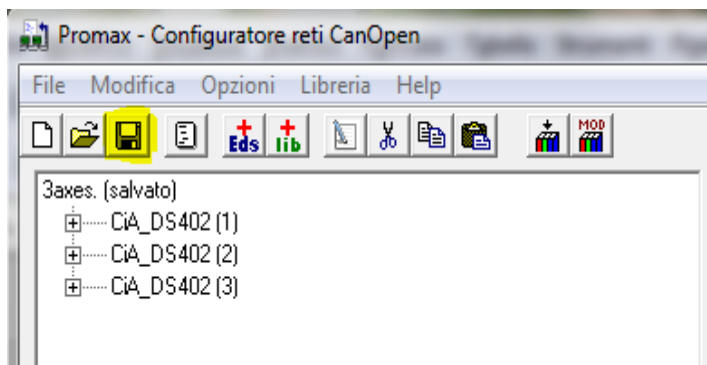
**E) Double click on pdo\_Rx1 (fast)****F) Double click on qx(32)**

Delete all text Name – (60C1 sub1) qx  
And insert only text Qy and press button Ok

**G) Press Ok for enable the modify**

**Repeat point C to G for node 3 Inserting Qc(Qb) and Qz(Qy)**

**H) Press Save for save configuration**



Now the CanOpen Axes configuration is Ready.  
The CanOpen Drives must be set in the following mode:

- X AXIS            **Node 1 Baud 500 Kb**
- Y AXIS            **Node 2 Baud 500 Kb**
- Z AXIS            **Node 3 Baud 500 Kb**

**3) Insert object ISOVIRTUAL and set the default properties**

*Objects → Iso\_Ns → IsoVirtual.vco*

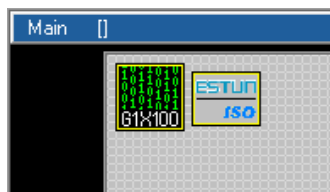


**4) Insert Axis X ISOCanOpen**

*Objects → Iso\_Ns → IsoCanOpen.vco*



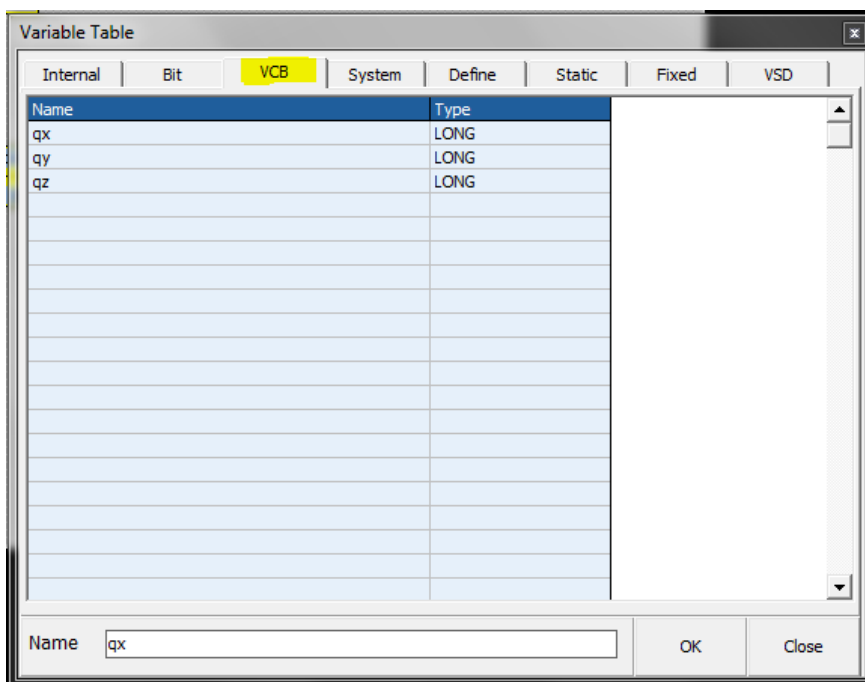
**5) Set the following properties**



AX1	
Property	Events
Property	Value
Nome	AX1
Left	50
Top	10
Nodo	1
Indice asse ISO	0
Nome processo	ISOV1
Nome quota pdo	qx
Mask Allarmi	0xFFFFFFFF
Mask Parametri	0x00

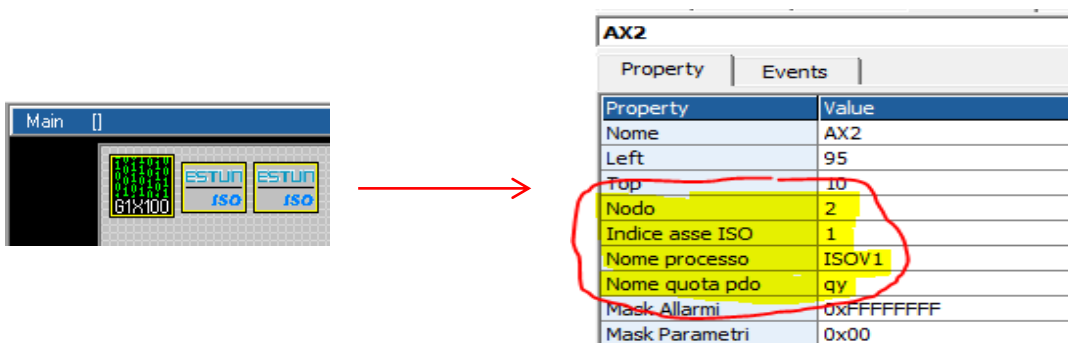
**Note:**

For select the PDO QX, QY and QZ, you must have first created the configuration CanOpen.  
 When you Double Click on "Nome quota pdo", the windows Variable List is open.  
 Select the VCB tab and choose the variable name with double click



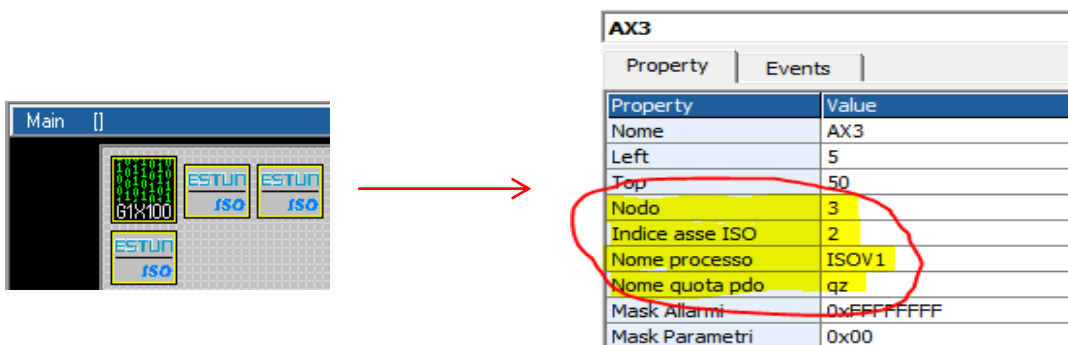
**6) Insert Axis Y ISOCanOpen and set the following properties**

Objects → Iso\_Ns → IsoCanOpen.vco



**7) Insert Axis Z ISOCanOpen and set the following properties**

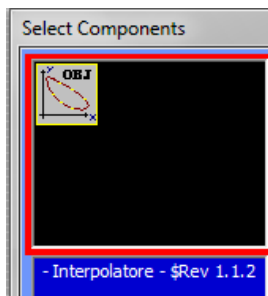
Objects → Iso\_Ns → IsoCanOpen.vco





## 8) Insert the object ObjInterpola

Objects → Motor Control → CobjInterpola.vco



## 9) Set the following properties



Interpola1	
Property	Events
Property	Value
Nome	Interpola1
Left	65
Top	110
N.assi	3
N.tratti	16
Vper	1024
Div. Vper	1024
Abilita arcto	1

## 10) Insert in the Task main (or task plc) the code management

```

TASK PLC Code
Init Task PLC  Task PLC
1      '-----
2      ' Read The Analog Inputs
3      ' For override control
4      '-----
5      ISOV1_vper=ng_adc(0)
6      '-----
7      ' Test Digital Inputs
8      '-----
9      ISOV1_ext_fcz_x=!ISOV1.inp0  ' Homing switch X
10     ISOV1_ext_fcz_y=!ISOV1.inp1  ' Homing switch Y
11     ISOV1_ext_fcz_z=!ISOV1.inp2  ' Homing switch Z
12     ISOV1_ext_emcy=!ISOV1.inp3  ' General EMERGENCY
13     ISOV1_ext_jogp_x=ISOV1.inp4  ' JOG X +
14     ISOV1_ext_jogm_x=ISOV1.inp5  ' JOG X -
15     ISOV1_ext_jogp_y=ISOV1.inp6  ' JOG Y +
16     ISOV1_ext_jogm_y=ISOV1.inp7  ' JOG Y -
17     ISOV1_ext_jogp_z=ISOV1.inp8  ' JOG Z +
18     ISOV1_ext_jogm_z=ISOV1.inp9  ' JOG Z -
19     '-----
20     ' Test Digital Outputs
21     '-----
22     ISOV1.out0=ISOV1_status_enable_x  ' X enabled
23     ISOV1.out1=ISOV1_status_enable_y  ' Y enabled
24     ISOV1.out2=ISOV1_status_enable_z  ' Z enabled
25     ISOV1.out2=ISOV1_status_error     ' CNC error

```

```

'-----
' Read The Analog Inputs
' For override control
'-----
ISOV1_vper=ng_adc(0)
'-----
' Test Digital Inputs
'-----
ISOV1_ext_fcz_x=!ISOV1.inp0  ' Homing switch X
ISOV1_ext_fcz_y=!ISOV1.inp1  ' Homing switch Y
ISOV1_ext_fcz_z=!ISOV1.inp2  ' Homing switch Z
ISOV1_ext_emcy=!ISOV1.inp3  ' General EMERGENCY
ISOV1_ext_jogp_x=ISOV1.inp4  ' JOG X +
ISOV1_ext_jogm_x=ISOV1.inp5  ' JOG X -
ISOV1_ext_jogp_y=ISOV1.inp6  ' JOG Y +
ISOV1_ext_jogm_y=ISOV1.inp7  ' JOG Y -
ISOV1_ext_jogp_z=ISOV1.inp8  ' JOG Z +
ISOV1_ext_jogm_z=ISOV1.inp9  ' JOG Z -
'-----
' Test Digital Outputs
'-----
ISOV1.out0=ISOV1_status_enable_x  ' X enabled
ISOV1.out1=ISOV1_status_enable_y  ' Y enabled
ISOV1.out2=ISOV1_status_enable_z  ' Z enabled
ISOV1.out2=ISOV1_status_error     ' CNC error

```

## 14.4 NGM EVO+NGQx (CanOpen) Axes 3 Step/Dir, Spindle and handwheel

Link RS32 on COM1 NGM EVO

The following project use a handwheel connect to Ch 1 encoder inputs NGQx , selector for JOG AXES and spindle in Analog output 1 in NGQx. The selector Axes JOG and Handwheel speed are connect to NGQx digital inputs in CanOpen To enable a selector is necessary insert in the init TASK PLC the following code:

ISOV1\_soft\_sel\_man=0 ' Enable the internal VTB selector

### Digital Inputs

Digital Input 1 (NGM EVO ISOV1.inp0)	→ Switch Home X (N.C.)
Digital Input 2 (NGM EVO ISOV1.inp1)	→ Switch Home Y (N.C.)
Digital Input 3 (NGM EVO ISOV1.inp2)	→ Switch Home Z (N.C.)
Digital Input 4 (NGM EVO ISOV1.inp3)	→ GENERAL EMERGENCY INPUT (N.C.)
.	
.	
Digital Input 1 (NGQx ISOV1.inp16)	→ Selector JOG X (N.O.)
Digital Input 2 (NGQx ISOV1.inp17)	→ Selector JOG Y (N.O.)
Digital Input 3 (NGQx ISOV1.inp18)	→ Selector JOG Z (N.O.)
Digital Input 4 (NGQx ISOV1.inp19)	→ Button JOG - (N.O.)
Digital Input 5 (NGQx ISOV1.inp20)	→ Button JOG + (N.O.)
Digital Input 6 (NGQx ISOV1.inp21)	→ Handwheel Speed x1
Digital Input 7 (NGQx ISOV1.inp22)	→ Handwheel Speed x10
Digital Input 8 (NGQx ISOV1.inp23)	→ Handwheel Speed x100

### Analog Inputs

Analog Inputs 1 (NGQx) → Feed Potentiometer Override Axes  
For enable this Override, you must select the Enable the “Ext OW” from IsoNs Interface”



### Digital Outputs

Digital outputs 1 (NGM EVO ISOV1.out0)	→ X axis enabled
Digital outputs 2 (NGM EVO ISOV1.out1)	→ Y axis enabled
Digital outputs 3 (NGM EVO ISOV1.out2)	→ Z axis enabled
Digital outputs 4 (NGM EVO ISOV1.out3)	→ CNC Error
.	
.	
Digital outputs 1 (NGQx ISOV1.out16)	→ Spindle start/stop
Digital outputs 2 (NGQx ISOV1.out17)	→ Spindle CW (M3)
Digital outputs 3 (NGQx ISOV1.out18)	→ Spindle CCW (M4)

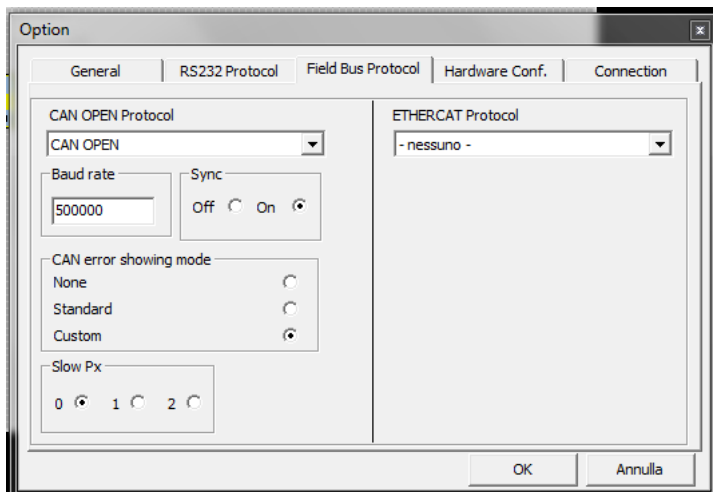
### Encoder Inputs

Encoder Ch 1 (NGQx) → HandWheel encoder

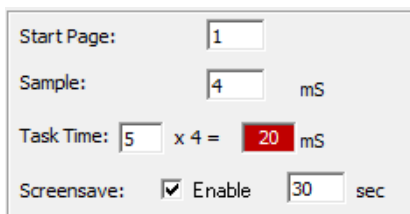
### Analog Outputs

Analog out 1 (NGQx) → SPEED Spindle

1) Open new project VTB and select NGM EVO – Enable the CanOpen Fieldbus – see following

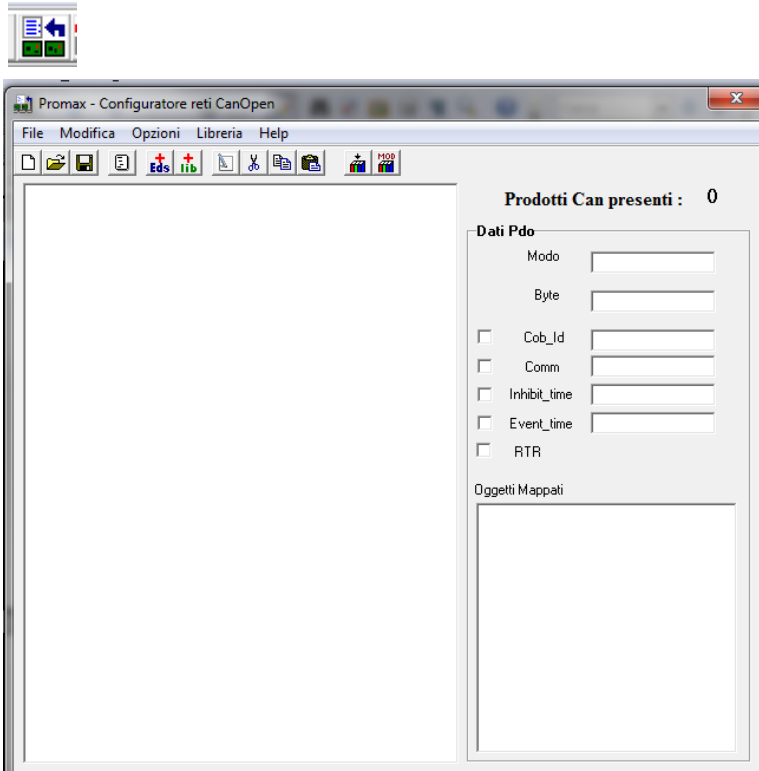


Select a 4 Ms sample

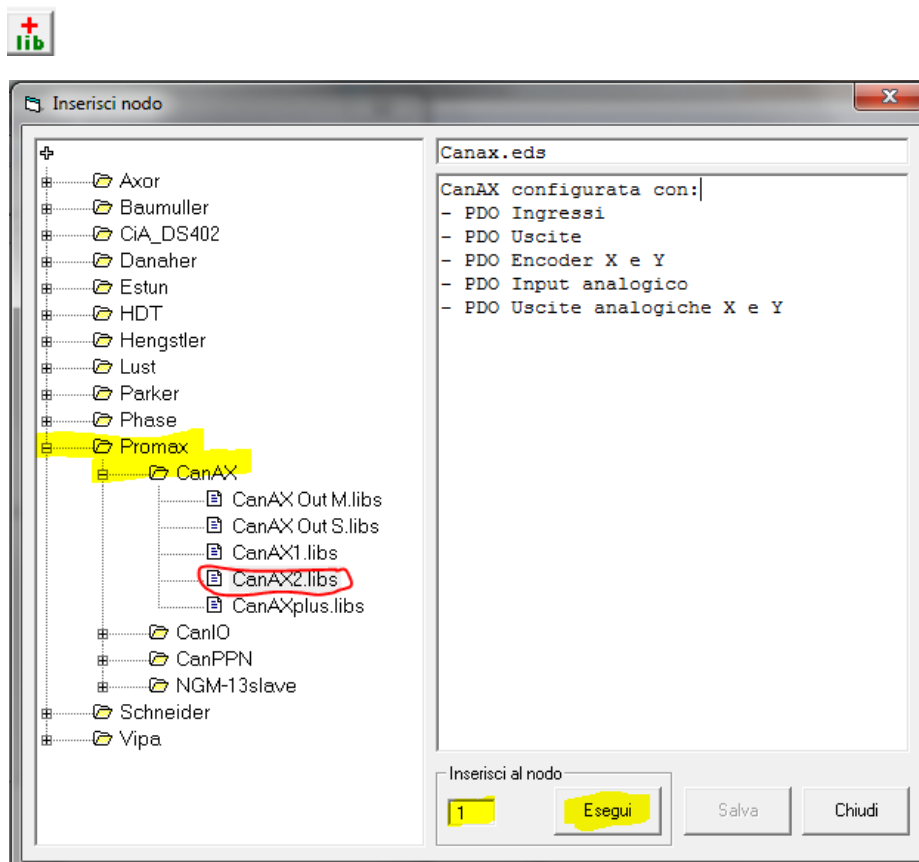


2) Prepare the file COP with CanOpen configurator

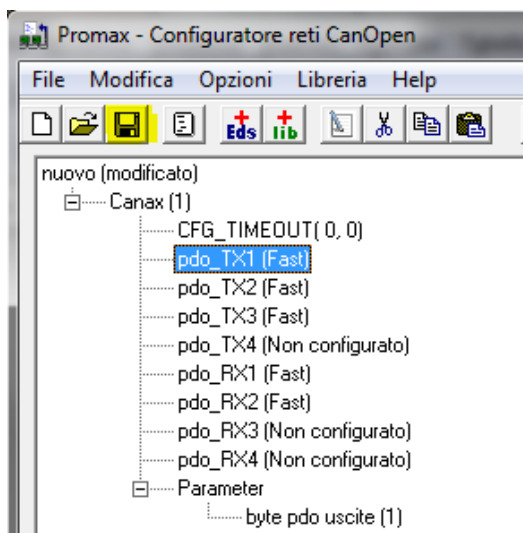
A) Open the CanOpen configurator



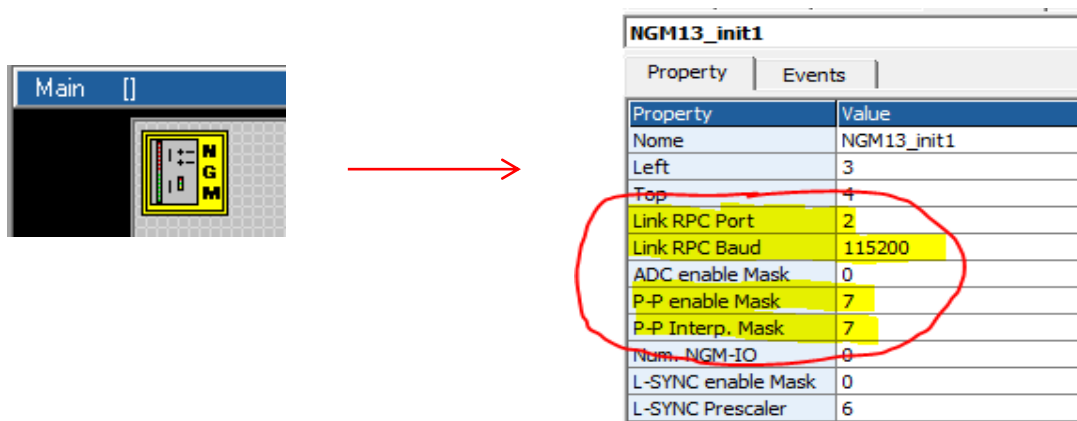
**B) Add from Lib the Promax type Canax → CanAX2.lib and insert node 1 – Press button “esegui”**



**C) Press button Save**



3) Set link on COM2 NGM EVO and PP Interp mask on 7 (X Y Z channel enabled)



NGM13_init1	
Property	Events
Property	Value
Nome	NGM13_init1
Left	3
Top	4
Link RPC Port	2
Link RPC Baud	115200
ADC enable Mask	0
P-P enable Mask	7
P-P Interp. Mask	7
Num. NGM-IO	0
L-SYNC enable Mask	0
L-SYNC Prescaler	6

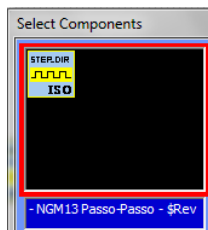
4) Insert object ISOVIRTUAL and set the default properties

Objects → Iso\_Ns → IsoVirtual.vco


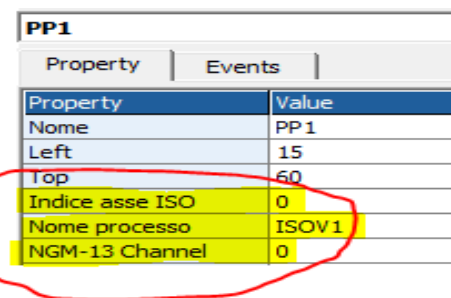


5) Insert Axis X ISOPP

Objects → Iso\_Ns → IsoPP.vco



6) Set the following properties

PP1	
Property	Events
Property	Value
Nome	PP1
Left	15
Top	60
Indice asse ISO	0
Nome processo	ISOV1
NGM-13 Channel	0

7) Insert Axis Y ISOPP and set the following properties

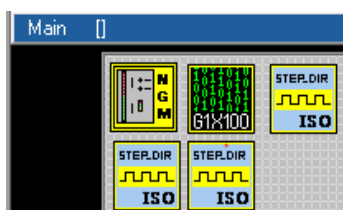
Objects → Iso\_Ns → IsoPP.vco



PP2	
Property	Events
Property	Value
Nome	PP2
Left	65
Top	60
Indice asse ISO	1
Nome processo	ISOV1
NGM-13 Channel	1

8) Insert Axis Z ISOPP and set the following properties

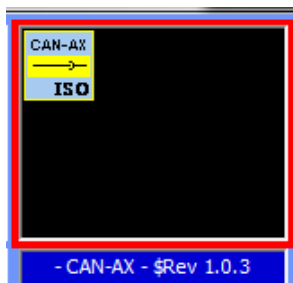
Objects → Iso\_Ns → IsoPP.vco



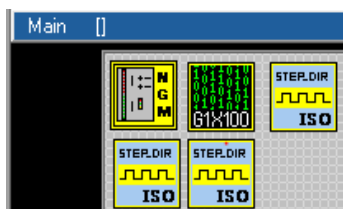
PP3	
Property	Events
Property	Value
Nome	PP3
Left	115
Top	80
Indice asse ISO	2
Nome processo	ISOV1
NGM-13 Channel	2

9) Insert the ISO-IO Can-Ax (is the same of NGQx)

Objects → Iso\_Ns → ISO-IO.vco



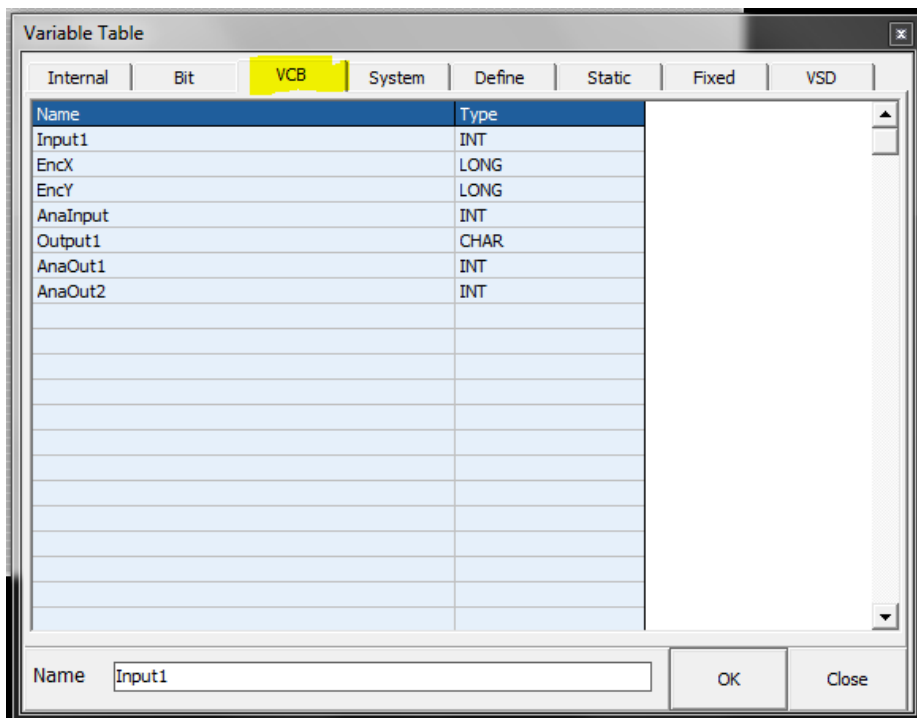
10) Set the following properties



digio1	
Property	Events
Property	Value
Nome	digio1
Nodo	0
Allarme cfg	41
Nome processo	ISOV1
Indice ISO-IO (16 bit)	1
Hardware enable	True
Variabile inp	Input1
Variabile out	Output1

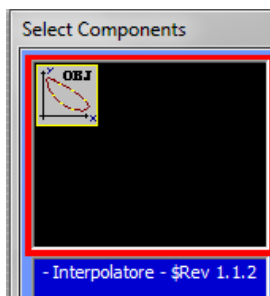
**Note:**

For select the *Variabile Inp* and *Variabile out*, you must have first created the configuration *CanOpen*.  
 When you Double Click on “*Variabile Inp*” or “*Variabile out*”, the windows *Variable List* is open.  
 Select the *VCB* tab and choose the variable name with double click

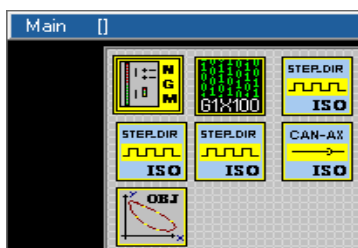


**11) Insert the object ObjInterpola**

*Objects* → *Motor Control* → *CobjInterpola.vco*



**12) Set the following properties**



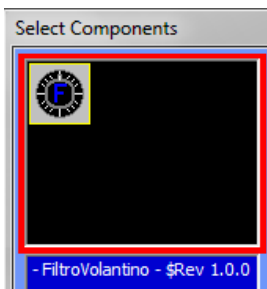
Interpola1	
Property	Events
Property	Value
Nome	Interpola 1
Left	65
Top	110
N.assi	3
N.tratti	16
Vper	1024
Div. Vper	1024
Abilita arcto	1



13) Declare a following Global Variables

Internal VAR	Bit VAR	Define	Static VAR	VSD VAR	Fixed VA
			No	EXP	<input type="checkbox"/>
Variable	Type	Shared	Export in Class		
EncoderInput	LONG	No			
EncoderOut	LONG	No			

14) Insert the FiltroVol Object for handwheel  
Objects → Motor Control → CfiltroVol.vco



15) Set the following properties

FiltroVol1	
Property	Events
Property	Value
Nome	FiltroVol1
Left	100
Top	55
Numero elementi	10
Molt. filtro	100
Encoder	EncoderInput
Variabile	EncoderOut

16) Insert in "Init task PLC " the entry point for M Functions and the enable external selector

```

TASK PLC Code
Init Task PLC: Task PLC
1 ISOV1_soft_sel_man=0 ' Enable the internal VTB selector
2 ISOV1_Start_m=StartMacro ' entry point MACRO management
    
```

17) Insert 2 Define Constant in Global Variables → Define

The MAXSPEEDSPINDLE depend to MAX rpm at 10 Volt your spindle

Internal VAR	Bit VAR	Define	Static VAR
Variable	Type		
MAXDAC		2047	
MAXSPEEDSPINDLE		24000	

## 18) Insert in Task Main Functions Page M functions management

Page Init	Master Event	Master Cycle	Paste Functions
-----------	--------------	--------------	-----------------

```

1 function StartMacro() as char
2 dim VelSpindle as long
3
4     ISOV1_m_ACK=1
5     select ISOV1_M_cmd
6         case 3             ' start spindle in CW
7             ISOV1.out17=true     ' set CW mode
8             ISOV1.out18=false    ' reset CCW mode
9             ' calculate the Speed
10            VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
11            AnaOut1=VelSpindle   ' Update the Speed on PDO
12            ISOV1.out16=true     ' Start spindle
13            ISOV1_status_m_run=0
14        case 4             ' start spindle in CCW
15            ISOV1.out17=false    ' reset CW mode
16            ISOV1.out18=true     ' set CCW mode
17            ' calculate the Speed
18            VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
19            AnaOut1=VelSpindle   ' Update the Speed on PDO
20            ISOV1.out16=true     ' Start spindle
21            ISOV1_status_m_run=0
22        case 5             ' Stop spindle
23            ISOV1.out16=false    ' Stop spindle
24            VelSpindle=0         ' set Speed to 0
25            AnaOut1=VelSpindle   ' Update the Speed on PDO
26            ISOV1_status_m_run=0
27        case else
28            ISOV1_m_ACK=0
29    endselect
30 endfunction
31

```

```

function StartMacro() as char
dim VelSpindle as long

```

```

ISOV1_m_ACK=1
select ISOV1_M_cmd
    case 3             ' start spindle in CW
        ISOV1.out17=true     ' set CW mode
        ISOV1.out18=false    ' reset CCW mode
        ' calculate the Speed
        VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
        AnaOut1=VelSpindle   ' Update the Speed on PDO
        ISOV1.out16=true     ' Start spindle
        ISOV1_status_m_run=0
    case 4             ' start spindle in CCW
        ISOV1.out17=false    ' reset CW mode
        ISOV1.out18=true     ' set CCW mode
        ' calculate the Speed
        VelSpindle=(ISOV1_generic(9)*MAXDAC)/MAXSPEEDSPINDLE
        AnaOut1=VelSpindle   ' Update the Speed on PDO
        ISOV1.out16=true     ' Start spindle
        ISOV1_status_m_run=0
    case 5             ' Stop spindle
        ISOV1.out16=false    ' Stop spindle
        VelSpindle=0         ' set Speed to 0
        AnaOut1=VelSpindle   ' Update the Speed on PDO
        ISOV1_status_m_run=0
    case else
        ISOV1_m_ACK=0
endselect
endfunction

```

## 19) Insert in Task Main (Master Cycle) Or Task PLC the Call at Functions for I/O Management

	Page Init	Master Event	Master Cycle	Page Functions
1				AxesHoming () ' Contro homing Switch
2				GetEMCY () ' Get emergency status
3				AxesManualJog () ' Control Manuale JOG
4				SpeedHandWheel () ' Control Speed handwheel
5				GetOverride () ' Read the Override potentiometer
6				SetOutputs () ' Set Digital outputs

```

AxesHoming()           ' Contro homing Switch
GetEMCY()              ' Get emergency status
AxesManualJog()       ' Control Manuale JOG
SpeedHandWheel()      ' Control Speed handwheel
GetOverride()         ' Read the Override potentiometer
SetOutputs()          ' Set Digital outputs

```

## 20) Insert the functions in Task Main (Page Functions)

	Page Init	Master Event	Master Cycle	Page Functions
1				'*****
2				' Control the switch Axes homing
3				'*****
4				function AxesHoming() as void
5				ISOV1_ext_fcx_x=!ISOV1.inp0 ' Homing switch X
6				ISOV1_ext_fcx_y=!ISOV1.inp1 ' Homing switch Y
7				ISOV1_ext_fcx_z=!ISOV1.inp2 ' Homing switch Z
8				endfunction
9				.
10				.
11				.

```

'*****
' Control the switch Axes homing
'*****
function AxesHoming() as void
    ISOV1_ext_fcx_x=!ISOV1.inp0 ' Homing switch X
    ISOV1_ext_fcx_y=!ISOV1.inp1 ' Homing switch Y
    ISOV1_ext_fcx_z=!ISOV1.inp2 ' Homing switch Z
endfunction
'*****
' Control the General Emergency
'*****
function GetEMCY() as void
    ISOV1_ext_emcy=!ISOV1.inp3 ' General emergency
endfunction
'*****
' Control Manuale JOG
'*****
function AxesManualJog() as void
    if ISOV1.inp16 ' Set Axis X
        ISOV1_asse_man=0
    endif
    if ISOV1.inp17 ' Set Axis Y
        ISOV1_asse_man=1
    endif
    if ISOV1.inp18 ' Set Axis Z
        ISOV1_asse_man=2
    endif
endfunction

```

```

    endif
    ISOV1_ext_jogp=ISOV1.inp20 'Update the Jog Input +
    ISOV1_ext_jogm=ISOV1.inp19 'Update the Jog Input -
endfunction
*****
' Control Manuale JOG
' The update Encoder is in Task PLC
*****
function SpeedHandWheel() as void
    if ISOV1.inp21          'multiplier x1
        ISOV1_msofv=1
    endif
    if ISOV1.inp22          'multiplier x10
        ISOV1_msofv=10
    endif
    if ISOV1.inp23          'multiplier x100
        ISOV1_msofv=100
    endif
endfunction
*****
' Control Override potentiometer
' from PDO declare in configurator CanOpen
*****
function GetOverride()as void
    ISOV1_vper=AnallInput
endfunction
*****
' Set digital outputs
*****
function SetOutputs()as void
    ISOV1.out0=ISOV1_status_enable_x      ' X enabled
    ISOV1.out1=ISOV1_status_enable_y      ' Y enabled
    ISOV1.out2=ISOV1_status_enable_z      ' Z enabled
    ISOV1.out3=ISOV1_status_error        ' CNC error
endfunction

```

## 21) Insert in theTask PLC the handwheel encoder update

This functions must be sync with task PLC

TASK PLC Code	
Init Task PLC	Task PLC
1	EncoderInput=EncX          'read the HandWheel encoder from NGQx PDO EncX
2	ISOV1_qvola=EncoderOut      'Update the Handwheel from FiltroVol Object
3	

```

EncoderInput=EncX          'read the HandWheel encoder from NGQx PDO EncX
ISOV1_qvola=EncoderOut      'Update the Handwheel from FiltroVol Object

```

## Index

1	PREFACE .....	3
2	BASE COMPONENTS .....	3
2.1	Select Hardware .....	3
2.2	Insert Object <i>cobjinterpola</i> .....	3
2.3	Properties setting of <i>objinterpola</i> .....	4
2.4	Insert Object Iso Virtual .....	4
2.5	Properties setting of IsoVirtual .....	4
2.6	Insert a long fixed variable name <i>fixed0</i> .....	5
3	Insert the Axes type .....	5
4	Insert the I/O .....	10
5	USE the M functions to internal the CNC .....	11
5.1	Read the M parameters .....	12
5.2	Write the M parameters.....	12
5.3	M flags .....	13
5.4	Example M3 M4 M5 start/stop Spindle.....	14
6	Standard I/O.....	15
7	Status Word.....	22
8	PLC I/O management .....	23
8.1	Defined bit digital Inputs .....	23
8.2	Defined bit digital Outputs.....	23
9	FEED External Override.....	23
10	Electronic HandWheel .....	24
11	Machine Parameters.....	27
11.1	General Parameters .....	27
11.2	Axes X Parameters.....	28
11.3	Custom Parameters .....	29
12	Alarms bit mapped.....	30
13	CanOpen Alarms .....	31
14	Examples.....	32
14.1	NGQ-NGM EVO 3 Axes Step/Dir.....	32
14.2	NG35+2xNGIO Axes 3 Analog +/- 10V and handwheel .....	36
14.3	NG35+1xNGIO Axes 3 CanOpen .....	42
14.4	NGM EVO+NGQx (CanOpen) Axes 3 Step/Dir, Spindle and handwheel.....	51