VTB Visual Tool Basic

www.promax.it

Objects References

PROMAX Motion & Control

Rev. 3.00.0

1 PREFACE

This manual is a guide to the objects of VTB,
The objects represent an important feature of the programming language and simplify application development. There are technological objects and functions of various types suitable for resolving specific situations.

1.1 CbrowserMC.vco – Browser for management FLASH DISK

Browser class that contains the operating FLASH DISK. These allow you to save data
this type of memory managing their use as numerical programs. The data are saved only type
Long and for a maximum of **1023 Long** for the program. These data must be stated in a vector
CKeyPopupPadMC objects in the class.

Hardware ***PeC70 – NGM EVO+.... - NG35+.....***

Property

Nome	Name object. Not managed in run time
Col.testo Nomi	Text names colori. Not managed in run time
SfondoNomi	Back color names. Not managed in run time
ColScrollbar	Scroll Bar color. Not managed in run time
Colore Griglia	Grid color. Not managed in run time
ColbarraPuls.	Keys bar color. Not managed in run time
Col.Num.Pag.	Number page color. Not managed in run time
Card Ok	Message FLASH DISK OK Not managed in run time
Card Er	Message FLASH DISK error. Not managed in run time
Pagina	Page number show in the browser. Not managed in run time
Larghezza Tasti	Key “ESC” and “FORMAT”width. Not managed in run time
Esc	Message show on exit from browser. Not managed in run time
Format	Message show on format FLASH DISK. Not managed in run time

Methods

No

Event

OnEnter	On press CR to numeric PAD
OnCancel	On press ESC to numeric PAD
OnSelect	Occurs when you press on a row of browser and the field "Load Save" the KeyPopupPadMC is False. In the vector array_cartella returns the name under which you saved the program in FLASH DISK

WARNING!
The use of this object makes necessary the presence of a page in the alphanumeric pad AlfaPadC60 type, a type of InputKey KeyPopupPadMC and a MsgBox the type PopMsgBoxMC.

1.2 CstdAllarm.vco – Alarms Browser

 Class that contains objects that can display ALARMS. Alarms are managed by a vector of type INT contains more or less every bit of these registers and enables a corresponding alarm.

Vett(0) ***Bit 1*** ***Alarm 1***

Vett(0) ***Bit 2*** ***Alarm 2***

.
.

Vett(1) ***Bit 1*** ***Alarm 17***

Hardware ***PeC70 – NGM EVO+.... - NG35+.....***

The alarm name is taken from a table of TEXT. Names must be located in proper sequence for correct display (first name ALARM 1 etc..). In the case of

2 CLASS INPUTBIT

The class contains objects related to InputBit 'use of bit variables, is associated with normal variables, is associated with digital inputs. Consequently it is possible to manage the state of bits in a simple and immediate detecting rising edges and falling using events already prepared.

2.1 CstdBit.vco – BIT management

Class that contains objects that have no graphical display. Generate only two events and **StatoOn StatoOff** where you can insert the control code.

Hardware ***All***

Property

Nome Name object. Not managed in run time
Variabile Bit variable name. Not managed in run time
Enable NameObject.Enable = True enable events
NameObject.Enable = False disable events

Methods

No

Event

StatoOn Occurs when the bit is to logical state 1 (rising edge)
StatoOff Occurs when the bit is to logical state 0 (falling edge)

3 CLASS MOTOR CONTROL

The class includes MotorControl objects that deal with the motion control axes. In general motion control axes provides several specific types (electrical shafts, CAM, positioners, etc..) That are entirely in this macro-class collections. VTB controlling external devices that respond to the standard profile DS301 and DSP402 CAN OPEN, STEP / DIR, Analog + /-10V or EtherCAT, so in principle, all components that use these protocols, can be fully managed. Significant facilitation derives from the fact of being able to use objects already prepared and tested, this does not preclude, however, to be able to use a device outside of the library objects provided.

3.1 CbitCam.vco – Management Bit CAM

Class refers to CAMS that are generated by activating a digital output to a certain value of a unit MASTER and disabling it to another value. This is useful for the management of PISTONS, synchronizing the movement with the MASTER.
The MASTER can be any numerical size available (External Encoder, Axis external virtual axis, etc..)
The only limitation is that the magnitude must necessarily be a multiple MASTER TRACK:
512,1024,2048,4096 etc..

Hardware ***All***

Property

Impulsi/giro	Pulses per revolution of the master source. Not managed in run time **WARNING this value must necessarily be BINARY (256-512-1024 etc)**
MasterSet	Value of the master pulses of variable relative to the SET BIT
MastertRes	Value of the master pulses of variable relative to the RESET BIT
Master	Source variable that defines the MASTER . Not managed in run time
VarBit	Variable destination which can be a bit that is generally associated to a digital output. Not managed in run time

Methods

Enable	ObjectName.Enable=True Enable the BIT control ObjectName.Enable=False Disable the BIT control

Event

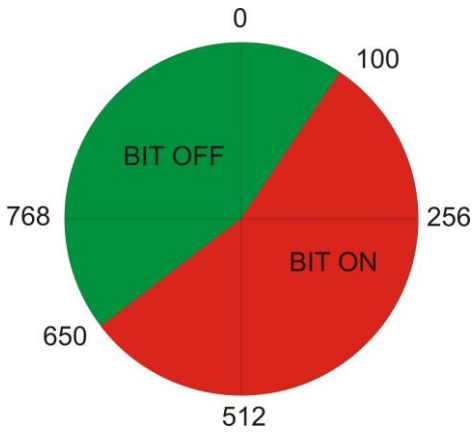
No

Example:

Value setted

Impulsi/giro=1024	Pulses per revolution master
MasterSet=100	Set Bit at 100 pulses
MasterRes=650	Reset Bit at 650 pulses

Result



3.2 Ccam.vco eCam /Continue eCam -Management eCAM for MOTOR CONTROL

Class refers to generic ELECTRONIC TIMING created with external tools (CAD, etc..). types of CAM may be continuous (Cam Continue) or return to zero (Cam). This class essentially works on a carrier must have a size equal to the pulse of a MASTER. Latter 'can be any internal or external source to the platform hardare (ENCODER COUNTER, VARIABLE etc..) For each position of the vector corresponds to the relative share that the board must take SLAVE. In practice, the encoder is taken as the MASTER INDEX OF VECTOR which therefore corresponds to its share of the slave. The position of the SLAVE is placed in a variable that can be associated with a PDO, PID filter, share STEP.

Continue cam defines an axis SLAVE that goes in one direction.
Cam-return to zero defines an axis running from a share SLAVE (ZERO) and returns to this

Hardware ***All***

Property

Nome	Name object. Not managed in run time
N. Punti	Number of eCAM points it is the same of MASTER pulses .Not managed in run time
Master	Variabile Sorgente che definisce il MASTER. Non gestibile in run time
Slave	Allocation of variable contents of the vector must necessarily correspond to the PDO if axis is in CANopen. Not managed in run time
VetCam	Vector that contains the points of CAM . Not managed in run time
Rotazione	Reverses the motor rotation. Value admitted 0 o 1

Methods

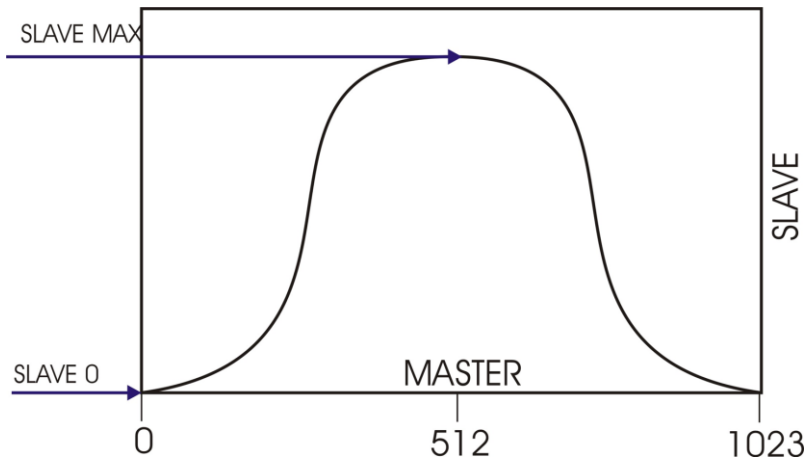
Enable	ObjectName.Enable=True Enable the slave motor control (eCam) ObjectName.Enable=False Disable the slave motor control
Fase	ObjectName.Fase=Value Indicates the starting phase of CAM. In practice defines the departure from the position indicated in the carrier. Phase

Event

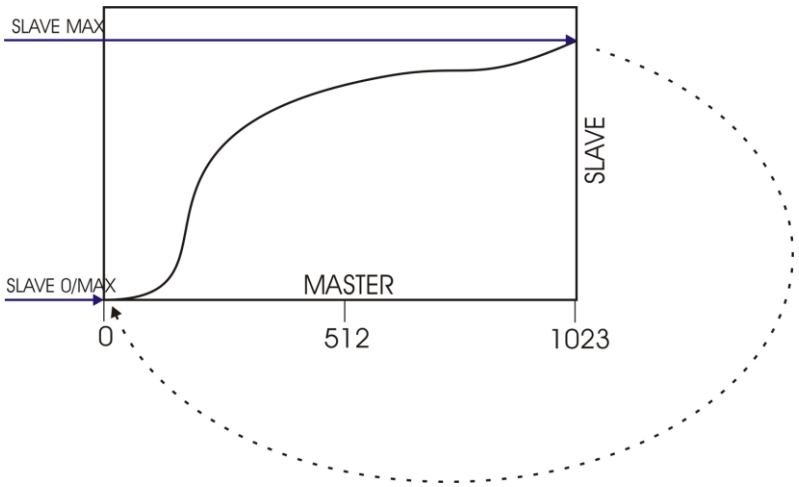
No

For clarity is drawn an example of a CAM which simulates a master from 0 to 1024 impulsii, continuous, ie without returning to zero, and with return to zero

CAM RETURN TO ZERO



CONTINUE CAM



Example
This example represents the code for a generic CAM WITH RETURN TO ZERO Camma1

Golbal Variables

MasterEnc	Long	Master Source
SlaveMot	Long	Slave destination
VetCam(20)	Long	CAM Vector

Camma1 Property
Numero Punti=20
Master=MasterEnc
Slave=SlaveMot
VetCam=VetCam

```
' VetCam init points
vcam1(0)=0
vcam1(1)=1
vcam1(2)=2
vcam1(3)=3
vcam1(4)=4
vcam1(5)=5
vcam1(6)=6
vcam1(7)=7
vcam1(8)=8
vcam1(9)=9
vcam1(10)=10
vcam1(11)=9
vcam1(12)=8
vcam1(13)=7
vcam1(14)=6
vcam1(15)=5
vcam1(16)=4
vcam1(17)=3
vcam1(18)=2
vcam1(19)=1
'Enable the eCam control
Camma1.Enable=True
' Task PLC cycle at 2 Ms
Inc MasterEnc
```

The variable SlaveMot follows the profile of VETCAM. Initially, the PHASE parameter is zero, this means that the position of the ZERO MASTERENC corresponds to the position of VETCAM (0). Changing the value of PHASE is possible to change the start point of reference in the CAM MASTER ZERO. The example is purely indicative, since both the values is the number of points are not sufficient to generate a continuous profile of the slave axis.

3.3 CcamPulse.vco – Management impulsive BIT CAM

Class refers to CAMS that are generated by activating a digital output pulse for a set duration. At a certain value of MASTER greater than or equal to STEP, the bit is activated for the time VarBit DURATION. The variable MASTER to be managed externally, that is, its value must be reset (to zero) from outside to resume management of the bit. In addition to having an effective activation should also be resetting the DONE bit.
In fact, the MASTER is variable in value running from a value of 0 and in any case returns to this value.
In the case of rotary axes must be reported a virtual axis from 0 to N.

Hardware ***All***

Property

Fase	Value of the master pulses of variable relative to the SET BIT.
Durata	Time to SET in Millisecond.
Master	Variable MASTER source. Not managed in run time
VarBit	Destination variable BIT (ex digital output). Not managed in run time
Fatto	This is setted by object when the bit is setted. Must be reset by code for reload

Methods

Enable	NameObject.Enable=True Enable the BIT control NameObject.Enable=False Disable the BIT control

Event

No

Example to continue rotative axis

Global variables

MasterEnc	Long	Master source (Axis rotative encoder or PDO)
VirtualEnc	Long	Internal virtual encoder.
ImpulsiMaster	Long	Pulses per revolution MASTER
PulsBit VarBit		BIT variable

Property object CammaBitP1

```
Fase=300
Durata=20
Master=VirtualEnc
VarBit=PulseBit
```

'Init TASK PLC

```
ImpulsiMaster=1024      'Set pulses master
```

Function for calculated module (insert in main functions)

```
'Calculated module
function  Module(V as long, M as long) as long
        Dim Ris as long
        Dim P as float
        Dim A as Long

        P =V/M
        A =P                        ' integer value
        P = P - A           ' decimal
        Ris = P * M                 ' module
        if Ris < 0
                Ris =Ris + M        ' only positive value
        endif
        Modulo= Ris
endfunction

'Task Plc
VirtualEnc=Module(MasterEnc,ImpulsiMaster)

'reset flag “fatto”
if CammaBitP1.Fatto=1 && VirtualEnc< CammaBitP1.Fase
        CammaBitP1.Fatto=0
endif
```

3.4 CfiltroVol.vco – Filtering for handwheels or ENCODERS

Contains objects which filter a variable, giving an average increase in output.
Its use is associated with values that are read from an external encoder and subsequently used as axes movements eg:

HANDWHEELS
ENCODER MASTER FOR CAM

Not having such a synchronism with the system may experience an effect NOISE on axis SLAVE.
To avoid this CfiltroVol carries on an average VALUES beds.

Hardware ***All***

Property

N. elementi	Number of elements that make up the filter. The higher the number, the greater the effect of the filter, but the slower the response of the system. Values = 10 gives a good compromise. Not managed in run time
Molt. Filtro	Value of the multiplier of calculated items. If equal to 1, the pulses of the SOURCE MASTER are transferred to the engine in a direct . Not managed in run time but can be changed with “Moltiplica” method
Encoder	Encoder source Variable to filtering. Not managed in run time
Variabile	Destination Variable. PDO or PID FILTER etc. Not managed in run time

Methods

Enable	ObjectName.Enable=True Filetr enable ObjectName.Enable=False Filter disable
Moltiplica	ObjectName.Moltiplica = Multiplication value change

Event

No

3.5 CInterpPos.vco

OBSOLETE

3.6 MonoAx.vco – MONOAXIS positioner

Monax is an object that is responsible for the complete management of an axis that is CanOpen, +-10V with encoder feedback, STEP, DIR, etc. ETHERCAT.
This contains all the properties, methods and events for the complete management of a positioner evolved COMPLETE SYSTEM basically a single axis of RAMP, LIMIT, EXTERNAL MANAGEMENT electronic handwheel, etc. POTENTIOMETER OVERRIDE.
The 'object operates all its calculations on a variable which is then associated to' axis. (PDO if CanOpen, PID filter, etc..)

Hardware ***All***

Property

abvol	1 Enable the electronic handwheel
volantino	Varieble ENCODER for electronic handwheel. Not managed in run time
uscita	Variable Output poisition (PID Filter,PDO CanOpen). Not managed in run time
Vel	Speed Axis. Impulse at cycle
Vmax	Speed Max
Acc	Axis Acceleration
Dec	Axis Deceleration
Abs	1 for absolute position (origin axis) 0 for relative position (to actual position)
Vper	% feed. (related a tmaxvper) This value can be a external potentiometer connected to analog input
MaxVper	Max value for VPER
LimitSwP	Positive limit software (value in impulse)
LimitSwN	Negative limit software (value in impulse)
LimitHwP	BIT Variable that indicate the POSITIVE switch limit. Not managed in run time
LimitHwN	BIT Variable that indicate the NEGATIVE switch limit. Not managed in run time
MolVol	Multiplier for handwheel
Nelem	Number of elements for Handwheel
QuickDec	Quivk Stop Deceleration(When the limit Hardware or Software occurs)
LimitOn	Limit Enable (bit mapped) ***Bit 0 Limit Hw enabled*** ***Bit 1 Limit Sw enabled*** ***Bit 2 Limiti on handwheel enabled (automatically bit 0,Bit are enabled)*** ***Bit 3 Limit on move enabled (automatically bit 0,Bit are enabled)***
Fczero	BIT variable for home switch. Not managed in run time
Vzero	First speed for homing. Not managed in run time
Vfine	Fine speed for homing. Not managed in run time
Senso	CW or CCW rotation axis 1 CW 0 CCW

GLOBAL VARIABLES

STATUS WORD (Bit Mappedread only)

ObjectName.status

Bit 0	***Control enabled***
Bit 1	***Axis Move***
Bit 2	***Handwheel enabled***
Bit 3	***Move direction***
Bit 4	***Positive Limit HW occurs***
Bit 5	***Negative Limit HW occurs***
Bit 6	***Positive Limit SW occurs***
Bit 7	***Negative Limit SW occurs***
Bit 8	***Homing in progress***
Bit 9	***0 Relative move– 1 Absolute move***

ObjectName.Post	Actual Position (read only)
ObjectName. _vela	Actual speed (read only)

Methods

ObjectName.enable	1 Control enabled 0 Control disabled
ObjectName.Start	1 start move to target 0 stop move
ObjectName.Quota	Target position (long value +/-)
ObjectName.StartHome	1 Start homing Homing sequence ***1 Negative move at VZERO up to Fczero is setted*** ***2 Positive move at VFINE up to Fczero is resetted*** ***3 Negative move at VFINE up to Fczero is setted*** 0 Stop homing
ObjectName.Home	Preset value for home

Event

OnEndMove	Called when the move is finished

Exampe for homing

Global Variables

StatoRicerca long

```
'Init Main
MonoAx1.enable=true
MonoAx1.Vzero=1000          ' Feed homing
MonoAx1.Vfine=200           ' fine Feed homing
MonoAx1.Acc=20              'Acc
MonoAx1.Dec=20              'Dec


'Insert this code in the main
'Starthoming by Bit (eg digital input)
If StartHoming=true
        MonoAx1.StartHome=1
        StatoRicerca=10
Endif

CicloZero()         'Call for check state home

'Insert this code in functions MAIN
```

```
Function CicloZero() As Void

        if StatoRicerca=0
                return              ' not home status
        endif
        Select StatoRicerca
                case 10
                        'Check status Word for homing finished
                        if  MonoAx1.status & 0x100          ' Homing in progress
                                return
                        else
                                StatoRicerca=20
                        endif
                case 20
                        ' Preset axis to 0
                        MonoAx1.Home=0
                        StatoRicerca=0              ' End Homing
        EndSelect
EndFunction
```

Motion Example for 4 **positioning** ***with set out when the movment is finished***

Global Variables

```
VectPos(4) Long , VectVel(4) Long , PuntPos Long , StatoCiclo long
'Init Main
MonoAx1.enable=true
MonoAx1.Acc=20                  'Acc
MonoAx1.Dec=20                  'Dec
MonoAx1.Abs=1           'ABS mov
'positioner array
VectPos(0)=3000 'Pos 1
VectPos(1)=5000 ' Pos 2
VectPos(2)=8000 ' Pos 3
VectPos(3)=3500 ' Pos 4
'Feed Array
VectVel(0)=6000  ' Pos 1
VectVel(1)=6000  ' Pos 2
VectVel(2)=3000  ' Pos 3
VectVel(3)=8000  ' Pos 4
Start Motion Insert this code in MAIN
If StartMotion=true
        StatoCiclo=10
Endif
CicloMotion()     'Call for check state motion
'Insert this code in functions MAIN
Function CicloMotion() As Void
        if StatoCiclo=0
                return
        endif
        Select  StatoCiclo
                case 10
                        MonoAx1.Quota=VectPos(PuntPos)          ' Load Pos
                        MonoAx1.Vel=VectVel(PuntPos)    ' Load Vel
                        MonoAx1.Start=true                      ' Start Motion
                        StatoCiclo=20
                case 20
```

```
                    'Check status word
                    if  MonoAx1.status & 2    ' Mov in progress
                            return
                    else
                            OUT1=true                ' Enable out
                            Tempo=100/TAU ' Delay 100 Ms
                            StatoCiclo=30
                    endif
            case 30
                    if Tempo=0
                            OUT1=false        ' Disable oput
                            if  PuntPos=0     ' End cycle
                                    MonoAx1.Quota=0 ' move to 0
                                    MonoAx1.Vel=10000
                                    MonoAx1.Start=true
                                    StatoCiclo=0
                            else
                                    Inc PuntPos       'Increase  index
                                    StatoCiclo=10
                            endif
                    endif
    EndSelect
EndFunction
'Insert this code in TASK PLC
if Tempo>0
    Dec Tempo ' Tempo=Tempo-1
endif
```

Example management Potentiometer ovveride on NGM EVO board (analog input 1)

```
'Init Main
MonoAx1.enable=true
MonoAx1.MaxVper=4096 ' 12 Bit for analog input NGM EVO

' task PLC
MonoAx1.Vper=Ng_Adc(0) ' Read the first analog input
```

Example management handwheel by Encoder Variable on NG35 Board (ch 1)
Property
MonoAx1.volantino Encoder
Global Variable
Encoder Long

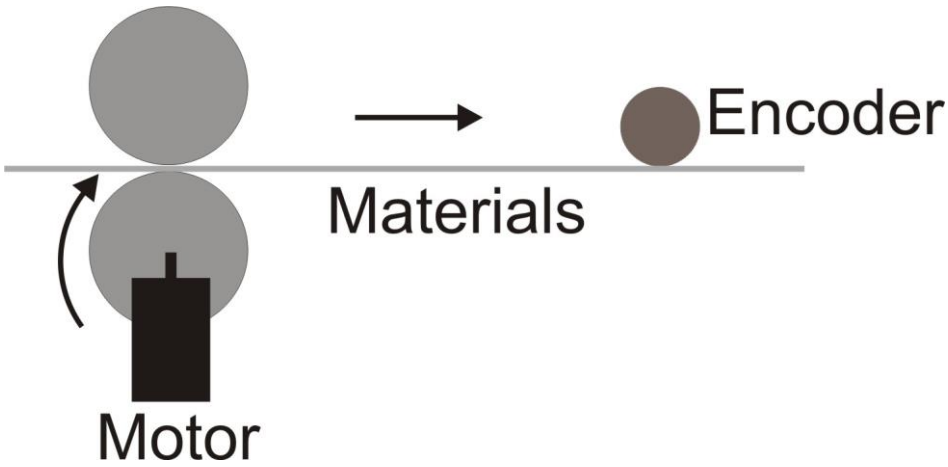
```
'Init Main
MonoAx1.enable=true
MonoAx1.AbVol=1         ' Enable Handwheel
MonoAx1.MolVol=1        ' Set multiplier to1
MonoAx1.Nelem=10        ' 10 deep filtering

' task PLC
Ng_Enc(0,Encoder())     ' Read the ch 1 to NG35
```

3.7 MonoAxEnc.vco – Double loop encoder

This class manages a comprehensive positioner that unlike MonoAx, making a precise positioning of an external encoder. Generally this is used when you have problems SLIP MAERIALE (eg Cutting sheet metal).
The position is managed by an encoder in order to recover any loss

Hardware ***All***



The object manages the placement of a Motor (Brushless etc..) By the encoder is made up for the slippage .

Property

Enable	**True** Enable the motor control **False** Disable motor control
Uscita	Variable Output poisition (PID Filter,PDO CanOpen). Not managed in run time
Vel	Speed Axis. Impulse at cycle
Vmax	Speed Max
Acc	Acc

Dec	Dec
Abs	1 for absolute position (origin axis) 0 for relative position (to actual position)
Vper	% feed. (related a tmaxvper) This value can be a external potentiometer connected to analog input
MaxVper	Max value for VPER
QuickDec	Quick Stop deceleration (When the limit Hardware or Software occurs)
Senso	CW or CCW rotation axis 1 CW 0 CCW
Posr	Variable for external encoder (this value must be converted in micron)
Spazio Frenatura	Anticpo for engine braking before you get to measuring This value anticipates the position of the target set. The motor is stopped at this value and then a precise positioning is done via external encoder speed calculated by the system

Methods

Start	Start to Target **True** Start Axis **False** Stop Axis
Quota	Axis Target
Home	ZERO positioin Axis (set value for zero position)

GLOBAL VARAIBLES

STATUS WORD (Bit Mapped read only)

ObjectName.status

Bit 0	***Control enabled***
Bit 1	***Mov in progress***
Bit 2	***Reserved***
Bit 3	***0 CCW move***
	1 CW move
Bit 4	***Reserved***
Bit 5	***Reserved***
Bit 6	***Reserved***
Bit 7	***Reserved***
Bit 8	***Reserved***
Bit 9	***0 Abs move***
	1 Rel move

Event

OnEndMove Called when the target is reached (or stop move)

3.8 CobjInterpola.vco – Multiprocess interpolator up to 9 axes

The class CObjInterpola is the basis for interpolation AXES. Can be inserted on any type of hardware and the number of axes to be interpolated is configurable as is configurable also the type of the axes:

CanOpen
+/-10V
Step/Dir
Ethercat

These can also be combined with each other.
In general the various functions of handling interpolator working on a vector of units of n positions, where n is the number of axes to manage. Before each operation will then need to enter into this vector, for each of the axes managed, the values of the target shares. Even the number of strokes of the buffer handling (lookahead).
Obviously, the programmability of No. of axles and traits, is limited by the availability of system memory, then you must make the correct proportion between these characteristics.

In the same system you can insert more objects interpolator

For details of the various methods and properties, see the manual GUIDE TO USE OF VTB chapter:

AXIS INTERPOLATION FUNCTIONS

Hardware ***All***
generally these properties are changed by ISONS application
Property

N.assi	Number of interpolator Axes. - Not managed in run time
N.tratti	**lookahead** deep. Not managed in run time
Vper	Internal variable for override axes (generally analog input). Not managed in run time
Div.Vper	Numbero of division for Vper (analog resolution). Default 4096. Not managed in run time
Abilita Arcto	Enable the arcto functions. Not managed in run time
ACC	Acceleration axes. (same deceleration).
SGLP	Angle limit stop on the edge of the tract
PC()	Axes position vector. The len of vector is the same of axes number

Methods

Stop	Axes stop with wait for axes at vel 0
Fstop	Axes stop without wait for axes at vel 0
Move	Return the movment stato (1 axes move – 0 axes stop)
Moveto	Linear interpolation 3D
Lineto	Linear interpolation 2D (only on work plane, the axes outside the work plane are move in transported mode)
Arcto	Circular Interpolation
Preset	Axes Preset
Setpiano	Set Work Plane (X,Y-X,Z ecc).

Event

No

Examples

Feed on Fly

Through the two properties and **Vper and Div.Vper** is possible to modify the fly, ie during the execution of the various sections, the speed of interpolation. In practice the speed that is set for each movement, at each sampling is then multiplied by the ratio V to / Div.Vper. So if for example you want to control the speed of movement with the reading of an external potentiometer, whereas the maximum value of the ADC NGM EVO is 4096, just enter 4096 and as precisely as Div.Vper V to assign the variable in which it is reads the value of the potentiometer.

Property Object eg: on board NGM EVO

Vper = ngadc (Internal variable ngdac type long)
Div.Vper = 4096 (Rersolution analog input NGM EVO 12 BIT)

Task PLC

```
Ngadc=ng_adc(0)  'read the first analog input
```

The interpolator will automatically vary the speed of movement to vary the analog input (potentiometer). It works on all aspects of handling, MoveTo, LineTo, ArcTo.

Preset Axes

ObjecName.preset(vect_pos())

This method, gives the possibility to set the current position of the axes. May for example be useful in the case of a zero search to switch.
A procedure has been completed, with the axes stop in the desired position

```
pos_vect(0)=0    'Axis X
pos_vect(1)=0    ' Axis Y
pos_vect(2)=0    ' Axis Z
pos_vect(3)=0    ' Axis A
pos_vect(4)=0    ' Axis B
pos_vect(5)=0    ' Axis C
ObjectName.preset(pos_vect())
```

Set work plane

ObjecName.*setpiano(ax1,ax2)*

Set the work plan and LineTo ArcTo axes indicated. By default, the plan is set to X, Y (ax1 ax2 = 0 = 1).
can not be equal to ax1 ax2. Aces take the following numbers:

0 X
1 Y
2 Z
Etc…

MoveTo – Linear interpolation

ObjecName.*moveto(vel,stop,vect_pos)*

Moving with linear interpolation of the axes indicated at speed **VEL**. The speed is calculated on all axles in motion. The parameter **STOP** defines whether the axes should still stop at the end point or continue to the next movement.
This provides that there are more entries in the buffer of the movements.

Parameters

vel	Speed interpolation (mm/min)
Stop	Stop on end point
vect_pos	Axes target vector

Return

Char	0 Not insert in the buffer (buffer is full while up to the function return 1)
	1 Insert in the buffer

LineTo - Linear interpolation on Work Plane

ObjecName.*lineto(vel, vect_pos)*

Moving with linear interpolation of the axes selected for the work plan. The calculation of the velocity (**vel**) of interpolation is performed on the axes of the working plane, is not considered the speed of other possible axes simultaneously in motion. It must therefore pay attention to the case of very short movements on the floor, associated, for example, for a movement along about a third axis. This is to go into place simultaneously to the two axes of the plan, it had no speed considered in the overall calculation, will tend to position themselves with a "snap" that will not always executable.
The stop of the axes is controlled by the property SGLP on the two axes of the work plan set. If the corner formed by two portions is greater than SGLP is still carried out a stop.
This provides that there are more entries in the buffer of the movements.

Paramers

vel	Speed interpolation
vect_pos	Axes target vector

Return

Char	0 Not insert in the buffer (buffer is full while up to the function return 1)
	1 Insert in the buffer

ArcTo – Circular interpolation

ObjecName.*arcto(vel, sense, vect_pos,i,j)*

Movement circular interpolation axes setup of the work plan, with the center.
The axes of the working plane making an interpolation of the circular type, while the other axes of the linear type.
In a similar way to LineTo, the speed speed is calculated on the plane and the property SGLP identifies the stop on the next leg.
The sense of 'clockwise or counterclockwise circular interpolation is determined by the "**sense**" parameter.

Parameter

vel	Speed interpolation
sense	2 CW, 3 CCW
vect_pos	Axes target vector
i,j	Center coordinate of arc

Return

Char	0 Not insert in the buffer (buffer is full while up to the function return 1)
	1 Insert in the buffer
	>1 Arc Error

Move – Status movment
ObjecName.*move()i*
return the status of movment

Parameter
No

Return
char **0** All axes stop
1 Axes move

Stop – Stop Axes
ObjecName.*stop()i*
Stop motion with programmed acceleration and still waiting for axes

Parameter
No

Return
No

Fstop – Stop Axes
Nomeoggetto.fstop()
Stop motion with programmed acceleration WITHOUT axes still waiting

Parameter
No

Return
No

3.9 CstdCanOpen.vco – DRIVES CanOpen DS301 DS402

CanOpen Drives CIA DS 402 o DS301. The library contains brands of commercial drivers. In the case where the brand is not present, it is reported an OBJECT STANDARD DS402.

Hardware ***All***

Property

Nodo	CanOpen node. Not managed in run time
Modo	Mode. 0 = Position Mode 1 = Velocity Mode 2 = Interpolation Mode
Velocita	Numerical value of speed of movement. The speed is referred to as the unit of measurement of specific DSP GROUP FACTOR 402 (not for interpolation mode).
Quota	Numerical value of the share of target displacement. The target is referred to as the unit of measurement of specific DSP GROUP FACTOR 402.
Abs	**True** for **absolute** value to target **False** for **relative** value to target
EnStato **True**	enable events **OnEndMove** e **OnError** **False** disable events **OnEndMove** e **OnError** **WARNING** **With EnStato = true is enabled the sending SDO line OBJECTS** **This could cause a slowdown in the application.** **Therefore, we recommend enabling EnStato only when necessary**

Methods

Enable	**True** the driver
	False disable the driver
Start	**True** Start position to target
	False stop motion
Home	Preset home value
Posr	Read actual position
Post	Read desired position
PosE	Read deviation position

Events

OnEndMove	Occurs at the end of the movement, if enabled EnStato
OnError([Coderr as Long)	Occurs when an error occurs on the control axes. EnStato if enabled. Codeerr contains the error code.

HOME REFERNCE

Using the method **ObjectName.Home** = value defines the location of Home with the assigned value. Value can be any variable of VTB. So **ObjectName.Home = 0** defines the zero point as the current position of the axis, **ObjectName.Home = 1000** defines the position of the axis at the current point has the value 1000.
In the following example is represented the operation of a motor object representing a significant ease of management. The simple design and consists of multiple objects on a page, for clarity is shown the object and its associated code to event. Obviously, the object must also be present motor, which in this case has as its name motor1.

SET INTERPOLATION MODE

In inteprolation mode you can connect this object to all OBJECTS MOTION operating variables of generic ex: MONOAX, etc. COBJINTERPOLA.
In this case it is necessary that the driver in question is able to use this type of operation.
Also you must have configured via the CANopen PDO CONFIGURATOR share interpolated.

To use the interpolator so drivers must be in mode 2 and start = true:

asse.modo=2
asse.start=true
Please note that this WAS the platform sends HARDWARE time constant (defined by the sampling of TASK PLC) positions for each axis DRIVER.
Therefore in order to avoid malfunctions that quotas should be generated by the system are consistent with those of drivers.

Generally the main problems occur when it is necessary to establish a common position of the platform HOME HARDWARE and DRIVER.
In this condition it is necessary first of all presets on the platform, so the driver from disabiliare interpolation, to make sure he does not consider the shares sent by the system until the two values do not agree.

Correct procedure for axis Preset in interpolation mode:

```
asse.modo=0      'driver in position mode
asse.start=false  ' disable pdo
Hardware platform preset  (interpolatorepreset... ecc.)
asse.home=Hardware platform preset

'a this point the position on platform is  the same of driver

asse.start=true   'Enable PDO driver
asse.modo=2       'driver in interpolation mode
```

3.10 CstdGear.vco – Electrical Gear

Contains objects that define a digital lock that moves with SLAVE relationship reportedly set to MASTER. The report is generated by two parameters KEM and KED (multiplier and divider pulses). In essence, the slave axis moves with the following definition:

Imp Slave = Imp Master * KEM / KED

Using the two parameters you can select any gear ratio.

KEM = 1 e KED =1 ***Ratio 1:1***
KEM = 2 e KED =1 ***Ratio 2:1***
KEM = 1 e KED =2 ***Ratio1:2***

The axis SLAVE can be enabled or disabled through the method ENABLE, to do so as to function as a simple positioner.
The electrical axis ratio can be changed to "In Flight".
The MASTER can be any numerical size coming for example from an encoder.

The object **CStdGear** operates on a variable GENERIC, which may in turn be associated with PDO CanOpen, Ethercat, to a fiktro or PID output to a STEP / DIR.

Hardware ***All***

Property

KEM	Multiplier Ratio
KED	Divisor Ratio
Master	Source Variable associated to MASTER Not managed in run time
Slave	Destination Variable associated to Slave (PDO, STEP/DIR,PID etc) Not managed in run time

Methods

Enable	**True** Enable the electrical gear
	False Disable the electrical gear

Events
No

Example management Electrical Gear by external Encoder connected to NG35 NGIO and KEM/KED changed by digital Inputs

Global Variables
Encoder Long
GenericOut Long
Property CStdGear
KEM=1
KED=1
Master=Encoder
Slave=GenericOut

```
'Init Main
CStdGear.enable=true

'task Main
if Input1=true              ' Input 1 set ratio  2:1
        KEM=2
        KED=1
```

```
endif
if Input2=true          ' Input 2 set ratio 1:2
        KEM=1
        KED=2
endif
'Insert code in Task PLC
Ng_Enc(0,Encoder())     ' Read Master encoder on ch 1
```

3.11 CstdStep.vco – Step Dir Axes on board NGQUARK with CanOpen

The **CStdStep** contains objects that deal with the motion control axes of Stepper PROMAX series NGQUARK using a card connected to the Master via the CAN OPEN protocol.

Hardware ***All***

Property

Nodo	NGQUARK CanOpen Node. Not managed in run time
Quota	Target Position.
Stato	**True** Enable Event **OnEndMove** **False** Disable Event **OnEndMove** **WARNING** **Enabling this property, the system generates cyclically SDO that could in some cases slow the cycle of the application. It is recommended that this option only when you are in a position**

Methods

The methods with postfix 'A' refers to the first channel of the card, 'B' to the second channel, 'C' and the third 'D' on the fourth.

AbsA(B-C-D)	**True** Absolute motion **False** Realtive Motion
StartA(B-C-D)	**True** Start motion to Target **False** Stop Motion
QuotaaA(B-C-D)	Target Position
PosaA(B-C-D)	Read Actual Position
HomeA(B-C-D)	Homex=0 Home position
AccA(B-C-D	Acceleration expressed in Hertz sampling. Eg: Sampling = 2 Ms ACC= 10 the axis goes from 0 to 1000 Hz in 200 msec
DecA(B-C-D)	Deceleration expressed in Hertz
VelA(B-C-D)	Speed Axis (HERTZ)

Events

OnEndMoveA(B-C-D)	Occurs when the Axis reached the target position

Example for 2 Axes

```
Step1.homea=0           ' Home CH A
Step1.homeb=0           ' Home CH B

Step1.acca=50           ' ACC A
Step1.deca=50           ' DEC A
Step1.vela=500          ' SPEED A
Step1.quotaa=4000       ' TARGET A

Step1.accb=100          ' ACC B
Step1.decb=100          ' DEC B
```

```
Step1.velb=1000                 ' SPEED B
Step1.quotab=8000               ' TARGET B

Step1.starta=1                  ' Start A to Target
Step1.startb=1                  ' Start B to Target
```

3.12 CPPpos.vco – Step Dir Axes on board NGM EVO

This object manages the outputs STEP / DIR card NGM EVO as a positioning or speed. You can insert up to 4 OBJECTS to cover all 4 outputs of the board NGM EVO.
Compared to the objects MONOAX that operate as interpolator, this object is able to occupy less resources and generate higher frequencies.

WARNING
The channel of the NGM EVO should not be set as PP interpolated (see object NGM EVO_init)

Example enable 2 Ch
P-P enable Mask=3
P-P Interp. Mask=0

Hardware ***NGM EVO+... - NGM EVO***

Property

NGM EVO Channel	NGM EVO Channel. Not managed in run time
Vel	Speed for movements in the position(HERTZ)
Acc	Acceleration expressed in Hertz sampling. Eg: Sampling = 2 Ms ACC= 10 the axis goes from 0 to 1000 Hz in 200 msec
Dec	Deceleration expressed in Hertz

Methods

MoveRel(Target long)	Realtive Motion at Target (number of step)
MoveAbs(Target long)	Absolute Motion at Target (number of step)
MoveVel(Vel long)	Velocity motion at Vel (Hertz) You can change the Vel **"on Fly"**. Negative value change axis direction
Stop()	Stop Motion
Move() char	Read axis status **0** Axis stop **1** Axis in movment
Qact() long	Read actual position
Preset(Value long)	Preset axis to value

Events
No

Example

```
CPPpos1.vel=1000
CPPpos1.acc=10
CPPpos1.dec=10

if inpstart=true    && Move()=0     ' Input  start motion if axis is stop
        moverel(1000)               ' Relative motion to 1000 STEPi
endif
if inpstopt=true                    'Input Stop – Stop the motion
        Stop()
endif
```

3.13 CasseMRot.vco – Rotative Axis For Cut On Fly

This object manages a VIRTUAL MASTER rotary axis with speed change and STOP at the inside of the ZERO TURN. It can be used to CUT TO FLIGHT, THE FLIGHT etc. WELDING. The axis always rotates in one direction and performs a search of the initial homing. The axis can vary the speed inside the set around an Angle.

Hardware ***All***

Property

Vel	Speed in Rpm
Vel2	Second speed expressed as a percentage of **Vel**, This rate is applied between **Qstart** and **Qend** ***100*** ***Vel2=Vel*** ***<100*** ***Vel2<Vel*** ***>100*** ***Vel2>Vel***
Velz	Speed for Homing Rpm
Qgiro	Number for inpulse for revolution axis
Qstart	Quote of the speed change from start to VEL2 Vel. expressed in DEGREES (depends Qum)
Qend	Quote of the second speed end and return to speed Vel. expressed in DEGREES (depends Qum)
Qum	Units of measurement units default = 360 360 Degrees 3600 0,1 Degrees 36000 0,01 Degrees Etc
Acc	Acceleration expressed as a sampling pulse
Uscita	Variable associated to the real axis (PDO etc..)
Input Zero	Bit for switch HOME

Global Variables

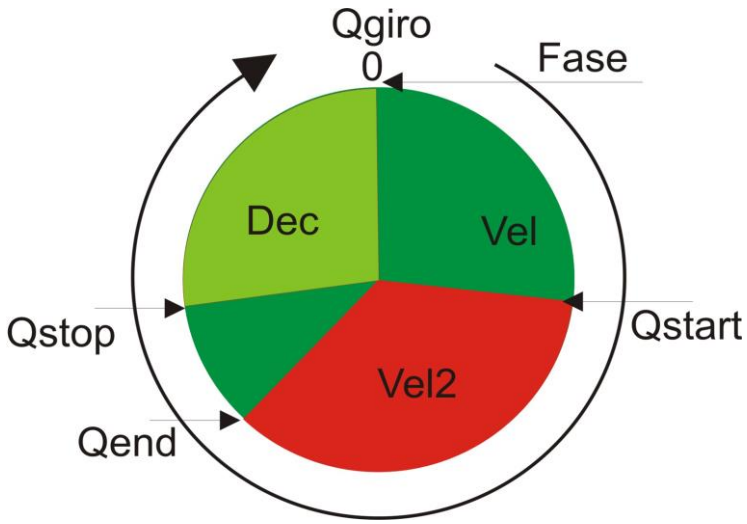
Stato (Char)	Axis Status: ***Read Only*** ***0 → Waiting for Start*** ***1 → Normal Cycle*** ***2 → Cycle in STOP phase (*expected phasing*)*** ***3 → Cycle in STOP***
rzero_ok (Char)	Homing status ***Read/Write*** ***0 → Zero search is not performed*** ***1 → Zero search OK*** **The variable can be reset to run the search again to zero at the next START CYCLE**
Qg (Long)	Actual Position (per revolution) ***Read Only***
Qt (Long)	Absolute Actual Position ***Read Only***
Sync (Char)	***S***et (value=1) when the axis passes through the ZERO ***Read/Write*** Reset Manually by application (Value=0)
Velm (Long)	Manual Speed in Rpm ***Read/Write***

Methods

Startl()	Start Motion. Performing the homing if ***Rzero_ok=0***
Stop()	Stop Motion with Phase (ZERO Position)
Stopi()	Quick Stop motion (immediate stop)
Startm()	Manual START at Speed ***Velm***
Stopm()	Stop manual start
Update()	Update the properties in RunTime (Qstart, Qend etc..) Must be called when you want to apply any changes made to the values

Events

No



Section on Speed Vel
Section on Speed Vel2
Deceleration when called Stop()

3.14 CgenFreq.vco - Frequency generator for NGM EVO

This object manages the generator FREQUENCY NGM EVO cards.
You can insert up to 4 items to handle the 4-channel card. Using the channel as a frequency generator, you can not manage it as AXIS STEP / DIR.
The corresponding channel MUST NOT BE ACTIVATED as interpolated (see object NGM EVO_init)

Eg. for activation first 2 channels

P-P enable Mask=3

P-P Interp Mask=0

This object allows the generation of frequencies up to 10 MHz without increasing the CPU time.
The object is indicated for the driving devices in frequency or to be connected to converters frequency / voltage.
The frequency is generated on the channel output STEP

Hardware ***NGM EVO+... - NGM EVO***

Property

NGM EVO channel Channel NGM EVO (valore da 0 a 3). Not managed in run time
Enable **True** Enable the frequency output
False Disable Enable the frequency output
Freq Frequency in HERTZ.
Acc Acceleration expressed in Hertz sampling. Eg: Sampling = 2 Ms ACC= 10 the axis goes from 0 to 1000 Hz in 200 msec
Dec Deceleration expressed in Hertz

Global Variables

ActFreq (Long) Frequency Actual value

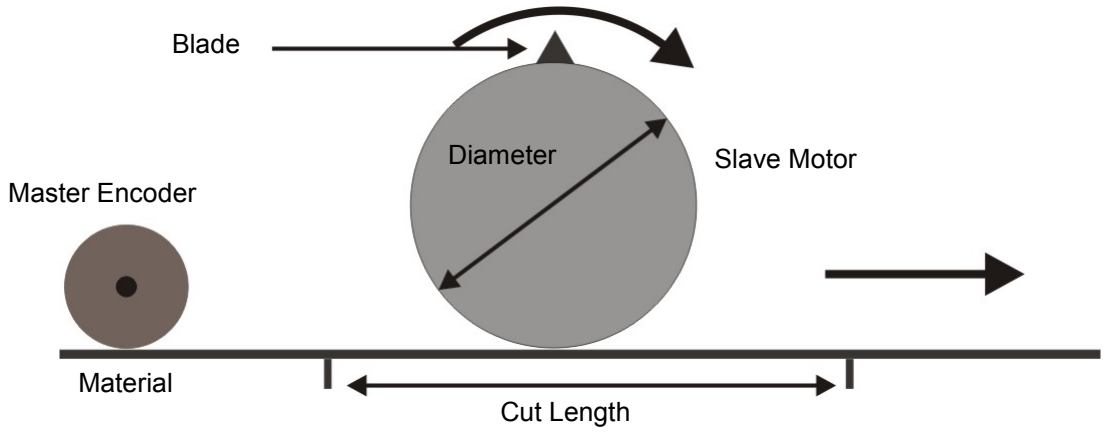
3.15 CTaglioRot.vco – Rotary Cut on Fly

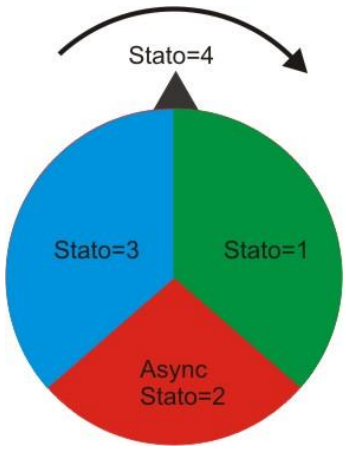
Class refers to items suitable for handling TYPE OF ROTARY CUT.

Hardware ***All***

Rotary cutting

The material flows in a continuous manner. A blade or a gripper, perform the cutting or welding of a given length of material, while it is in motion. The speed and extent of displacement of the material, are detected by an encoder (MASTER) generally put in contact with the material itself. At the time of cutting, the blade must be in perfect synchronism with the speed of the material. The phase of the rotation of the blade and 'determined by the length and velocity of the material to be cut.





Property

Len	Cut Length (mm)
Imaster	Pulse for revolution Master Encoder
Sviluppo	Development in units of material which passes around a master expressed in thousandths of a millimeter
Islave	Impulse for revolution BLADE
Diam	Blade Diameter (mm)i
Async	Angle at which the blade performs the CUT with the material During this angle the axis is in syncro with material
KSync	Ratio electrical axis when the blade cuts the material. This is expressed as a percentage.

	100 means a 1:1 ratio between blade and material. Increasing this value increases the blade speed, decreasing the blade decreases the speed
Master	Source Variable MASTER ENCODER.
Slave	Destination Variable for Slave Axis (PDO, PID ecc.)
Slave Cont.	**False** is never reset the slave axis. This can create problems of overflow in the rotation continues. The zero in this case is done externally. **True**, The slave axis is reset on any round

Methods

Update	**True** Must be used when is changed any property on run time
Restart	**True** Restart CUT. All value is setted to default
Incx	Value for shit phase at sampling
Csfx	Offset value between master and slave pulses expressed in MASTER
Chiuse	**1** - indicates blade contact with the material (read only)
Enable	**True** Enable CUT **False** Disable CUT
Stato	Blade Status (read only) ***1 First on synk*** ***2 Synk angle*** ***3 Ousside to synk angle*** ***4 Turn complete***

Events

No

3.16 NgmInit.vco – Init Board NGM EVO

The NGM_init object is automatically added to the project from the development system, when you select options in the code or terminal NGM EVO NGM EVO/LPCxx.
In particular, it allows you to set:

- Activation the RPC Link (connection to PC) and set the Baud rate
- Configuration Analog Inputs
- Configuration STEP/DIR
- Configuration on Board expansions (NGMIO)

Only ONE object NGM_init must be insert in the VTB project

Hardware ***NGM EVO+... - NGM EVO***

Property

Link RPC port	COM on NGM EVO for RPC link (Host). Value: **0 No RPC Link** **COM SER1/PROG** in this case is disabled DEBUG and download the application must be carried out manually using the BOOT / RESET card NGM EVO **1 COM SER2**
Link RPC baud	Baud rate RPC (Generally 115.200)
ADC enable mask	Analog Inputs MAK BIT **Bit 0** if set enable analog input 0 (Digital Input 0 is excluded) **Bit 1** if set enable analog input 1 (Digital Input 1 is excluded) etc.
P-P enable mask	STEP DIR MASK **Bit 0** if set enable STEP CH 0 **Bit 1** if set enable STEP CH 1 (Digital Outputs 9-12 are excludede)

Bit 2 if set enable STEP CH 2 (Digital Outputs 10-13 are excludede)
Bit 3 if set enable STEP CH 3 (Digital Outputs 11-14 are excludede)

P-P Interp. Mask Interpolation STEP MASK BIT
Bit 0 if set CH 0 in interpolation mode
Bit 1 if set CH 1 in interpolation mode
Bit 2 if set CH 2 in interpolation mode
Bit 3 if set CH 3 in interpolation mode

Num. NGM-IO Number of expansion BOARD NGMIO. The first is already included

L-Sync enable mask Reserved

L-Sync Prescaler Reserved

Methods

No

Events

No

4 TIMER

Timers class contains objects that manage the use of timers. A timer generates an event when the time is set, so the code is run which handles the event. Please note that timers and placed in the MAIN TASK OF PAGE does not have a perfectly constant cycle, this depends on the amount of source code in MAIN TASK or PAGE.

Hardware ***All***

4.1 CBitTimer.vco – Bit Timer

Timer class that contains the operating variables of the SET and RESET bits. Contain two properties which relate to the time of Bit ON and the time of Bit Off, ie the time that the variable bit must remain at logic state 1 and the time that must remain at logic state 0. The variable bit can be of any type (this can also be a variable non-BIT, in this case the value is switched from 1 to 0). If the variable bit is referred to a digital output, this is set and reset in the set times.

Property

Variabile	Variable bit
BitOn	Time BIT ON (in milliseconds).
BitOff	Time BIT OFF (in milliseconds)
Enable	**True** Enable the events
	False Disable the events

Methods

No

Events

OnSet	Occurs when the BIT change at state ON
OnReset	Occurs when the BIT change at state OFF

4.2 CStdTimer.vco – Generic Timer

Timer class that contains that manage to handle timing generic. This generates an event when the time expires.

Property

Intervallo	Timer Interval (in milliseconds)
Enable	**True** Enable the events
	False Disable the events

Methods

No

Events

OnTimer	Occurs when the time is expires (automatic reload)

5 COMMASTER

The class contains Communications management of the most common transmission protocols managed by a PLC or external devices. The data transmission is performed in RS232 or RS485 and refers to a MASTER.

Hardware ***All***

5.1 CommMaster_Modbus.vco – Master Modbus

Master ModBus RTU

Property

BaudRate	Comm Baud rate
TimeOut	Time Out for SLAVE response (millisecond). This must be more great by a slaves TimeOut
Parita	Parity 0 none - 1 odd - 2 even
N. bit car	Number bit for char
N. bit stop	Number stop bit

Methods

function .write_regn(nodo as char, addr as uint, value as *nt) as char

Preset single register func 16 ModBus RTU

Parameters

nodo	Node slave modbus
addr	Start Address register to write (Slave)
Value	Unsigned integer (values to write)

Return

0	Write OK
1	Error respons
2	Time Out
3	Data len > 127

Example

```
' Write node 1 register 10  Data in  regmodbus as uint
regmodbus=100
Valret=modbusmaster1.write_reg(1, 10, regmodbus)
if valret>0
        ' Writing error
endif
```

function .read_regn(nodo as char, addr as uint, value as *int) as char

Read single register func 3 ModBus RTU

Parameters

nodo	Node slave modbus
addr	Start Address register to read (Slave)
Value	Pointer to unsigned integer (value to read)

Return

0	Read OK
1	Error respons
2	Time Out
3	lData len > 127
4	Checksum error

Example

```
Valret=modbusmaster1.read_regn(1, 10, regmodbus()) ' Read at node 1 address 10  in  regmodbus variable
if valret>0
        ' error
endif
```

5.2 CommMaster_Omron.vco – Master omron BCD

MASTER protocol OMRON BCD

Property

BaudRate	Comm Baud rate
TimeOut	Time Out for SLAVE response (millisecond).
Parita	0 none - 1 odd - 2 even
N. bit car	Number bit for char
N. bit stop	Number stop bit

Methods

function .write_regn(nodo as char, addr as uint, value as *int,n as uint) as char

Preset register

Parameters

nodo	Node slave modbus
addr	Start Address register to write (Slave)
Value	Pointer to unsigned integer (values to write)
n	Number of registers to write

Return

0	Write OK
1	Error respons

Example

```
' Write node 1 register start 10 for 3  registers
' Data in vector regomron(127) as uint
regomron(0)=100
regomron(1)=200
regomron(3)=300
Valret=omron1.write_regn(1, 10, regomron(),3)
if valret>0
        ' error
endif
```

function .read_regn(nodo as char,ad as uint,buf as *uint,n as uint) as char

read register

Parameters

nodo	Node slave modbus
addr	Start Address register to read (Slave)
Buf	Pointer to unsigned integer (values to read)
n	Number of registers to read

Return

0	Read OK
1	Error respons

Example

```
' Read at node 1 start address 10 3 registers in regomron(127)
Valret=omron1.read_regn(1, 10, regomron(),3)
if valret>0
        ' error
endif
```

Events

No

5.3 TCP_Client.vco – Client TCP/IP

For NG 35 only

This Object manages the TCP/IP Client communication and the RPC Promax Protocol.

Property

IP address	Remote connection IP Address - ***not in Run time***
Port	Remote connection Port - ***not in Run time***
Idle TimeOut	Time out for inactivity connection (seconds) - ***not in Run time***
RPC TimeOut	Time out for RPC responses (milliseconds) - ***not in Run time***
bytes_received	Number of Bytes in receive buffer ***– Read Only***
status_connected	True - connection occurred ***– Read Only***
status_closed	True - connection closed ***– Read Only***
status_abort	True - connection closed (by remote IP or error) ***– Read Only***
status_overun	True- Data lost ***– Read Only***

Methods

These Methods manages the TCP/IP CLIENT communication.

function .connect(wait_time *as long*) *as char*

Connection request at remote IP Address and PORT number setting in the properties.
This function, waits for remote response or **"wait_time"** parameter.
WARNING: The connection manage, is not dependent by **"wait_time"** parameter, the parameter **"wait_time"** is used only for exit to function. The system try to establish connection. The TIME OUT, occurred when the bit **status_closed** or **status_abort** is true. If the bit **status_connected** is setted, the connection is established regulary.
Do not execute another function connect, up to that the TIME OUT is not finished.

Parameters

Wait_time	Time for waiting connection (see above)

Return

>0	Connection OK
-1	Connection Error
-2	Wait Time finished

Example

```
' Open connection
TCP.connect(0)   ' In this example is not used the time out Wait_time (the function return immediatly)
                 ' but the bit status_connected is dinamically read
'……..
if TCP.status_connected
                        ' connection activated
' …….
' …….
endif
```

function .close() *as void*

Connection close request. Terminates the active connection, freeing the system resources.

function .send(buf as **char, len as uint*) *as int*

Sending Data to active connection. This function, sends the bytes and returns immediately. If the network errors are occurred, the system try automatically to resend the data bytes for more time. If the network errors are persistent , the connection is closed.

Parameters

buf	Data pointer to send
len	Number of Bytes to send

Return

>=**0**	Number of Bytes sent
-**1**	Error data sent

Example

```
command(100) as char
bufrx(100) as char
nbyte as int

strcpy(command(),"START") ' Copy strin START in command
TCP.send(command(),5) ' send the string START

' Wait response
While true
        nbyte=TCP.recv(bufrx(),20) ‘ waits for 20 bytes received
        If nbyte>0 ' Data process
                .
                exitwhile
        endif
loop
```

function .recv(buf as **char, len as uint*) *as int*

Bytes reading. With the property bytes_received (only read) is possible know the bytes number that are present in the system receive buffer. The **len** parameter, indicates the number of bytes to discharge from system receive buffer.
The return value, indicates the effective number of data read (normally it is equal to **len** parameter).
If the value is less to len, you must try to call the **function.recv** more times.

Parameters

buf	Pointer to destination buffer
len	Max number of data reading

Return

>=**0**	Number of data read

Example

```
command(100) as char
bufrx(100) as char
nbyte as int

strcpy(command(),"START") ' Copy strin START in command
TCP.send(command(),5) ' send the string START

' Wait response
While true
        nbyte=TCP.recv(bufrx(),20) ‘ waits for 20 bytes received
        If nbyte>0 ' Data process
                .
                exitwhile
        endif
loop
```

5.3.1 PROMAX RPC functions protocol

The RPC is the protocol for communicate by Promax systems. The SERVER is listening at the PORT **6000**.
So for use the RPC protocol, is necessary open a connection at this port.

function .rpc_write(ad as *long, len as uint,* buf as **char*) *as int*

Data writing in the remote boards.
The system sends a request to write a data array at the remote memory Address.
For speeding the data transfer, the system do not waits that the data are effectively writings in the remote system.
If the syncronism is request, is necessary call the function rpc_read, after the call function rpc_write. (this mode guarantees a perfect synchronism)

Parameters

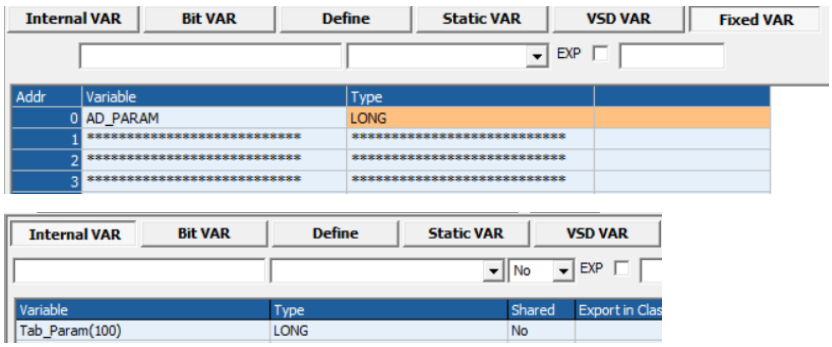
ad	Remote System Memory Address to data writing
len	Bytes number to write
buf	Data pointer to send

Return

>=**0**	Number of data writings
-**1**	Send error
-2	Time out elapsed

Example

In the SLAVE board define **AD_PARAM** in the fixed variables (es at ADDR 0) and an array named Tab_Param in the internal variables:



Assigned in the Fixed **AD_Param**, the **tab_param** address(insert this code in the TASK PLC INIT)

AD_PARAM=tab_param()

In the MASTER system, define **AD_PARAM** in the fixed variables at the same address of the SLAVE board, and define the same array **tab_param** (this array must be the same dimension and data type)

```
' open the connection
TCP.connect(0)
.
.
Tab_param(0)=20
Tab_param(1)=30
.
.
TCP.rpc_read(AD_PARAM(),4, AD_PARAM())          ' read the remote pointer
TCP.rpc_write(AD_PARAM,8,tab_param())           ' write  8 bytes (2 long)
```

If is used a NGM-EVO for read the pointer, is necessary use the manual address :

```
TCP.rpc_read(FIXED_EVO+ADDR,4, AD_PARAM())    'Read Pointer
Where FIXED_EVO=536874496
```

function .rpc_read(ad as *long, len as uint,* buf as **char*) *as int*

Data read from the remote board.
The system send a request to read the data from a memory address of remote board and waiting the response. The data read are inserts at the buf pointer.

Parameters

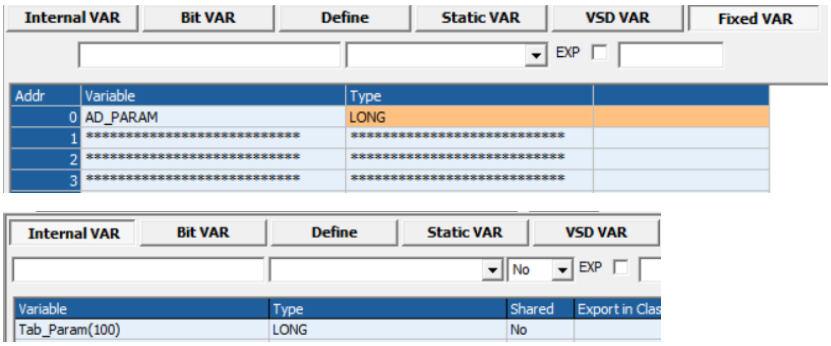
ad	Remote System Memory Address to data read
len	Number of bytes to read
buf	Destination data pointeri

Ritorna

>=**0**	Number of data read
-**1**	Send error
-2	Time out elapsed
-3	Time out RPC response

Example

In the SLAVE board define **AD_PARAM** in the fixed variables (es at ADDR 0) and an array named Tab_Param in the internal variables:



Assigned in the Fixed **AD_Param**, the **tab_param** address(insert this code in the TASK PLC INIT)

AD_PARAM=tab_param()

In the MASTER system, define **AD_PARAM** in the fixed variables at the same address of the SLAVE board, and define the same array **tab_param** (this array must be the same dimension and data type)

```
' Open the connection
TCP.connect(0)
.
TCP.rpc_read(AD_PARAM(),4, AD_PARAM())        ' read the remote pointer
TCP.rpc_read(AD_PARAM,100,tab_param())        ' read di 100 bytes
```

6 MODBUS

The class contains the management SLAVE MODBUS protocol MODBUS RTU and TCP / IP

Hardware ***All (NG35 and Pec70 for TCP/IP)***

6.1 CModbus.vco – Slave Modbus RTU/TCPIP

Modbus RTU RS232

Property

Nodo	Node slave
BaudRate	baud rate
PtData()	Array Data Register
Max Len Data	Data Register dimension
TimeOut	Master Time Out (millisecond) This must be smallest by a MASTER TimeOut

Methods

No

The following requests are handled MODBUS RTU:

Function Code 3 Read Multiple Registers
Function Code 6 Preset Single Registers
Function Code 16 Preset Multiple Registers

Events

No

Modbus TCP/IP

Property

Nodo	Node slave
IpAddress	Slave IP Adrress eg. “10.0.0.80”
Service Port	Slave IP Port (default 502)
PtData()	Array Data Register
Max Len Data	Data Register dimension

Methods

No

The following requests are handled MODBUS RTU:

Function Code 3 Read Multiple Registers
Function Code 4 Read Input Registers
Function Code 6 Preset Single Registers
Function Code 16 Preset Multiple Registers

Events

No

7 GENERAL

7.1 Cpwm.vco – Managing of PWM output on NG-PP

This object manages the pwm output of the NG-PP. There is only one output available for pwm: the channel 4 of step output (connector J21).

Property

Chan	Index of the chanel on NG-PP (only 3)
Polarità	Set the polarity of the output signal
Freq	Set the frequency of pwm

Methods

Name.val(val) as void

Write the value of pwm duty cycle.

Val	Value (0 .. 1024)

Events

No

Note

The frequency is calculate by a divisor:

div = 75000000 / 1024 / Freq

Setting a frequency, the real one will be the upper first available. Ex. Setting 10000 the output frequency will be 10463Hz.

Theese are some frequency available:

Div	Freq (Hz)	
2	36621	Freq max
3	24414	
4	18310	
5	14648	
6	12207	
7	10463	
8	9155	
9	8138	
10	7324	
11	6658	
12	6103	
13	5634	
14	5232	
	...	
18	4069	
	...	
24	3052	
	...	
36	2034	
	...	
73	1003	
	...	
36621	2	Freq min

7.2 Cpwm.vco – Gestione uscita PWM su NGM-EVO

This object manages the pwm output of the NGM-EVO. There are 4 output available corresponding to the first 4 digital output. The first output can be configured for an analog output.

Proprietà

Enable	By bit. Enable the PWM outputs.
Polarità	Set the polarity of the output signal
Center Align	Enable the center align mode
Freq	Set the frequency of pwm
Divisioni	Number of division corresponding to 100% (2-65535)

Methods

pwm_val(id, val) as void

Write the value of pwm duty cycle.

Id	Index of the chanel (0...3)
Val	Value of duty cycle from 0 to Divisioni

ATTENTION: This is a system function, the object name not must be write.

Eventi

No

Note

The maximum frequency is: 20000000 / Divisioni

7.3 FastInput.vco – Digital Interrupt for NGIO-NGPP-NGMsX-NGQx

Digital Inputs Management for NGIO, NGPP, NGMsX, NGQx. Only some digital inputs are enable to interrupt mode. The interrupt mode, allow to read at maximum speed, the digital input.
These cards use the following digital inputs enabled for INTERRUPT:

NGIO	→Encoder ZERO INDEX (2 per board)	***Pin 3,8***	***J17-J18***
	THIS FUNCTION ON NGIO IS ENABLED ONLY ON HARDWARE REV. 2.0		
NGPP	→ Ingressi FAST INPUT 1-4 (4 per scheda)	***Pin 1,2,3,4***	***J19***
NGMsX	→ Ingresso Tacca di Zero per ogni canale (2 per scheda)	***Pin 3,8***	***J22-J23***
NGQX	→ Ingresso Tacca di Zero per ogni canale (2 per scheda)	***Pin 3,8***	***J6-J8***

Property

Card Index Card Index on the BUS
NGIO, NGPP, NGMsX – from 0 to 7
NGQx - 0

Channel Digital inputs Channel
NGIO. NGMsX, NGQx – from 0 to 1
NGPP – da 0 a 3

Methods

Name.get() as void

Updates the latch registers used for read rising and falling EDGE
(call this function first to read the edge with **.UP** and **.DN** variables)

Name.clear() as void

Reset the latch registers
This function, reset the variables .**UP** and .**DN**

Variables read only

Name.stato	Contains the actual input state (0 o 1)
Name.up	Contains RISING EDGE LATCH state. Management in INTERRUPT MODE
Name.dn	Contains FALLING EDGE LATCH state. Management in INTERRUPT MODE

Example

Insert a Fast Input Object named ***FastInput1***
Insert the following code in the Master Cycle of Main Task
Declares the following Variables:
RisingEdge1 char
FallingEdge1 char
State1 char

```
Fastinput1.get()                'get fastinput1
RisingEdge1=FastInput1.up       'ceck the rising edge
FallingEdge1=FastInput1.dn      'ceck the falling edge
State1=FastInput1.inp           'read the state
FastInput1.clear()              'reset latch up e dn
```

Events

No

8 MOTOR CONTROL PLUS

The class MotorControlPlus is a group of object similar to MotorControl descripted previoously.

Hardware ***All***

8.1 CobjPos.vco – SINGLE-AXIS POSITIONER

This is the evolution of MonoAx object, that allows to select, directly from the object, the axis type we want drive (stepper, CanOpen DS402, analogical).
Furthermore this object is derived from the CobjInterpola presenting and then will present its main feature (functions, movement buffer, acceleration mode, etc.).

Property

NOTE: the upper-case property can not be modify at run-time.

N. TRATTTI	Number of segment in the movement buffer
Vper	% feed (related to programmed speed)
Div. Vper	Vper divisor
AccQstop	Deceleration in qstop function
Acc	Acceleration / deceleration
RzeroMode	Type of homing search. See below for details
RzeroOffset	Axis offset after homing sequence
RzeroPreset	Preset quote at the end of homing sequence
RzeroVel	Speed (high) of homing search
RzeroVelf	Speed (low) of homing search
RzeroAcc	Acceleration / deceleration of homing search
Msof	Counts for one revolution
Dsof	Length of one revolution, negative to invert direction
LimitN	Software negative limit
LimitP	Software positive limit
Gioco	Value backlash in unit
Vgioco	Speed for bachlash recovery in counts for sample
MsofV	Multiplier of fly-wheel
DsofV	Divisor of fly-wheel, negative to invert direction
qvola	Fly-wheel increment
pc	Actual position before fly-wheel
qt	Actual position after sum with fly-wheel
qi	Actual position in counts
RZERO ENABLE	Compile the code for the homing search (used to reduce the code memory)
AXIS TYPE	Define the type of axis -1 = managing of axes type disabled 0 = custom axis (contact Promax to details) 1 = CANOPEN (use an object of CstdCanopen) 2 = STEP/DIR (NG-PP, NGM EVO, NGQ) 3 = STEP/DIR SLAVE (slave CanOpen NGM EVO o NGQ) 4 = ANALOGICAL (use an object of PidPlus)
VTB AXIS OBJECT	Name of the related object (only for AXIS TYPE 1 and 4)
PDO NAME	Name of pdo related variable of interpolated position (only for AXIS TYPE 1 and 3)
STEP CHANNEL	Index of step channel (0, 1, 2….) (only for AXIS TYPE 2 and 3)
STEP NODE	Node of NGM EVO or NGQ slave (only for AXIS TYPE 3)

Methods

function enable() as void
Enable axis

function disable() as void
Disable axis

function preset(q as long) as void
Preset to q position. To reset only the fly-wheel run a preset to actual position: obj.preset(obj.qt)

function move() as char
Test for movement in progress.

function stop() as void
Stop axis and wait end move.

function fstop() as void
Stop axis without waiting end move.

function qstop() as void
Stop axis and wait end move using AccQstop.

function StartHome() as void
Start homing sequence. To known when it is terminated read status_rzero.

function StopHome() as void
Stop of homing sequence.

function moveto(vel as long, stop as char, q as long) as char
Move to position setting the stop between segments.

vel speed
stop 0 – Don't stop at the end of previous segment
1 - Stop at the end of previous segment
q Target Position

Return 0 = buffer full
1 = command write in the buffer

function lineto(vel as long, q as long) as char
Move to position with automatic calculation of stop between segments. In other words it stop only if a segment is in the opposite direction of the previous one.

vel speed
q Target Position

Return 0 = buffer full
1 = command write in the buffer
-1 = position is the same of the previous one

Events

No

8.1.1 I/O bit

These bits must be associated to the phisical input of the used hardware.

Bit Name	Description
ext_fcn	Negative limit switch
ext_fcp	Positive limit switch
ext_fcz	Homing sensor
ext_tacca	Encoder index

8.1.2 Status Bit

Bit Name	Variable	Description
status_rzero		Homing in execution
status_home		Set when axis has executed the homing sequence
status_enable		Set when axis is enabled
ErrorLimitN	Error.0	Axis on negative limit
ErrorLimitP	Error.1	Axis on positive limit

8.1.3 Homing sequence

The homing sequence depends to the property RzeroMode. After homing sensor is found axis move to RzeroOffset and then the position is preset at RzeroPreset.
There are two homing speed RzeroVel (fast) and RzeroVelf (slow) while the acceleration/deceleration is always RzeroAcc.

RZERO_MODE		INITIAL DIRECTION	SEQUENCE
0	Homing at sensor on	Backward (negative)	- Backward fast to homing sensor - Forward slow to sensor off - Backward slow to sensor on - Positioning to offset position
1		Forward (positive)	- Forward fast to homing sensor - Backward slow to sensor off - Forward slow to sensor on - Positioning to offset position
2	Homing at sensor on and encoder index	Backward (negative)	- Backward fast to homing sensor - Forward slow to sensor off - Backward slow to sensor on - Continuation to first encoder index - Positioning to offset position
3		Forward (positive)	- Forward fast to homing sensor - Backward slow to sensor off - Forward slow to sensor on - Continuation to first encoder index - Positioning to offset position

4	Homing at sensor off	Backward (negative)	- Backward fast to homing sensor - Forward slow to sensor off - Positioning to offset position
5		Forward (positive)	- Forward fast to homing sensor - Backward slow to sensor off - Positioning to offset position
6	Homing at sensor off and encoder index	Backward (negative)	- Backward fast to homing sensor - Forward slow to sensor off - Continuation to first encoder index - Positioning to offset position
7		Forward (positive)	- Forward fast to homing sensor - Backward slow to sensor off - Continuation to first encoder index - Positioning to offset position
8	Homing only with encoder index	Backward (negative)	- Backward slow to first encoder index - Positioning to offset position
9		Forward (positive)	- Forward slow to first encoder index - Positioning to offset position
18	Homing at sensor on and preset at index position	Backward (negative)	- Backward fast to homing sensor - Forward slow to sensor off - Backward slow to sensor on - Preset at index position - Positioning to offset position
19		Forward (positive)	- Forward fast to homing sensor - Backward slow to sensor off - Forward slow to sensor on - Preset at index position - Positioning to offset position
22	Homing at sensor off and preset at index position	Backward (negative)	- Backward fast to homing sensor - Forward slow to sensor off - Preset at index position - Positioning to offset position
23		Forward (positive)	- Forward fast to homing sensor - Backward slow to sensor off - Preset at index position - Positioning to offset position
32	No sensor	Homing on enable	When axis is enabled it is preset at RZERO_PRESET position
64		Absolute Encoder	When axis is enabled the system is preset to axis position
128	Custom homing (by VTB code)	When Homing is executed the variable RzeroStato is set to 11. Custom code will be provide to execute the correct sequence, update the status bits (status_home, status_rzero) and reset the RzeroStato variable.	

* Read of index position must be done by VTB code. The variable RzeroStato is set at 51 and the custom code will write in qtacca variable the index position read from driver (in counts) then update to 55 the RzeroStato variable to terminate the sequence.

8.1.4 Select axis type

AXIS TYPE	
-1	Object doesn't manage any type of axis working virtually on the output position (property qt)
0	Object doesn't manage any type of axis but call the interfacing function. Setting AXIS TYPE to zero, compiler will generate some error “function not find”. Contact PROMAX to implement a custom axis.
1 - CANOPEN DS402	- Add an object from class CstdCanopen - Set VTB AXIS OBJECT with the name of the object - Write the CanOpen configuration - Set PDO NAME with the name of variable related to PDO of interpolated position - STEP CHANNEL and STEP NODE are not used
2 – STEP/DIR	- No object must be add to the application - Set STEP CHANNEL with the index of the channel on NG-PP, NGM EVO, NGQ related with the axis - VTB AXIS OBJECT, PDO NAME and STEP NODE are not used
3 – STEP/DIR on slave Canopen	- No object must be add to the application - Write the CanOpen configuration - Set PDO NAME with the name of variable related to PDO of interpolated position - Set STEP CHANNEL with the index of the channel on the slave Canopen NGM EVO or NGQ related with the axis - Set STEP NODE with the number of node of the slave CanOpen - VTB AXIS OBJECT is not used
4 – ANALOGICAL	- Add an object from class CPidPlus - Set VTB AXIS OBJECT with the name of the object - Set the eventual PID parameters - PDO NAME, STEP CHANNEL e STEP NODE are not used

9 CLASSE PID PLUS

The class PidPlus applies PID regulation to an analogical axis. It is an evolution of older object FiltroDigitale that, unlike the previous, manages directly the hardware to be used

Property

NOTE: the upper-case property can not be modify at run-time.

EnablePid	Enable the pid regulation without activating the servo enable relay
Kp, Ki, Kv, Kd	PID parameters
Err_Sat	Limit of integral component
NG ENC CHANNEL	Index of the encoder channel
NG DAC CHANNEL	Index of the analog output channel
ENABLE KP, KI, KV, KD	Enable compiling of pid components (to optimize the code memory)
Divisore	Divisor for all PID parameters
Dir	Polarity of analog output (0 or 1)
ServoErr	Parameter of servo error threshold
TServoErr	Intervention time of servo error (in mSec)
EnableDelay	Delay between servo enable relay activation and PID regulation (in mSec)
Err	State of servo error 1 error reached (only in run time)
Post	Demand Position (only in run time)
Posr	Actual Position (only in run time)

Methods

function enable() as void

Enable axis. Activate the servo enable relay and the PID regulation.

function disable() as void

Disable axis. Stop the PID regulation and disable of the servo enable relay

9.1 CRotaryKnife.vco – Rotary knife

The object allows to manage a rotary knife with trapezoidal or sinusoidal profile. It represents an evolution of object CtaglioRot.

Once set the parameters must be call the function to calculate the working cam (TrapUpdate or SinUpdate). The new cam will be activated at the begin of next turn of rotary knife.

As the previous object, it works like a slave axes of a master position from external code. For this reason evntual stop “in-phase” should be written in VTB code.

Property

Enable	Object enable
Len	Cut length (resolution 0.1mm, 0.01mm, etc)
Imaster	Pulses of encoder master
Emaster	Linear measure of one turn of encoder master (resolution 0.1mm, 0.01mm, ec)
Islave	Pulses of encoder slave (rotary knife)
Diam	Diameter of the slave (rotary knife) (resolution 0.1mm, 0.01mm, etc)
Async	Angle of synchronism in degree
Ksync	Speed of slave in synchro phase in % (100=syncro)
Vextra	Extra velocity in synchro phase in % (0=disable)
StartExtra	Point (in %) inside the synchro angle where to start extra velocity
StopExtra	Point (in %) inside the synchro angle where to stop extra velocity
Shift	Number of master pulses to correct knife phase.
Vshift	V Max number of pulses per sample of correction (Shift property)
Master	Variable containing the value of encoder master counts
Slave	Variable where object writes the slave pulses (to be pass to axes)
Cam Len	Number of points of working cam

state State of slave (read only):
0 = not initialized
1 = phase of achievement of synchro speed
2 = synchro phase
3 = phase of return to positioning speed or wait new cycle

qmaster Actual position of slave in 0.001 degree (read only)

vslave Actual speed of slave in pulse per sample (read only)

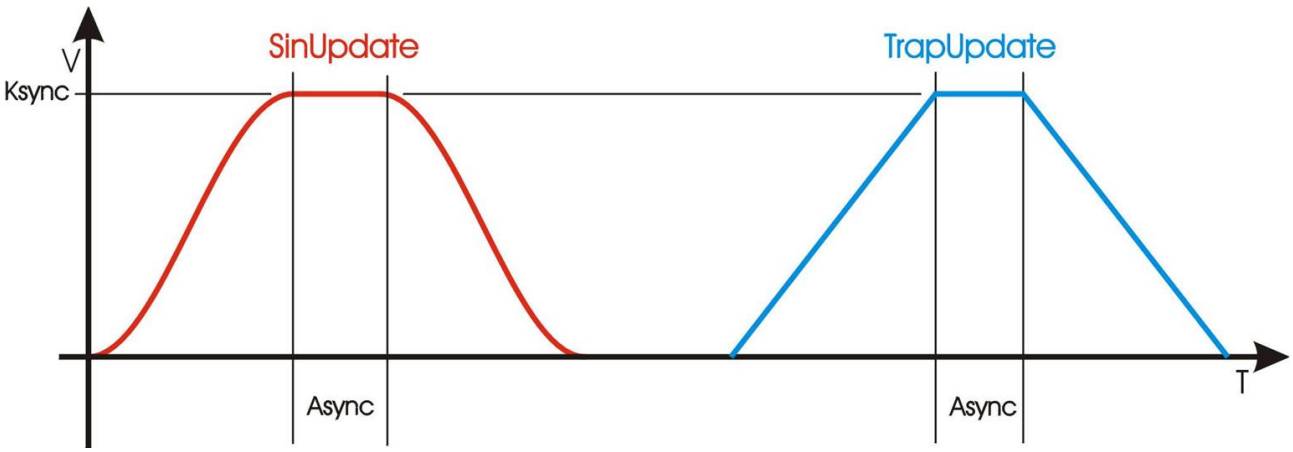
WARNING!!!
Property identifying linear measure must have ALL the same unit of measure (hundredths of millimeter, thousandths of millimeters, etc,).

Methods

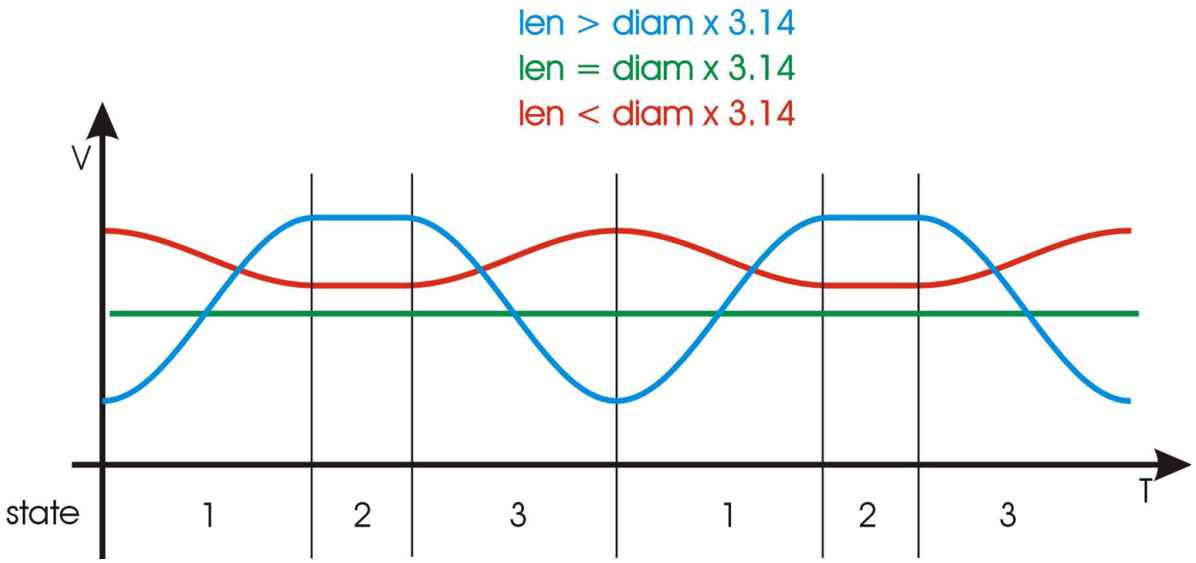
function reset() as void
Reset of internal counters and disable of the object (enable=0). The last calculated cam remains in memory. To restart adjust the external variables and set to 1 the enable property.

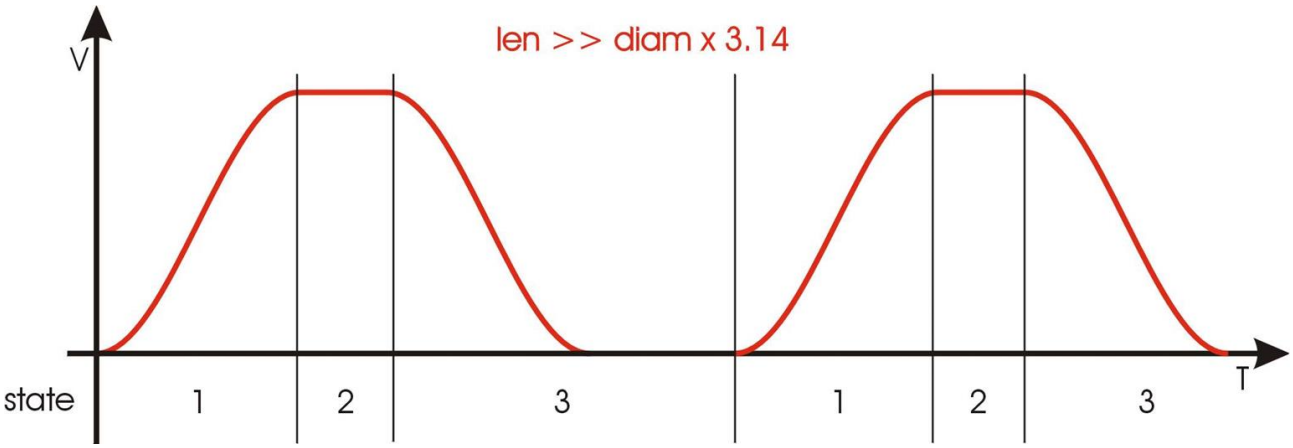
function TrapUpdate() as void
Update the parameters and calculate the cam with trapezoidal form.

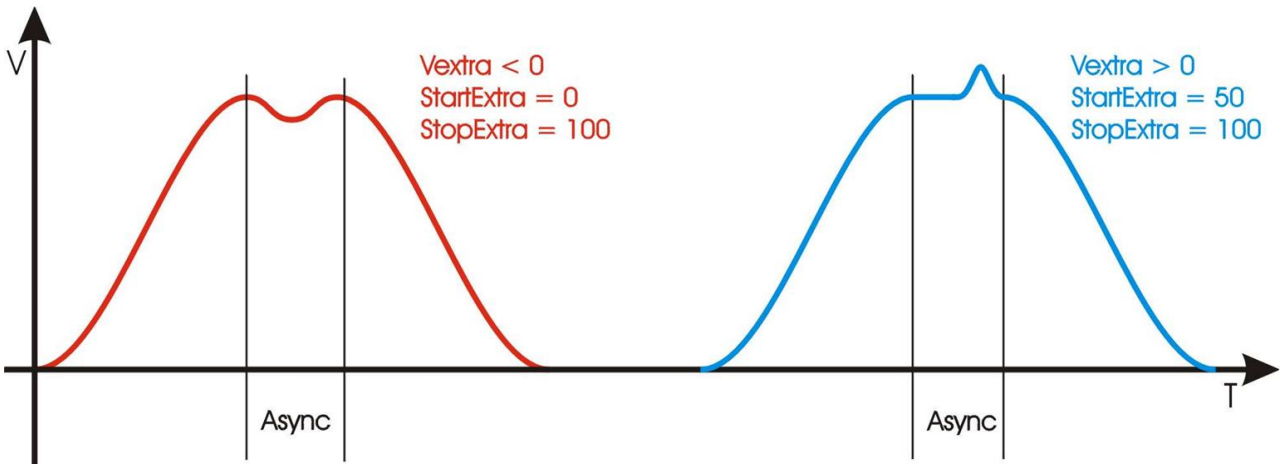
function SinUpdate() as void
Update the parameters and calculate the cam with sinusoidal form.



9.1.1 Examples







9.1.2 Re-phasing by signal mark

To obtain the re-phasing of rotary knife by external signal mark you must add some VTB code according to the operating requirements. Object provides two property: Shift and Vshift.
The first one set the number of master pulses we want to correct, the second one tells the maximum number of pulses must be correct per sample (TaskPlc). That allows “to spread the pulses” over time to avoid a sharp correction.

Index

1 PREFACE 3
1.1 CbrowserMC.vco – Browser for management FLASH DISK 3
1.2 CstdAllarm.vco – Alarms Browser 3
2 CLASS INPUTBIT 5
2.1 CstdBit.vco – BIT management 5
3 CLASS MOTOR CONTROL 6
3.1 CbitCam.vco – Management Bit CAM 6
3.2 Ccam.vco eCam /Continue eCam -Management eCAM for MOTOR CONTROL 7
3.3 CcamPulse.vco – Management impulsive BIT CAM 9
3.4 CfiltroVol.vco – Filtering for handwheels or ENCODERS 10
3.5 CInterpPos.vco 11
3.6 MonoAx.vco – MONOAXIS positioner 11
3.7 MonoAxEnc.vco – Double loop encoder 15
3.8 CobjInterpola.vco – Multiprocess interpolator up to 9 axes 17
3.9 CstdCanOpen.vco – DRIVES CanOpen DS301 DS402 20
3.10 CstdGear.vco – Electrical Gear 22
3.11 CstdStep.vco – Step Dir Axes on board NGQUARK with CanOpen 23
3.12 CPPpos.vco – Step Dir Axes on board NGM EVO 24
3.13 CasseMRot.vco – Rotative Axis For Cut On Fly 25
3.14 CgenFreq.vco - Frequency generator for NGM EVO 26
3.15 CTaglioRot.vco – Rotary Cut on Fly 27
3.16 NgmInit.vco – Init Board NGM EVO 28
4 TIMER 30
4.1 CBitTimer.vco – Bit Timer 30
4.2 CStdTimer.vco – Generic Timer 30
5 COMMASTER 31
5.1 CommMaster_Modbus.vco – Master Modbus 31
5.2 CommMaster_Omron.vco – Master omron BCD 32
5.3 TCP_Client.vco – Client TCP/IP 33
5.3.1 PROMAX RPC functions protocol 35
6 MODBUS 38
6.1 CModbus.vco – Slave Modbus RTU/TCPIP 38
7 GENERAL 39
7.1 Cpwm.vco – Managing of PWM output on NG-PP 39
7.2 Cpwm.vco – Gestione uscita PWM su NGM-EVO 41

7.3 FastInput.vco – Digital Interrupt for NGIO-NGPP-NGMsX-NGQx....41
8 MOTOR CONTROL PLUS....43
8.1 CobjPos.vco – SINGLE-AXIS POSITIONER....43
8.1.1 I/O bit....45
8.1.2 Status Bit....45
8.1.3 Homing sequence....45
8.1.4 Select axis type....47
9 CLASSE PID PLUS....48
9.1 CRotaryKnife.vco – Rotary knife....48
9.1.1 Examples....50
9.1.2 Re-phasing by signal mark....51