NGWARP

www.promax.it

VTB Software Resources

Rev. 1.0.2

© Promax s.r.l. – Via Newton, 5/G – Z.I. Malacoda – CastelFiorentino (Fi) ITALY

email:info@promax.it - internet:www.promax.it

1 Prefazione

Questo manuale spiega come utilizzare le risorse HARDWARE della scheda NGWARP e espansioni in ambiente VTB.
Per maggiori approfondimenti sul linguaggio VTB, si rimanda ai relativi manuali:

Manuale di Programmazione
Guida agli Oggetti

Gli esempi descritti non si riferiscono a nessuna applicazione reale, pertanto le loro informazioni sono puramente indicative.

2 Porta Ethernet

Le piattaforme con ETHERNET a bordo gestiscono in modo AUTOMATICO lo STACK TCP/IP. La gestione di protocolli che si appoggiano a questo è demandata a livello di applicazione. Per esempio la gestione del protocollo MODBUS-TCP è gestita completamente da uno specifico oggetto ed usa questo gruppo di funzioni. Allo stesso modo è possibile creare protocolli personalizzati.
Gli esempi delle funzioni sono riportati al fine del paragrafo

2.1 SET_IP

Setta i parametri del protocollo TCP/IP.

Sintassi

```
SET_IP(ip as *char, sm as *char, gw as *char)        as void
```

Parametri

ip Indirizzo IP della scheda
sm Subnet mask
gw Gateway

ATTENZIONE
E' necessario inizializzare l' indirizzo IP su INIT della MAIN o del TASK PLC

2.2 PXETH_ADD_PROT

Aggiunge la gestione di un protocollo che si appoggia al TCP/IP. Occorre scrivere la funzione di gestione del protocollo e passare il puntatore a questa funzione.

Sintassi

```
PXETH_ADD_PROT(port as long, fun as delegate) as void
```

Parametri

port Porta TCP sulla quale aggiungere il protocollo
fun Puntatore alla funzione di gestione del protocollo

2.2.1 Funzione di GESTIONE PROTOCOLLO

Questa funzione non è definita dal sistema ma deve essere scritta nell'applicazione. Il sistema chiama questa funzione, tramite il puntatore passato in ***pxeth_add_prot,*** ogni volta che viene ricevuto un pacchetto dati sulla porta definita per questo protocollo. Per leggere i dati occorre utilizzare la funzione **pxeth_rx** mentre per inviare dei dati di risposta andranno scritti nel buffer di trasmissione (bufftx) e uscire dalla funzione ritornando il numero di byte che si vogliono inviare.

Sintassi

```
MY_PROTOCOL(len as long, buftx as *char) as long
```

Parametri

len Lunghezza del pacchetto dati ricevuto
buftx Puntatore al buffer di trasmissione

Valore di ritorno

long Numero di byte da inviare scritti nel buffer di trasmissione

2.3 PXETH_RX

Legge un byte dal buffer di ricezione TCP/IP. Viene utilizzata nella funzione di gestione del protocollo per leggere i dati ricevuti.

Sintassi

PXETH_RX() **as char**

Valore di ritorno

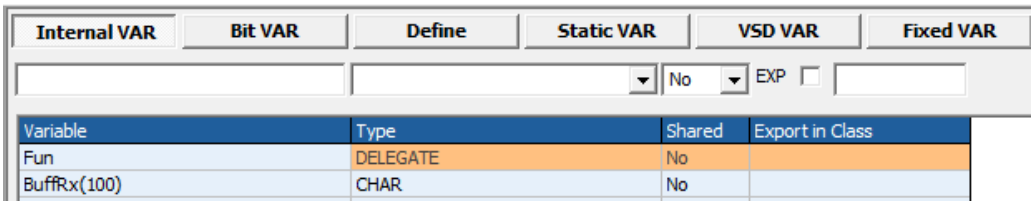
Char Dato letto dal buffer di ricezione

2.4 Esempio

Nell' esempio riportato sotto, quando ricevuto un blocco TCP/IP, vengono controllati i primi **TRE CARATTERI**. Se questi corrispondo all stringa “VTB” viene risposto “YES”, altrimenti “NO”
Viene controllato il codice ASCII dei caratteri:

V = 86 **Y** = 89 **N** = 78
T = 84 **E** = 69 **O** = 79
B = 66 **S** = 83

Variabili Utilizzate

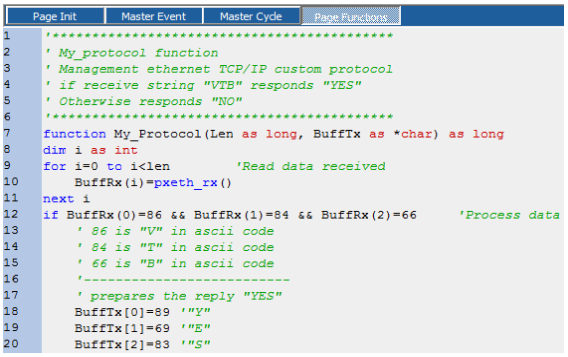


Codice in Init Main

Set_ip(“10,0,0,15”,”255,255,255,0”,0) **'IP = 10,0,0,15**
'SUBNET = 255,255,255,0
'GATEWAY = none

Fun=my_protocol
pxeth_add_prot(502,Fun)**'Add Function my_protocol to 502 port**

Codice in Funzioni di Pagina Main



```
'*****************************************
' My_protocol function
' Management ethernet TCP/IP custom protocol
' if receive string "VTB" responds "YES"
' Otherwise responds "NO"
'*****************************************
function My_Protocol(Len as long, BuffTx as *char) as long
dim i as int

for i=0 to i<len                'Read data received
        BuffRx(i)=pxeth_rx()
next i



if BuffRx(0)=86 && BuffRx(1)=84 && BuffRx(2)=66                 'Process data
        ' 86 is "V" in ascii code
        ' 84 is "T" in ascii code
        ' 66 is "B" in ascii code
        '-------------------------
        ' prepares the reply "YES"
        BuffTx[0]=89 '"Y"
        BuffTx[1]=69 '"E"
        BuffTx[2]=83 '"S"
        My_Protocol=3 ' Data len for YES 3 Chars
else
        ' prepares the reply "NO"
        BuffTx[0]=78 '"N"
        BuffTx[1]=79 '"O"
        My_Protocol=2 ' Data len for NO 2 Chars
endif
endfunction
```

Scarica l' esempio

3 Modbus TCP/IP

La Porta Ethernet può essere configurata anche con il protocollo TCP/IP MODBUS.
Ovviamente lo STACK TCP/IP può supportare multi connessione di più protocolli
Il MODBUS TCP/IP è interamente gestito da un Oggetto per VTB

3.1 OGGETTO Modbus TCP/IP

Tramite quest' oggetto è possibile gestire in modo TRASPARENTE, il protocollo ModBus TCP/IP

Proprietà principali

Nodo Nodo slave
IpAddress Indirizzo IP dello slave es. “10.0.0.80”
Service Port Porta di servizio (default 502 dedicata la ModBus TCPIP)
PtData()Registri Unsigned Integer dello slave messi a disposizione del master.
Max Len Data Dimensione del buffer PtData (numero registri Modbus)

Metodi

Nessuno

Vengono gestite le seguenti richieste MODBUS TCP/P:

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

Eventi

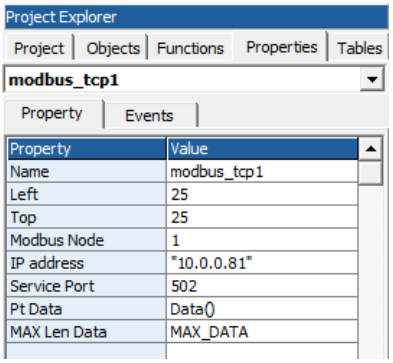
Nessuno

3.2 Esempio

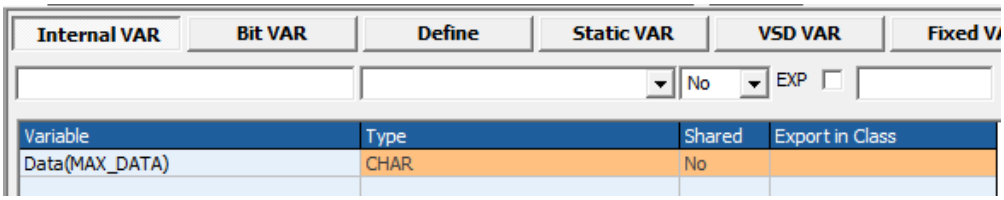
Nell' esempio riportato, vengono scritti e letti dei registri 16 bit all' interno della scheda.
L' array dei registri Modbus è definito col Nome **Data** e il numero massimo di registri è definito dalla DEFINE **MAX_DATA** (100 nell' esempio)
Pertanto :
Lettura/Scrittura da Modbus registri Nr.1 → Data(0)
Lettura/Scrittura da Modbus registri Nr.2 → Data(1)
ecc.
L' esempio legge il registro 2 - Data(1) e scrive nel registro 1 Data(0)

Oggetti utilizzati:

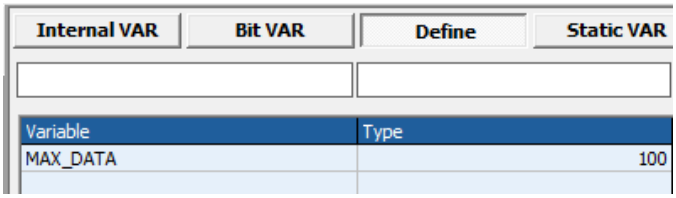
Modubus → CModbus → Modbus protocol TCP



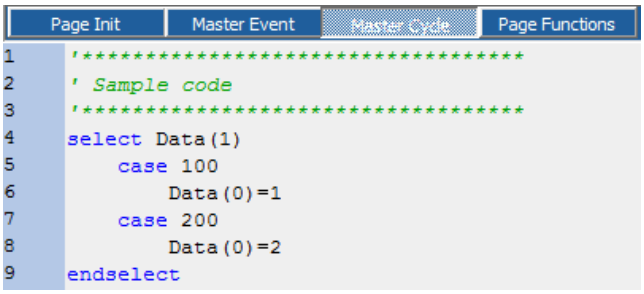
Dichiarare le variabili del progetto



Dichiarare le DEFINE del progetto



Inserire il seguente codice nella Master Ciclo – Main




```
'******************************
' Sample code
'******************************
select Data(1)
      case 100
            Data(0)=1
      case 200
            Data(0)=2
endselect
```

Scarica l' esempio

4 CLIENT TCP/IP

La Porta Ethernet può essere configurata come CLIENT TCP e connettrsi automaticamente a dei sistemi esterni per scambiarsi informazioni tramite protocollo TCP/IP. L' intera gestione del CLIENT è affidata ad un oggetto.

4.1 OGGETTO TCP_Client

Questo oggetto gestisce una comunicazione generica TCP in modo CLIENT e il protocollo RPC per lo scambio dati tra schede PROMAX.

Proprietà principali

IP address	Indirizzo IP remoto a cui connettersi - ***no in Run time***
Port	Porta alla quale connettersi - ***no in Run time***
Idle TimeOut	Time out disconnessione per inattività (in secondi) - ***no in Run time***
RPC TimeOut	Time out risposte protocollo RPC (in millisecondi) - ***no in Run time***
bytes_received	Numero byte nel buffer di ricezione ***– Read Only***
status_connected	Connessione avvenuta ***– Read Only***
status_closed	Connessione chiusa ***– Read Only***
status_abort	Connessione chiusa (da remoto, errore, ecc.) ***– Read Only***
status_overun	Perdita dati in ricezione ***– Read Only***

Metodi

Questi metodi consentono di gestire una comunicazione TCP dal lato CLIENT. Sono indipendenti dal protocollo utilizzato, ovvero i dati in ricezione/trasmissione non hanno un significato definito ma dipendono dal protocollo utilizzato a livello superiore.

function .connect*(wait_time *as long*) *as char
Richiesta di connessione a IP e PORT impostato dalle proprietà. La funzione attende per il tempo impostato (wait_time) che il dispositivo remoto risponda alla richiesta dopo di che sia ha un ritorno dalla funzione.
ATTENZIONE: la gestione della connessione è indipendente da questo tempo. Il parametro "wait_time" serve esclusivamente per uscire dalla funzione ma il sistema continua a riprovare più volte anche se è scaduto questo tempo. Per capire quando effettivamente il time-out di sistema è scaduto occorre testare la status: almeno un bit tra status_connected, status_closed o status_abort deve essere settato.
Evitare di rieseguire una "connect" quando il time-out di sistema non è ancora scaduto.

Parametri

Wait_time	Tempo di attesa connessione

Ritorna

>0	Connessione OK
-1	Errore di connessione
-2	Time out scaduto

function .close*() *as void
Chiusura connessione. Termina la connessione attiva liberando le risorse di Sistema.

function .send*(buf as **char, len as uint*) *as int
Invio dei dati alla connessione aperta. La funziona effettua l'invio dati e ritorna immediatamente. In caso di errori di rete il sistema effettua vari tentativi dopo di che la connessione viene chiusa.

Parametri

buf	Puntatore ai dati da inviare
len	Numero di Bytes da inviare

Ritorna

>=0	Numero di Bytes inviati
-1	Errore invio dati

function .recv*(buf as **char, len as uint*) *as int
Lettura dati dal buffer di ricezione. Leggendo la proprità **bytes_received** (solo lettura) è possibile sapere quanti dati (bytes) sono presenti nel buffer di ricezione. Il parametro len indica quanti dati scaricare dal buffer. Il valore di ritorno conterrà i dati effettivamente scaricati, sarà normalmente uguale a **len** a meno che non si voglia leggere più dati di quanti siano stati ricevuti. In questo caso ritornerà il valore di dati effettivamente letti.

Parametri

buf	Puntatore ai dati di destinazione
len	Numero Massimo di Bytes da leggere

Ritorna

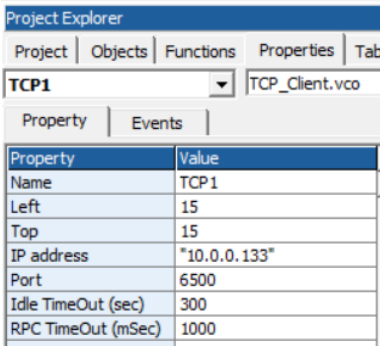
>=**0**	Numero di dati letti

4.2 Esempio TCP/IP generico

Esempio di collegamento con una scheda slave NG-35 ad indirizzo **10.0.0.133** porta **6500** per lo scambio di una stringa.
Viene scambiato una stringa “START”

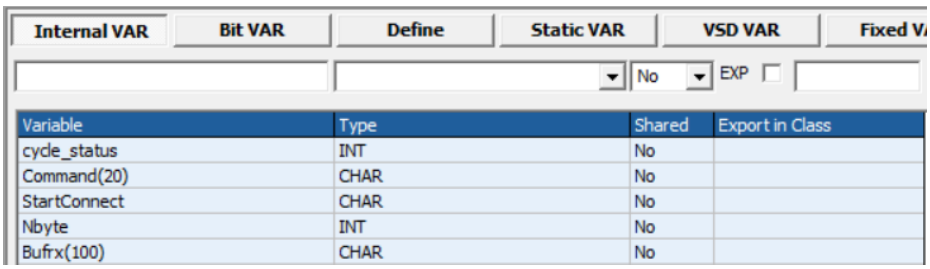
Oggetti utilizzati:

CommMaster → TCP_Client → TCP_Client



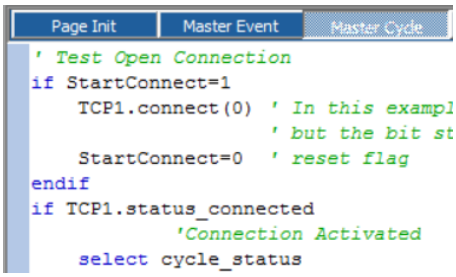
Property	Value
Name	TCP1
Left	15
Top	15
IP address	"10.0.0.133"
Port	6500
Idle TimeOut (sec)	300
RPC TimeOut (mSec)	1000

Dichiarare le variabili interne del progetto



Variable	Type	Shared	Export in Class
cycle_status	INT	No	
Command(20)	CHAR	No	
StartConnect	CHAR	No	
Nbyte	INT	No	
Bufrx(100)	CHAR	No	

Inserire il seguente codice nella Master Ciclo – Main




```
' Test Open Connection
if StartConnect=1
        TCP1.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
        StartConnect=0  ' reset flag
endif
if TCP1.status_connected
                        'Connection Activated
        select cycle_status
                Case 10                 ' Send String START
                        cycle_status=20
                        strcpy(command(),"START")
                        TCP1.send(command(),5)
                Case 20                 ' Wait response
                        nbyte=TCP1.recv(bufrx(),20)
                        If nbyte
                                ' Process Data
                                cycle_status=0
                        endif
        endselect
endif
if TCP1.status_closed || TCP1.status_abort
                        ' Connection closed
endif
```

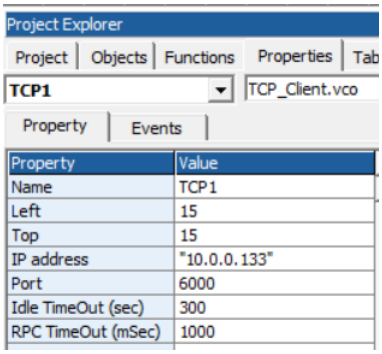
Scarica l' esempio

4.3 Esempio TCP/IP RPC

Esempio di collegamento con una scheda slave NG-35 IP **10.0.0.133** Porta **6000** per lo scambio di una tabella parametri tramite protcollo TCP/IP – RPC (Remote Procedure Call).
Nella scheda slave definire **AD_PARAMETER** nelle fixed all' indirizzo ZERO e assegnargli il puntatore della tabella parametri: **AD_PARAMETER =tab_param()** nell' init della TASK PLC. **Tab_param** è l' array per lo scambio dati.
Nella master definire **AD_PARAMETER** nelle fixed all' indirizzo 0 (lo stesso della slave) e definire la tabella parametri **tab_param** della stessa dimensione (nell'esempio 25 long)

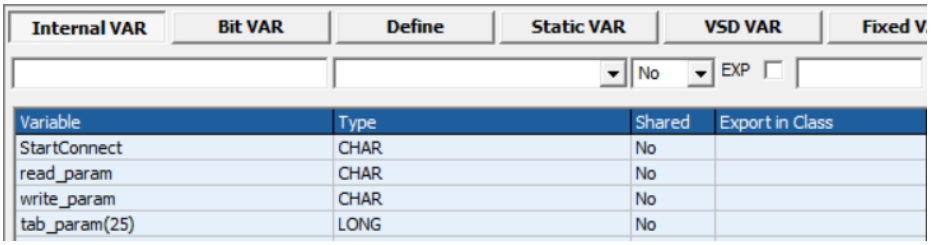
Oggetti utilizzati:

CommMaster → TCP_Client → TCP_Client



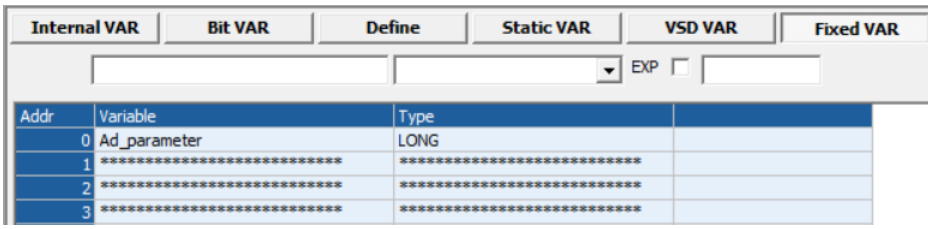
Property	Value
Name	TCP1
Left	15
Top	15
IP address	"10.0.0.133"
Port	6000
Idle TimeOut (sec)	300
RPC TimeOut (mSec)	1000

Dichiarare le variabili interne del progetto



Variable	Type	Shared	Export in Class
StartConnect	CHAR	No	
read_param	CHAR	No	
write_param	CHAR	No	
tab_param(25)	LONG	No	

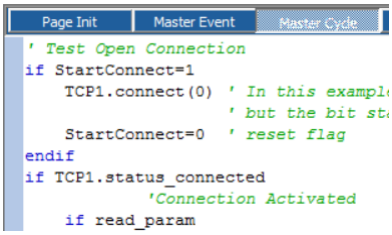
Dichiarare le Variabili Fixed del progetto



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

ATTENZIONE:
Questo indirizzo deve essere uguale a quello della slave (nell' esempio FIXED 0)

Inserire il seguente codice nella Master Ciclo – Main




```
' Test Open Connection
if StartConnect=1
        TCP1.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
        StartConnect=0  ' reset flag
endif
if TCP1.status_connected
                        'Connection Activated
        if read_param
                read_param=0
                TCP1.rpc_read(AD_PARAMETER(),4, AD_PARAMETER())     ' Pointer Read
                TCP1.rpc_read(AD_PARAMETER,100,tab_param())         ' Read 100 bytes - 25 long
        endif

        if write_param
                write_param=0
                TCP1.rpc_read(AD_PARAMETER(),4, AD_PARAMETER())     ' Pointer Read
                TCP1.rpc_write(AD_PARAMETER,100,tab_param())        ' Write 100 bytes - 25 long
        Endif
endif
if TCP1.status_closed || TCP1.status_abort
                        ' Connection closed
endif
```

Scarica l' esempio

5 Porta RS232/RS485

La scheda NGWARP mette a disposizione 1 porta RS232/RS485 completamente gestibile da protocolli CUSTOM. Oppure da MODBUS RTU MASTER SLAVE (tramite oggetto VTB)

5.1 SER_SETBAUD

Programma il BaudRate del secondo CANALE SERIALE SER2

Sintassi

SER_SETBAUD (Baud as long) as void

Parametri

Baud Valore del Baud Rate. Questo deve corrispondere ad un valore STANDARD:
1200-2400-4800-9600-19200-38400-57600-115200

5.2 SER_MODE

Programma la modalità del secondo CANALE SERIALE. Nel caso questa funzione non sia richiamata, di default la seriale di viene programmata con:
No parity
8 bit a carattere
1 bit di stop.

Sintassi

SER_MODE(par as char, nbit as char,nstop as char) as void

Parametri

par Parity (0=no parity, 1=odd parity, 2=even parity)
nbit Numero bit a carattere (7 o 8)
nstop Numero bit di stop (1 o 2)

5.3 SER_GETCHAR

Ritorna un carattere presente nel buffer della linea seriale. Il sistema operativo si occupa della gestione del buffer di ricezione. Questa funzione deve essere chiamata in polling dall' applicazione.
Il sistema operativo, gestisce comunque un BUFFER di ricezione ad INTERRUPT.

Sintassi

SER_GETCHAR () as int

Valore di ritorno

int ***-1*** Nessun carattere presente nel buffer
>=0 Codice (0-255) del carattere recuperato dal buffer

5.4 SER_PUTCHAR

Invia un carattere sulla linea seriale. Il sistema operativo si occupa della gestione del buffer di trasmissione.

Sintassi

SER_PUTCHAR (Car as int) as void

Parametri

Car Codice (0-255) del Carattere da inviare

5.5 SER_PUTS

Invia una stringa di caratteri sulla linea seriale. La stringa deve terminare con il carattere 0 (NULL).
Sintassi

SER_PUTS (str **as *char) as void**

Parametri

***str**	Puntatore alla stringa

5.6 SER_PRINTL

Stampa formattata di una variabile intera.

Sintassi

SER_PRINTL (format **as *char**, val **as long**) **as void**

Parametri

Format	Costante stringa che indica il formato da stampare
Val	Qualsiasi valore, espressione o variabile intera

Formati disponibili

######	Indica il numero di caratteri da stampare	23456
###.###	Stampa con il punto decimale nella posizione inserita	123.456
+####	Stampa sempre con segno	+1234
#0.##	Forza uno ZERO nel punto inserito	0.12
X##	Stampa in formato ESADECIMALE	F1A3
B##	Stampa in formato BINARIO	10110011

5.7 SER_PRINTF

Stampa formattata di una variabile float. E' come la precedente ma lavora con dei valori FLOAT.

Sintassi

SER_PRINTF (format **as *char**, val **as float**) **as void**

Parametri

Format	Costante stringa che indica il formato da stampare
Val	Qualsiasi valore, espressione o variabile di tipo float

5.8 SER_PUTBLK

Invia un blocco di caratteri con lunghezza specificata. Rispetto alla funzione ***ser_puts*** permette di inviare anche il codice 0, inoltre avvia la trasmissione in background gestendo anche il segnale RTS utilizzato per abilitare il buffer RS485.

ATTENZIONE

Questa funzione è adatta ad una trasmissione BINARIA dei caratteri ed è l'unica che consente la trasmissione con RS485.

Sintassi

SER_PUTBLK (Buffer **as *char**, Len **as int**) **as void**

Parametri

***Buffer**	Puntatore al buffer da trasmettere
Len	Lunghezza dei Byte da trasmettere

5.9 SER_PUTST

Ritorna lo stato della trasmissione in background avviata con ***ser_putblk.***

Sintassi

SER_PUTST () as int

Valore di ritorno

int	***-1***	Errore di trasmissione
	>=0	Numero di caratteri ancora da inviare

5.10 Esempio

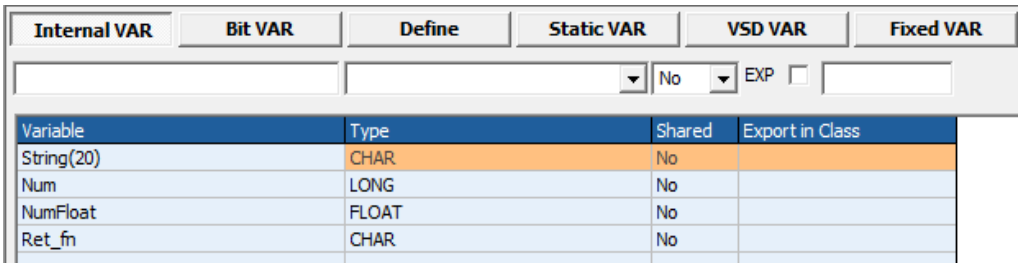
Nell' esempio riportato, viene chiamata la funzione Read_Data() in polling dal task main.
Seriale SER2 programmata con i seguenti parametri:

Baud rate → ***115,200***
Nr. bit dati → ***8***
Nr. bit Stop → ***1***
Parità → ***NO***

Questa controlla se ricevuto un carattere ed effettua un risposta:

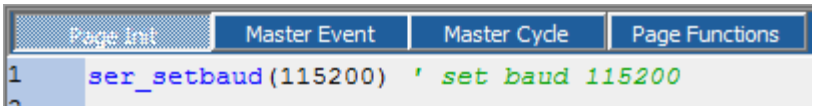
Carattere ricevuto=1 → ***Eco del carattere stesso (1) con ser_putchar***
Carattere ricevuto=2 → ***Invio testo "Test String" con Ser_puts***
Carattere ricevuto=3 → ***Print formattato di variabile Num (Long numero caratteri ricevuti)***
Carattere ricevuto=4 → ***Print formattato di variabile NumFloat (Float numero random)***
Carattere ricevuto=5 → ***Invio dati in Binario Nr. 789488 con Ser_putblk***
Carattere ricevuto=6 → ***Test stato Ser_putblk risponde:***
255 errore di invio dati
Nr caratteri rimanenti nel Buffer di trasmissione
Carattere ricevuto=Altri → Risponde 254 errore carattere non riconosciuto

Variabili Utilizzate



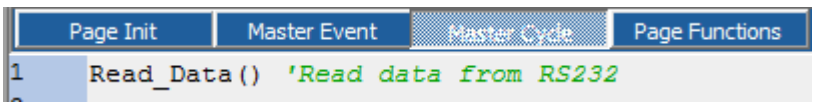
Variable	Type	Shared	Export in Class
String(20)	CHAR	No	
Num	LONG	No	
NumFloat	FLOAT	No	
Ret_fn	CHAR	No	

Codice in Init Main



ser_setbaud(115200) **' set baud 115200**

Codice in Master Ciclo Main



Read_Data() **'Read data from RS232**

Codice in Funzioni di Pagina Main

Page Init | Master Event | Master Cycle | Page Functions

```
'****************************
'Read Data From RS232
'****************************
function Read_Data() as void
Ret_fn=Ser_getchar()     ' Read one char from RS232 buffer
if Ret_fn=-1 ' none
    return ' return
endif
```

```
'****************************
'Read Data From RS232
'****************************
function Read_Data() as void

Ret_fn=Ser_getchar()      ' Read one char from RS232 buffer
if Ret_fn=-1 ' none
        return ' return
endif
inc Num ' increases the received chars
NumFloat=Num*2.13 'random number
'process data received
select Ret_fn
        case 1 ' ------ echo char with send_putchar
                Ser_putchar(Ret_fn)        ' send reply echo char
        case 2 ' ------ send string with ser_puts
                strcpy(String(),"Test String") ' Copy in array string text
                ser_puts(String()) ' put data
        case 3 ' ------ print a long formatted with ser_printl
                ser_printl("###.##",Num) ' print ex: 123.45 format
        case 4 ' ------ print a float formatted with ser_printf
                ser_printf("####.###",NumFloat) ' print NumFloat
        case 5 ' ------ put a block with ser_putblk
                'Send a number 789488
                String(0)=0xF0 'LSB
                String(0)=0xOB
                String(0)=0x0C
                String(0)=0 'MSB
                Ser_putblk(String(),4) ' Data len 4 byte
        case 6 ' ------ test if ser_putblk is busy
                Ret_fn= Ser_putst() ' check if function ser_putblk is busy
                if Ret_fn=-1
                        Ser_putchar(255)          ' send error
                else
                        Ser_putchar(Ret_fn)       ' send number of chars
                endif
        case else
                Ser_putchar(254)          ' send error no char
endselect

endfunction
```

Scarica l' Esempio

6 Modbus RTU

La Porta SER2 può essere configurata anche con il protocollo MODBUS RTU.
Il MODBUS RTU è interamente gestito da un Oggetto per VTB ed è disponibile nelle due configurazioni principali Master o Slave

6.1 OGGETTO Modbus RTU Slave

Tramite quest' oggetto è possibile gestire in modo TRASPARENTE, il protocollo ModBus RTU slave

Proprietà principali

Nodo Nodo slave
BaudRate baud rate inserire valori concordi
PtData() Registri Unsigned Integer messi a disposizione del master. Indirizzo di partenza 0
Max Len Data Dimensione del buffer PtData
TimeOut Tempo in millisecondi interrogazione del master
Questo valore deve essere di norma inferiore al timeout impostato sul master

Metodi

Nessuno

Vengono gestite le seguenti richieste MODBUS RTU:

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

Eventi

Nessuno

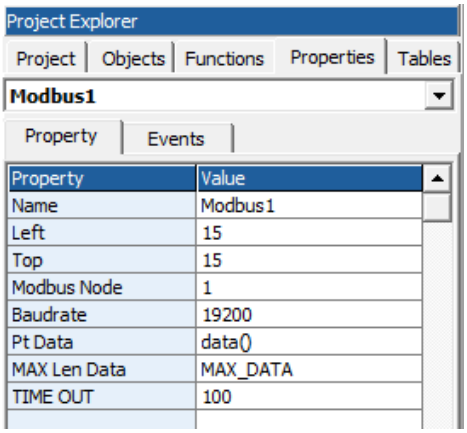
6.2 Esempio ModBus slave

Nell' esempio riportato, vengono scritti e letti dei registri 16 bit all' interno della scheda.
L' array dei registri Modbus è definito col Nome **Data** e il numero massimo di registri è definito dalla DEFINE **MAX_DATA** (100 nell' esempio)
Pertanto :

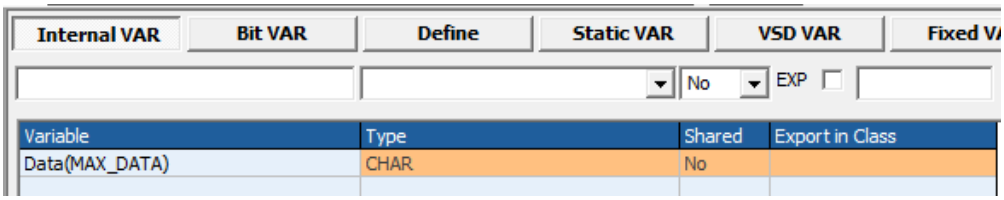
Lettura/Scrittura da Modbus registri Nr.1 → Data(0)
Lettura/Scrittura da Modbus registri Nr.2 → Data(1)
ecc.
L' esempio legge il registro 2 - Data(1) e scrive nel registro 1 Data(0)

Oggetti utilizzati:

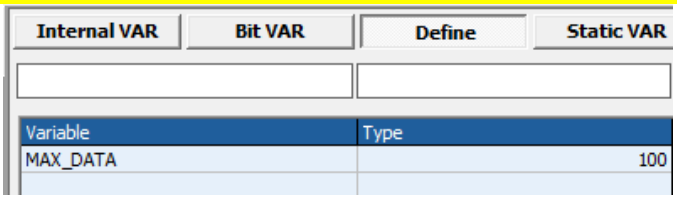
Modbus → Cmodbus → ModBus Protocol



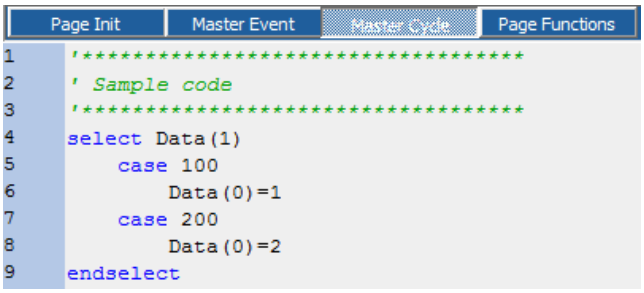
Dichiarare le variabili del progetto



Dichiarare le DEFINE del progetto



Inserire il seguente codice nella Master Ciclo – Main




```
'*****************************
' Sample code
'*****************************
select Data(1)
      case 100
            Data(0)=1
      case 200
            Data(0)=2
endselect
```

Scarica l' Esempio

6.3 OGGETTO Modbus RTU Master

Tramite quest' oggetto è possibile gestire in modo TRASPARENTE, il protocollo ModBus RTU Master

Proprietà principali

BaudRate	baud rate inserire valori concordi con periferiche esterne slave
TimeOut	Tempo in millisecondi del TIME OUT su risposta slave. Questo valore deve essere di norma maggiore al timeout impostato sugli slaves
Parita	0 nessuna - 1 odd - 2 even
N. bit car	Numero bit per carattere
N. bit stop	Numero bit di stop

Metodi

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

Preset multiple registers func 16 ModBus RTU

Parametri

nodo	Nodo slave modbus
addr	Indirizzo registro di partenza scrittura su slave (fare riferimento allo slave)
Value	Puntatore ad unsigned integer contenente i valori da scrivere
n	Numero di registri da scrivere

Ritorna

0	Scrittura OK
1	Risposta errata dallo slave
2	Time Out
3	lunghezza dati > di 127

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

Read multiple registers func 3 ModBus RTU

Parametri

nodo	Nodo slave modbus
addr	Indirizzo registro di partenza lettura su slave (fare riferimento allo slave)
Value	Puntatore ad unsigned integer di deposito dati letti
n	Numero di registri da leggere

Ritorna

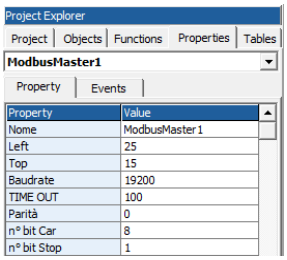
0	Scrittura OK
1	Risposta errata dallo slave
2	Time Out
3	lunghezza dati > di 127
4	errore checksum

6.4 Esempio ModBus Master

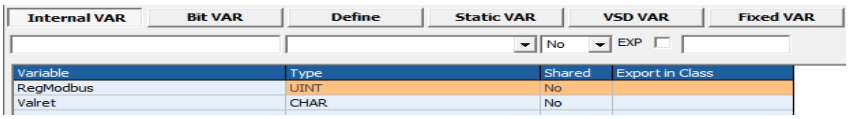
Nell' esempio riportato, vengono scritti e letti dei registri 16 bit di uno slave al nodo

Oggetti utilizzati:

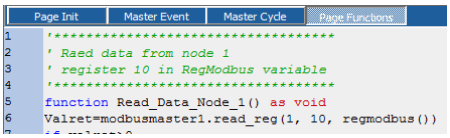
Modbus → CmodbusMaster → ModBus Master Protocol



Dichiarare le variabili del progetto



Codice in Funzioni di Pagina Main




```
'**********************************
' Raed data from node 1
' register 10 in RegModbus variable
'**********************************
function Read_Data_Node_1() as void
Valret=modbusmaster1.read_regn(1, 10, regmodbus())
if Valret>0
        ' read error
endif
endfunction
'**********************************
' Write data to node 1
' register 10 RegModbus variable
'**********************************
function Write_Data_Node_1() as void
RegModbus=100
Valret=modbusmaster1.write_regn(1, 10, RegModbus)
if valret>0
        ' write error
endif
endfunction
```

Scarica l' Esempio

7 Lettura Ingressi Analogici

La CPU NGWARP integra 8 ingressi analogici disponibili da funzioni VTB.
Attualmente gli ingressi sono a 10 bit (valore da 0 a 1023), ma nelle versioni successive saranno a 12 bit (valore da 0 a 4095)

7.1 Lettura Ingressi

Sintassi

NG_ADC(Channel **as Char**) **as uint**

Parametri

Channel Numero del canale (da 0 a 7)

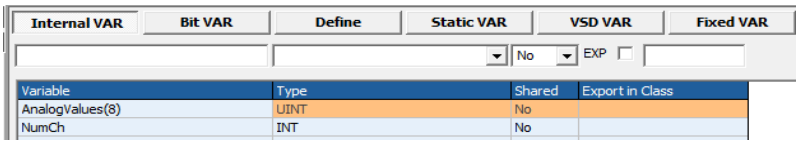
Valore di ritorno

Ritorna il valore analogico letto:
Da 0 a 1023 se convertitore 10 BIT- Da 0 a 4095 se convertitore 12 BIT
Dove 0 corrisponde al livello di tensione 0 e valore MAX (1023 o 4096) corrisponde al valore massimo di tensione configurato per l' ingresso

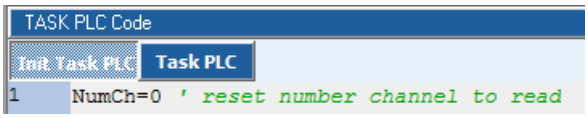
7.2 Esempio Lettura canali analogici

Nell' esempio riportato, vengono letti i canali analogici da 0 a 7 e scritti nella array AnalogValues
I dati vengono letti nel TaskPlc

Dichiarare le variabili del progetto

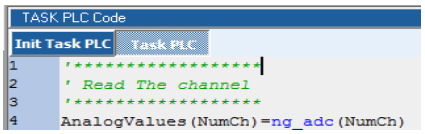


Codice su Init Task Plc



NumCh=0 **' reset number channel to read**

Codice in Task Plc




```
'******************
' Read The channel
'******************
AnalogValues(NumCh)=ng_adc(NumCh)
inc NumCh 'increase channel number
if NumCh=8 ' limit
        NumCh=0
endif
```

Scarica l' Esempio

8 Gestione CanOpen

Nella scheda NGWARP, sono presenti due Line CanOpen. Una di queste può essere configurata SLAVE e la sua gestione è affidata la sistema operativo.
La linea MASTER, può tramite configuratore, gestire PDO sincroni al sistema (per questa sezione si rimanda al manuale del configuratore CanOpen (link Cap. 14)
Tuttavia è possibile utilizzare funzioni manuali per configurare o dialogare con i vari nodo CanOpen

8.1 PXCO_SDODL

Questa funzione permette di inviare dati ad un nodo della rete utilizzando il protocollo SDO. E' supportato solo il protocollo SDO EXPEDITED consentendo quindi di inviare fino a 4byte di lunghezza dati.

Sintassi
PXCO_SDODL(node **as char**,index **as uint**,subidx **as uchar**,len **as long**,dati **as *char**) **as char**

Parametri
Node Node ID dello SLAVE
Index, subindex Indirizzo del dato da trasferire (Object-Dictionary)
Len Numero di Byte da trasferire
***Dati** Puntatore al buffer dati da trasferire (questo deve essere sempre un variabile)

Valore di ritorno
char **0** Nessun errore
<>0 Errore di comunicazione:
2 Il nodo ha risposto con SDO ABORT CODE, chiamando la funzione ***read_sdoac*** nelle variabili di sistema _SYSTEM_SDOAC0 e _SYSTEM_SDOAC0 sarà disponibile il relativo codice di errore.

ATTENZIONE
Non usare questa funzione nel TASK PLC

8.2 PXCO_SDOUL

Questa funzione permette di leggere dati da un nodo della rete utilizzando il protocollo SDO. E' supportato solo il protocollo SDO EXPEDITED consentendo quindi di leggere fino a 4byte di lunghezza dati.

Sintassi
PXCO_SDOUL(node **as char**, index **as uint**,subidx **as uchar**, dati **as *char**) **as char**

Parametri
Node Node ID dello SLAVE
Index,subindex Indirizzo del dato da richiedere (Object-Dictionary)
***Dati** Puntatore al buffer dati

Valore di ritorno
char **0** Nessun errore
<>0 Errore di comunicazione
2 Il nodo ha risposto con SDO ABORT CODE, chiamando la funzione read_sdoac nelle variabili di sistema _SYSTEM_SDOAC0 e _SYSTEM_SDOAC0 sarà

ATTENZIONE
Non usare questa funzione nel TASK PLC

8.3 READ_SDOAC

Lettura dello SDO ABORT CODE inviato da un nodo della rete a seguito di una richiesta tramite le funzioni PXCO_SDODL e PXCO_SDOUL. Il codice letto viene scritto nelle variabili di sistema **_SYSTEM_SDOAC0** e **_SYSTEM_SDOAC1**.
Per i codici di errore fare riferimento alle specifiche DS301 del CAN OPEN.

Sintassi

```
READ_SDOAC() as void
```

8.4 PXCO_SEND

Invio di un pacchetto CAN a basso livello. Questa funzione consente di inviare alla rete un frame CAN indicando COB-ID e DATI. Con questa funzione sarà possibile quindi inviare ad esempio PDO in modo manuale, pacchetti HEART-BEAT, ecc.
Occorre precisare che la gestione dei PDO viene eseguita in modo AUTOMATICO tramite il CONFIGURATORE CANOPEN.

Sintassi

```
PXCO_SEND(id as int,Len as char,Dati as *char) as char
```

Parametri

Id	Valore del COB_ID
Len	Numero di dati da trasferire
***Dati**	Puntatore al buffer dati da trasferire

Valore di ritorno

char	**0**	Nessun errore
	<>0	Errore di comunicazione

8.5 PXCO_NMT

Invio di un pacchetto NMT del CAN OPEN. I pacchetti NMT consentono di settare lo stato dei nodi nella rete. Occorre tenere presente che tutti i nodo presenti in configurazione (configuratore canopen) ed che l'hanno eseguita senza errori sono messi in stato START in modo automatico.

Sintassi

```
PXCO_NMT(state as char, node as char) as char
```

Parametri

state	Stato da inviare: 1 = START NODE 2 = STOP NODE 128 = PRE-OPERATIONAL 129 = RESET NODE 130 = RESET COMUNICATION
node	Numero del nodo

Valore di ritorno

char	**0**	Nessun errore
	<>0	Errore di comunicazione

ATTENZIONE
Non usare questa funzione nel TASK PLC

8.6 READ_EMCY

Legge l'ultimo pacchetto EMERGENCY OBJECT inviato da un eventuale nodo della linea CAN OPEN. Il codice di emergenza viene inserito nella variabile vettore di sistema _SYSTEM_EMCY(8) che conterrà gli 8 byte relativi al pacchetto EMERGENCY OBJECT previsto dalle specifiche DS301 del CAN OPEN. Si consiglia di leggere questa funzione in modo ciclico. I codice di allarme sono dipendenti dal tipo di dispositivo collegato, occorrerà quindi riferirsi al manuale di tale dispositivo.

Sintassi

READ_EMCY() **as char**

Valore di ritorno

char **0** Nessun errore
<>0 Nodo che ha generato l'emergency object.

_SYSTEM_EMCY							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Emergency Error Code		**Error Register**	**Manufacturer specific Error Code**				

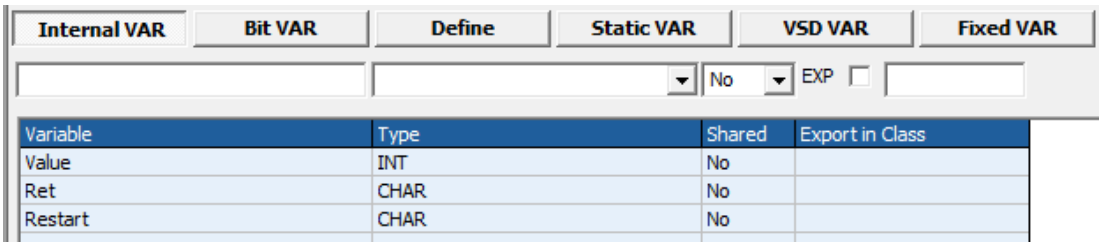
ATTENZIONE

Il sistema non bufferizza più errori, quindi nel caso in cui si verifichino più EMERGENCY OBJECT in contemporanea viene letto solo l' ultimo.
Un EMERGENCY OBJECT non significa che effettivamente ci sia un nodo in emergenza. Questo poiché le specifiche DS301 prevedono che tale pacchetto sia inviato anche al ripristino di una fase di emergenza. Alcuni dispositivi possono inviare all'accensione questo pacchetto

8.7 Esempio Utilizzo delle funzioni CanOpen

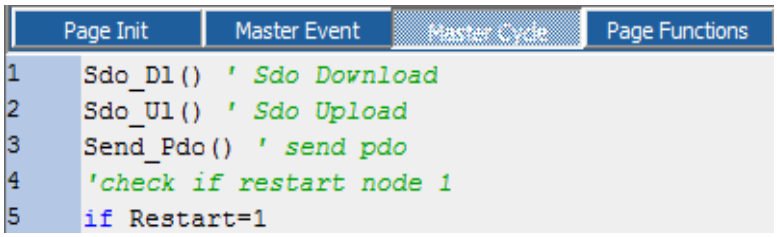
Nell' esempio riportato, vengono utilizzate le funzioni CanOpen. Ovviamente questo ci dà semplicemente un indicazione del loro utilizzo.

Dichiarare le variabili del progetto



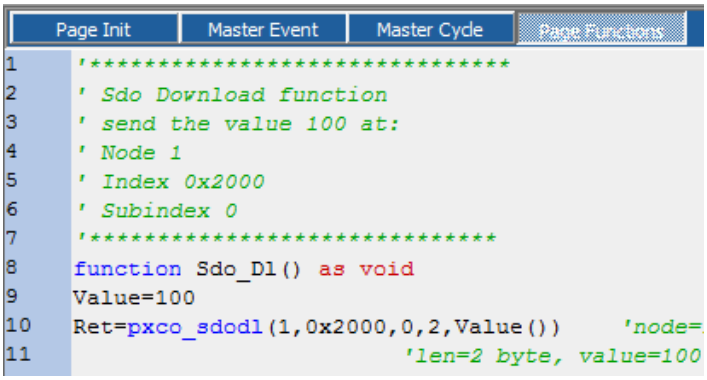
Variable	Type	Shared	Export in Class
Value	INT	No	
Ret	CHAR	No	
Restart	CHAR	No	

Inserire il seguente codice nella Master Ciclo – Main




```
Sdo_Dl() ' Sdo Download
Sdo_Ul() ' Sdo Upload
Send_Pdo() ' send  pdo
'check if restart node 1
if Restart=1
        Restart=0 ' reset flag restart
        Ret=pxco_nmt(1,1) ' Start Node
        if Ret<>0           'test error
                '...
        endif
endif

'polling emergency object
Ret=Read_emcy()
if Ret<>0
        ' in Ret node error
        ' in _SYSTEM_EMCY code error
endif
```

Codice in Funzioni di Pagina Main




```
'*******************************
' Sdo Download function
' send the value 100 at:
' Node 1
' Index 0x2000
' Subindex 0
'*******************************
function Sdo_Dl() as void
Value=100
Ret=pxco_sdodl(1,0x2000,0,2,Value())'node=1, index=0x2000, subidx=0,
                                                    'len=2 byte, value=100
if Ret<>0           'test error
        if Ret=2
                read_sdoac()'Read SDO ABORT CODE
                'in _SYSTEM_SDOAC0 code error
                'in _SYSTEM_SDOAC1 code error
        endif
        '...
endif
endfunction

'*******************************
' Sdo Upload function
' read the value at:
' Node 1
' Index 0x2000
' Subindex 0
'*******************************
function Sdo_Ul() as void
Ret=pxco_sdoul(1,0x2000,0,Value())          'node=1, index=0x2000, subidx=0,
                                                    'read in value
if Ret<>0           'test error
        if Ret=2
                read_sdoac()'Read SDO ABORT CODE
                'in _SYSTEM_SDOAC0 code error
                'in _SYSTEM_SDOAC1 code error
        endif
        '...
endif
endfunction
```

```
'*******************************
' Send PDO
' COB - ID = 0x201
' 2 Bytes
' SVariable in Value
'*******************************
function Send_Pdo() as void
Value=100
Ret=pxco_send(0x201,2,Value())    'cob-id=0x201) 2 bytes
if Ret<>0            'test error
                     '...
endif
endfunction
```

Scarica l' Esempio

8.8 Esempio Utilizzo Assi Interpolati CanOpen

Nell' esempio riportato, vengono gestiti TRE assi CanOpen in modo interpolazione lineare.

ATTENZIONE:
Tutte le velocità vengono inserite in mm/min se impostato correttamente i parametri:
RapX,RapY,RapZ
Tutte le quote assi vengono inserite in micron (0.001 mm) se impostato correttamente i parametri:
RapX,RapY,RapZ

Oggetti utilizzati:

Motor Control → CobjInterpola → Interpolatore

Project Explorer

Project | Objects | Functions | Properties | Tables

Interp

Property | Events

Property	Value
Nome	Interp
Left	15
Top	10
N.assi	3
N.tratti	16
Vper	1024
Div. Vper	1024
Abilita arcto	1

Motor Control → CstdCanOpen → Ds402 x 3

Project Explorer

Project | Objects | Functions

AxisX

Property | Events

Property	Value
Name	AxisX
Left	10
Top	85
Node	1
Mode	0
Speed	0
Position	0
Abs	True
State	False
home_delay	1000

Project Explorer

Project | Objects | Functions

AxisY

Property | Events

Property	Value
Name	AxisY
Left	55
Top	85
Node	2
Mode	0
Speed	0
Position	0
Abs	True
State	False
home_delay	1000

Project Explorer

Project | Objects | Functions

AxisZ

Property | Events

Property	Value
Name	AxisZ
Left	100
Top	85
Node	3
Mode	0
Speed	0
Position	0
Abs	True
State	False
home_delay	1000

Vengono gestite le seguenti funzioni:

Wait_Move – Stato movimento assi
Parametri *Nessuno*
Ritorna *1 Assi in movimento*
0 Assi fermi

Move_Axes – Interpolazione lineare
Parametri *Vel → Velocità assi in millimetri/min*
Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)
Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)
Px,Py,Pz → Quote Assi in 0.001 mm
Ritorna *0 Movimento inserito nel buffer*
1 Buffer pieno (occorre ripetere Move_Axes fino a che il buffer è libero)

Acc_Axes – Set Accelerazione di Interpolazione
Parametri *Value → Valore Accelerazione di interpolazione in count per TAU*
Ritorna *Nessuno*

Stop_Axes – Stop Movimento
Parametri *Nessuno*
Ritorna *Nessuno*

Enable_Axis_X_Y_Z – Abilita controllo Asse preset a zero
Parametri *Nessuno*
Ritorna *Nessuno*

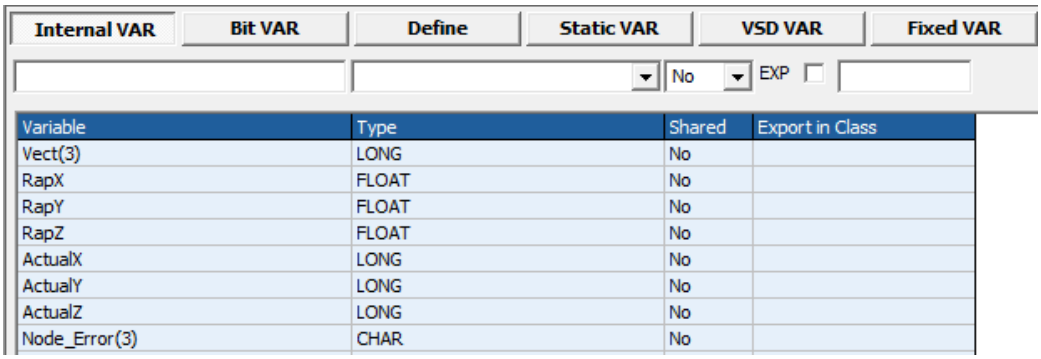
Disable_Axis_X_Y_Z – Disabilita controllo Asse
Parametri *Nessuno*
Ritorna *Nessuno*

cancfgerr - Errori Custom CanOpen questa funzione viene chiamata all' inizializzazione dei Nodi CanOpen (presenti in configurazione) quando uno di questi risponde errore
Parametri *Node → Numero nodo che ha generato l' errore*
Err → Codice di errore del nodo
Ritorna *Nessuno*

Close_cancfgerr - Errori Custom CanOpen questa funzione viene chiamata alla fine della scansione di tutti i nodi presenti in configurazione
Parametri *Nessuno*
Ritorna *Nessuno*

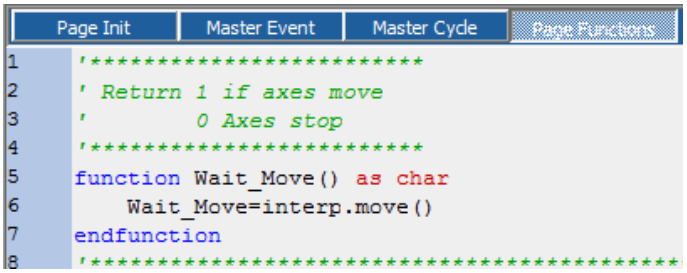
Open_cancfgerr - Errori Custom CanOpen questa funzione viene chiamata quando inizia la fase di inizializzazione di un anodo CanOpen
Parametri *Nodes → Numero di nodi presenti in configurazione*
Ritorna *Nessuno*

Dichiarare le variabili del progetto



Variable	Type	Shared	Export in Class
Vect(3)	LONG	No	
RapX	FLOAT	No	
RapY	FLOAT	No	
RapZ	FLOAT	No	
ActualX	LONG	No	
ActualY	LONG	No	
ActualZ	LONG	No	
Node_Error(3)	CHAR	No	

Codice in Funzioni di Pagina Main

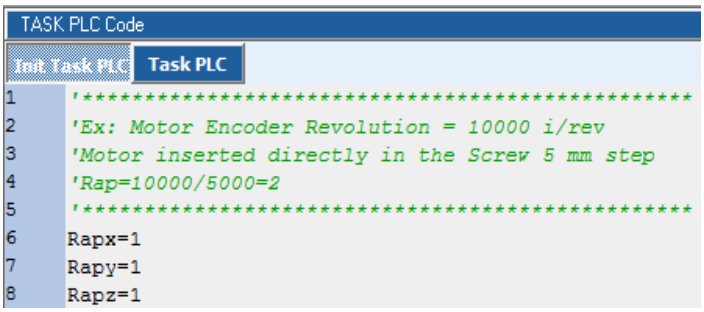



```
'*************************
' Return 1 if axes move
'      0 Axes stop
'*************************
function Wait_Move() as char
          Wait_Move=interp.move()
endfunction

'*********************************************************
' Move Axes
' Vel= interp vel Axes in mm/min
' Flg if 1 move without buffer
'      0 move in buffer mode
' Px,Py,Pz Axes value in 0.001 mm
'Return 1 if movement is inserted in the buffer
'     0 The movement is not inserted in the buffer
'       in this case, is necessary reload the movement
'*********************************************************
function Move_Axes(Vel as long, Flg as char, Px as long, Py as long,Pz as long) as char
          Vel=Vel*TAU/60 ' Transform in mm/min
          Vect(0)=Px
          Vect(1)=Py
          Vect(2)=Pz
          Move_Axes=interp.moveto(Vel, Flg, Vect())
endfunction

'*********************************************************
' Set ACC
' Value Acc value in count
```

```
'*************************************************************
function Acc_Axes(Value as long) as void
        interp.acc=Value
endfunction
'*************************************************************
' Stop Axes
'*************************************************************
function Stop_Axes() as void
        interp.stop()
endfunction
'*******************
' Axis X enable
'*******************
function Enable_X() as void
AxisX.modo=0 ' remove interpolation mode
AxisX.start=0 ' stop PDO Qx
'Preset Axis X 0, not change y,z
Vect(0)=0
Vect(1)=interp.pc(1)
Vect(2)=interp.pc(2)
interp.preset(Vect())
AxisX.home=0 'preset driver
'enable axis
AxisX.enable=1
AxisX.start=1 ' start PDO Qx
AxisX.modo=2 ' set interpolation mode
endfunction
'*******************
' Axis X Disable
'*******************
function Disable_X() as void
AxisX.enable=0
endfunction

'*******************
' Axis Y enable
'*******************
function Enable_Y() as void
AxisY.modo=0 ' remove interpolation mode
AxisY.start=0 ' stop PDO Qx
'Preset Axis Y 0, not change x,z
Vect(0)=interp.pc(0)
Vect(1)=0
Vect(2)=interp.pc(2)
interp.preset(Vect())
AxisY.home=0 'preset driver
'enable axis
AxisY.enable=1
AxisY.start=1 ' start PDO Qx
AxisY.modo=2 ' set interpolation mode
endfunction

'*******************
' Axis Y Disable
'*******************
```

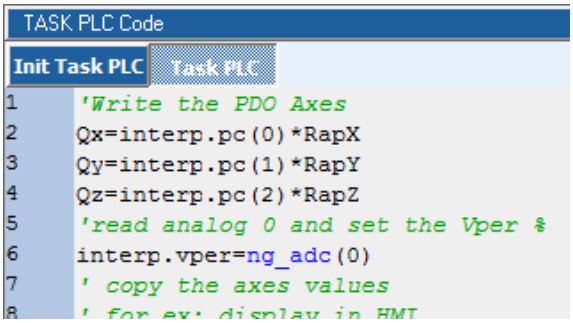
```
function Disable_Y() as void
AxisY.enable=0
endfunction

'*******************
' Axis Z enable
'*******************
function Enable_Z() as void
AxisZ.modo=0 ' remove interpolation mode
AxisZ.start=0 ' stop PDO Qx
'Preset Axis Z 0, not change x,y
Vect(0)=interp.pc(0)
Vect(1)=interp.pc(1)
Vect(2)=0
interp.preset(Vect())
AxisZ.home=0 'preset driver
'enable axis
AxisZ.enable=1
AxisZ.start=1       ' start PDO Qx
AxisZ.modo=2        ' set interpolation mode
endfunction

'*******************
' Axis Z Disable
'*******************
function Disable_Z() as void
AxisZ.enable=0
endfunction

'************************
' Error check
' CanOpen node
'************************
function cancfgerr(node as int,err as uchar) as void
Node_Error(node)=err ' copy the code error
endfunction
'************************
' Close init CanOpen
'************************
function close_cancfgerr() as void
endfunction
'************************
' Custom error init
' CanOpen node
'************************
function open_cancfgerr(nodes as int) as void
' Reset nodes  status error
Node_Error(0)=0
Node_Error(1)=0
Node_Error(2)=0
endfunction
```

Codice in Init Task PLC




```
'************************************************
'Ex: Motor Encoder Revolution = 10000 i/rev
'Motor inserted directly in the Screw 5 mm step
'Rap=10000/5000=2
'************************************************
Rapx=1
Rapy=1
Rapz=1
```

Codice in Task PLC




```
'Write the PDO Axes
Qx=interp.pc(0)*RapX
Qy=interp.pc(1)*RapY
Qz=interp.pc(2)*RapZ
'read analog 0 and set the Vper %
interp.vper=ng_adc(0)
' copy the axes values
' for ex: display in HMI
' value in 0.001 mm
ActualX=interp.pc(0)
ActualY=interp.pc(1)
ActualZ=interp.pc(2)
```

Scarica l' Esempio

8.9 Esempio Utilizzo Asse Posizionato CanOpen

Nell' esempio riportato, viene gestito un asse CanOpen posizionato utilizzando un oggetto VTB
Vedere il manuale relativo agli oggetti di VTB per ulteriori spiegazioni

ATTENZIONE:

Tutte le velocità vengono inserite in mm/min se impostato correttamente i parametri:

MSOF e DSOF

Tutte le quote assi vengono inserite in micron (0.001 mm) se impostato correttamente i parametri:

MSOF e DSOF

Oggetti utilizzati:

Motor Control Plus → CobjPos → Posizionatore

Project Explorer

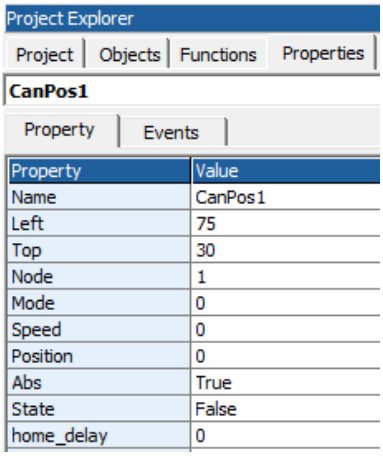
Project | Objects | Functions | Properties

Pos1

Property | Events

Property	Value
Vper	1024
Div. Vper	1024
AccQstop	10
Acc	5
RzeroMode	1
RzeroOffset	0
RzeroPreset	0
RzeroVel	10
RzeroVelf	5
RzeroAcc	10
Msof	10000
Dsof	5000
LimitN	-99999999
LimitP	99999999
Gioco	0
Vgioco	1
MsofV	1
DsofV	1
RZERO ENABLE	True
AXIS TYPE	1
VTB AXIS OBJECT	CanPos1
PDO NAME	qx
STEP CHANNEL	0
STEP NODE	1

Motor Control → CstdCanOpen → Ds402



Property	Value
Name	CanPos1
Left	75
Top	30
Node	1
Mode	0
Speed	0
Position	0
Abs	True
State	False
home_delay	0

Vengono gestite le seguenti funzioni:

Wait_Move – Stato movimento asse

Parametri ***Nessuno***

Ritorna ***1 Asse in movimento***

0 Asse fermi

Move_Axis – Interpolazione lineare

Parametri ***Vel → Velocità assi in millimetri/min***

Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)

Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)

Px → Quota Asse in 0.001 mm

Ritorna ***0 Movimento inserito nel buffer***

1 Buffer pieno (occorre ripetere Move_Axes fino a che il buffer è libero)

Preset – Preset quota Asse

Parametri ***Px → Quota Asse in 0.001 mm per preset***

Ritorna ***Nessuno***

Acc_Axis – Set Accelerazione/decelrazione di posizionamento

Parametri ***Value → Valore Accelerazione di posizionamento in count per TAU***

Ritorna ***Nessuno***

Stop – Stop Movimento

Parametri ***Nessuno***

Ritorna ***Nessuno***

Enable – Abilita controllo Asse

Parametri ***Nessuno***

Ritorna ***Nessuno***

Disable – Disabilita controllo Asse

Parametri ***Nessuno***

Ritorna ***Nessuno***

StartHome – Start ricerca homing Vel in pos1.rzerovel e pos1.rzerovelf

Parametri *Nessuno*
Ritorna *Nessuno*

CheckHome – Check stato homing in corso
Parametri *Nessuno*
Ritorna *1 homing terminato*

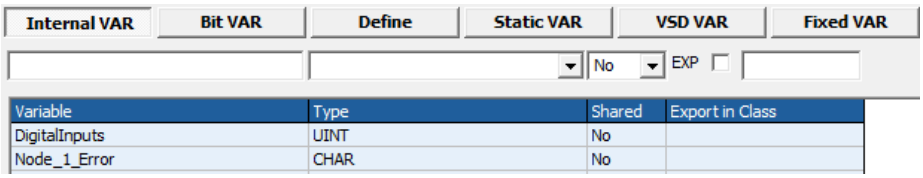
StopHome – Stop homing in corso
Parametri *Nessuno*
Ritorna *Nessuno*

cancfgerr - Errori Custom CanOpen questa funzione viene chiamata all' inizializzazione dei Nodi CanOpen (presenti in configurazione) quando uno di questi risponde errore
Parametri *Node → Numero nodo che ha generato l' errore*
Err → Cdice di errore del nodo
Ritorna *Nessuno*

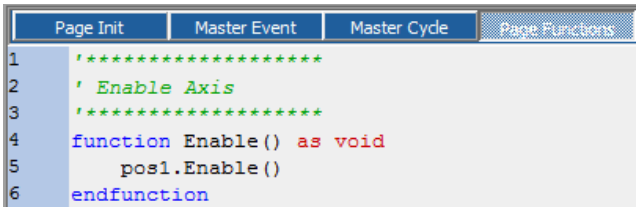
Close_cancfgerr - Errori Custom CanOpen questa funzione viene chiamata alla fine della scansione di tutti i nodi presenti in configurazione
Parametri *Nessuno*
Ritorna *Nessuno*

Open_cancfgerr - Errori Custom CanOpen questa funzione viene chiamata quando inizia la fase di inizializzazione di un anodo CanOpen
Parametri *Nodes → Numero di nodi presenti in configurazione*
Ritorna *Nessuno*

Dichiarare le variabili del progetto



Codice in Funzioni di Pagina Main




```
'********************
' Enable Axis
'********************
function Enable() as void
        pos1.Enable()
endfunction
'********************
' Disable Axis
'********************
function Disable() as void
        pos1.Disable()
endfunction
'********************
' Preset Axis
'********************
function Preset(Val as long) as void
```

```
        pos1.Preset(Val)
endfunction
'*************************
' Return 1 if axis move
'       0 Axis stop
'*************************
function Wait_Move() as char
        Wait_Move=pos1.move()
endfunction
'********************
' Axis Stop Move
'********************
function Stop() as void
        pos1.Stop()
endfunction
'*****************************
' Start Homing
' Homing input see in task plc
'*****************************
function StartHome() as void
        pos1.StartHome()
endfunction
'*****************************
' Check if homing finished
' Return 1 if finished
'*****************************
function CheckHome() as char
        CheckHome=pos1.status_home
endfunction
'*****************************
' Stop home function
'*****************************
function StopHome() as void
        pos1.StopHome()
endfunction
'*************************************************************
' Move Axis
' Vel=  vel Axis in mm/min
' Flg if 1 move without buffer
'       0 move in buffer mode
' Px  Axis value in 0.001 mm
'Return 1 if movement is inserted in the buffer
'      0 The movement is not inserted in the buffer
'       in this case, is necessary reload the movement
'*************************************************************
function Move_Axis(Vel as long, Flg as char, Px as long) as char
        Vel=Vel*TAU/60 ' Transform in mm/min
        Move_Axis=pos1.moveto(Vel, Flg, Px)
endfunction
'*************************************************************
' Set ACC
' Value Acc value in count
'*************************************************************
function Acc_Axis(Value as long) as void
        pos1.acc=Value
```

```
endfunction
'**********************
' Error check
' CanOpen node
'**********************
function cancfgerr(node as int,err as uchar) as void
Node_1_Error=err ' copy the code error
endfunction
'**********************
' Close init CanOpen
'**********************
function close_cancfgerr() as void
endfunction
'**********************
' Custom error init
' CanOpen node
'**********************
function open_cancfgerr(nodes as int) as void
' Reset node 1 status error
Node_1_Error=0
endfunction
```

Codice in Init Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1   pos1.msof=10000  ' motor 10000 i/rev
2   pos1.ext_fcz=Fc_Home  ' home input
```

pos1.msof=10000 **' motor 10000 i/rev**
pos1.dsof=5000 **' 5 mm per revolution motor**

Codice in Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1    DigitalInputs=ng_di(0)  ' read digital inputs
2    pos1.ext_fcz=Fc_Home  ' home input
```

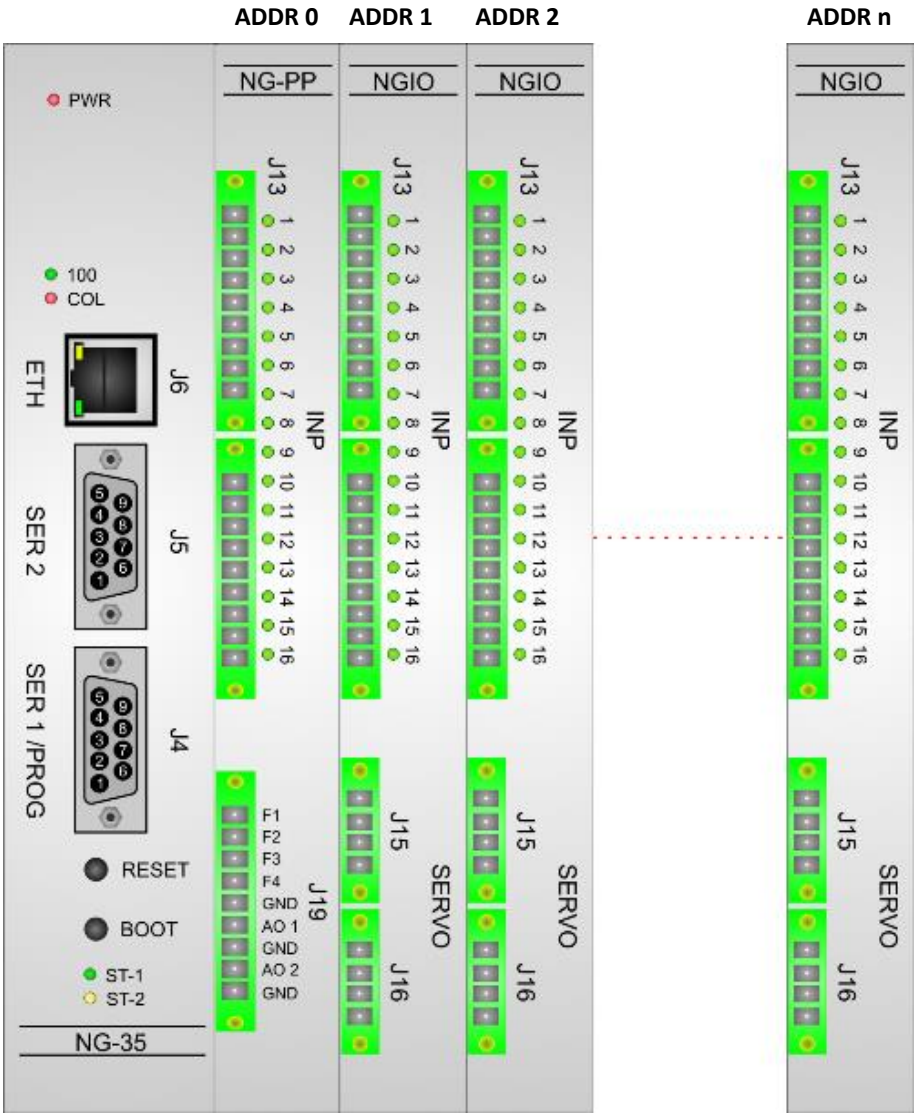
DigitalInputs=ng_di(0) **' read digital inputs**
pos1.ext_fcz=Fc_Home **' home input**

Scarica l' Esempio

9 INDIRIZZAMENTO NGIO-NGPP

Nella scheda NGWARP Possono essere montate schede di ESPANSIONE del tipo NGIO e NGPP.
Queste vengono viste dalle varie funzioni di gestione, con un indirizzo fisico da 0 a 7
L' indirizzo viene assunto in base all loro posizione nel BUS.
L' espansione più vicina alla scheda NGWARP assume indirizzo 0, quella successiva indirizzo 1 ecc.

Indice Fisico	Espansione Addr
0	**Scheda 0 (più vicina alla NGWARP)**
1	**Scheda 1**
2	**Scheda 2**
3	**Scheda 3**
4	**Scheda 4**
5	**Scheda 5**
6	**Scheda 6**
7	**Scheda 7**



10 I/ODigitali su NGIO-NGPP

Nella scheda NGWARP Possono essere montate schede di ESPANSIONE del tipo NGIO e NGPP.
Queste portano 16 ingressi digitali e 14 uscite Digitali gestibili da funzioni VTB.
Per l' indirizzamento vedi Cap. 8.

10.1 NG_DI – Lettura Ingressi Digitali

Legge lo stato degli ingressi digitali.
Lo stato degli ingressi viene gestito a bit partendo dal bit 0 fino al 15

Ingresso	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Sintassi

NG_DI(CardNumber as Char) as uint

Parametri

CardNumber Numero della scheda di espansione (da 0 a 7 vedi Cap. 8)

Valore di ritorno

Uint Valore dei 16 ingressi gestiti a BIT
bit = 1 → ingresso ON
bit = 0 → ingresso OFF

10.2 NG_DO – Scrittura Uscite Digitali

Scrive lo stato delle uscite digitali
Lo stato delle uscite digitali viene gestito a bit partendo dal bit 0 fino al 14

Uscita		14	13	12	11	10	9		8	7	6	5	4	3	2	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ATTENZIONE
I BIT 8 e 15 NON SONO UTILIZZATI

Sintassi

NG_DO(CardNumber as Char, StatoOutputs as Uint) as void

Parametri

CardNumber Numero della scheda di espansione (da 0 a 7 vedi Cap. 8)

StatoOutputs Stato delle uscite
bit = 1 → uscita ON
bit = 0 → uscita OFF

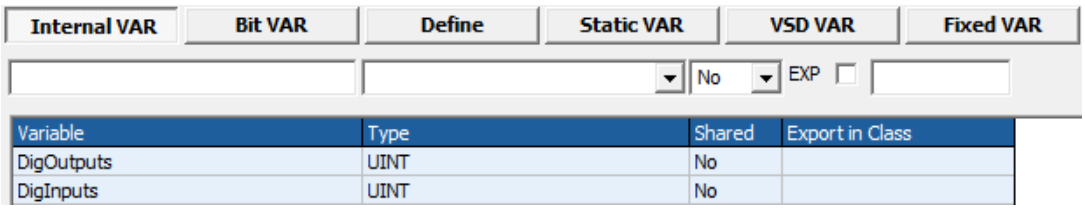
10.3 Esempio Utilizzo I/O Digitali

Nell' esempio riportato vengono gestite le I/O digitali nel seguente modo:

AGGIORNAMENTO DELLO STATO DEGLI INGRESSI E USCITE FATTO NEL TASK PLC

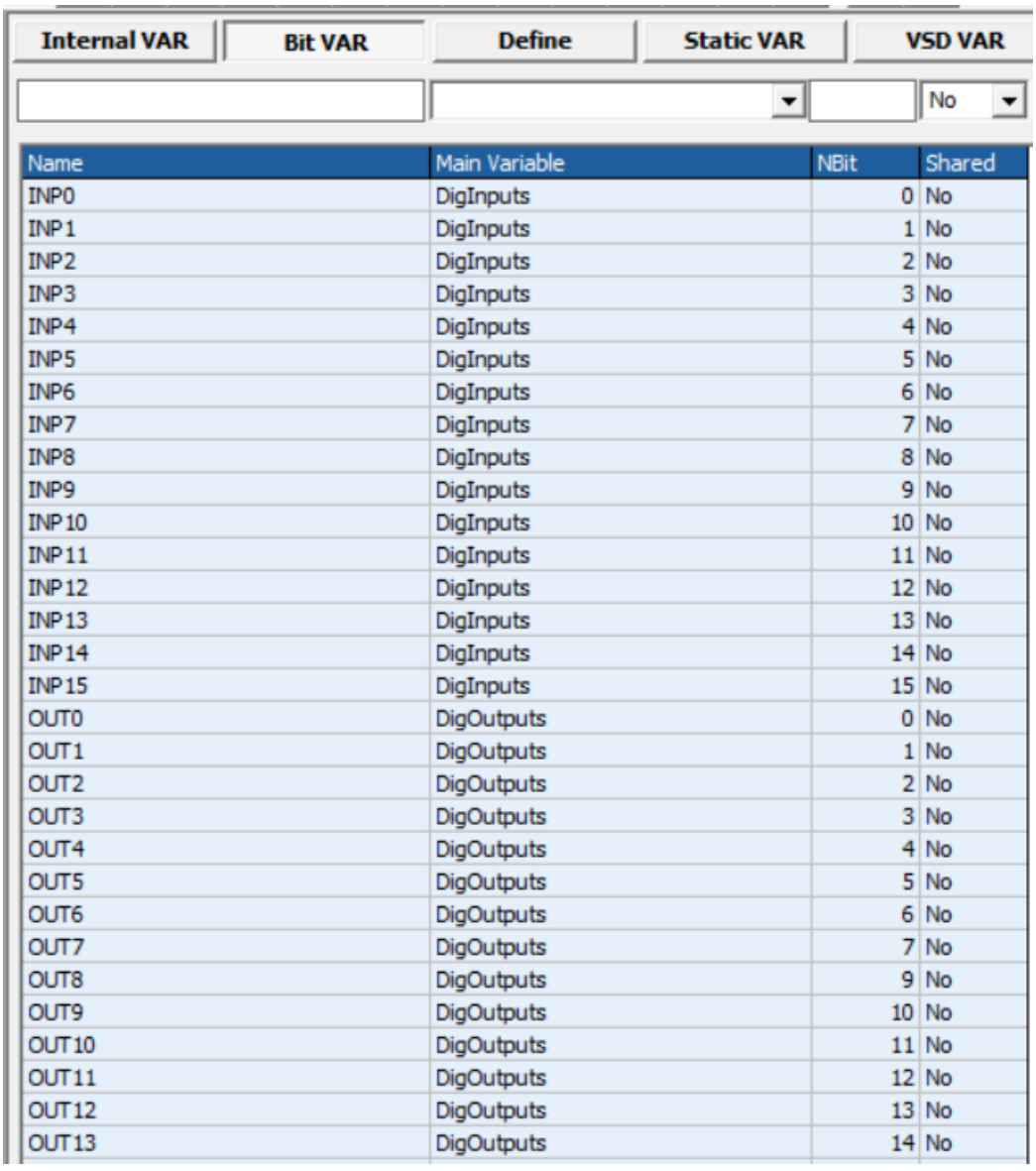
Gestione a bit delle I/O. I primi TRE ingressi ricopiano lo stato sulle prime 3 Uscite

Dichiarare le variabili del progetto



Variable	Type	Shared	Export in Class
DigOutputs	UINT	No	
DigInputs	UINT	No	

Dichiarare le variabili BIT del progetto



Name	Main Variable	NBit	Shared
INP0	DigInputs	0	No
INP1	DigInputs	1	No
INP2	DigInputs	2	No
INP3	DigInputs	3	No
INP4	DigInputs	4	No
INP5	DigInputs	5	No
INP6	DigInputs	6	No
INP7	DigInputs	7	No
INP8	DigInputs	8	No
INP9	DigInputs	9	No
INP10	DigInputs	10	No
INP11	DigInputs	11	No
INP12	DigInputs	12	No
INP13	DigInputs	13	No
INP14	DigInputs	14	No
INP15	DigInputs	15	No
OUT0	DigOutputs	0	No
OUT1	DigOutputs	1	No
OUT2	DigOutputs	2	No
OUT3	DigOutputs	3	No
OUT4	DigOutputs	4	No
OUT5	DigOutputs	5	No
OUT6	DigOutputs	6	No
OUT7	DigOutputs	7	No
OUT8	DigOutputs	9	No
OUT9	DigOutputs	10	No
OUT10	DigOutputs	11	No
OUT11	DigOutputs	12	No
OUT12	DigOutputs	13	No
OUT13	DigOutputs	14	No

Codice in Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1   OUT0=INP0  ' copy input 0 on outpus 0
2   OUT1=INP1  ' copy input 1 on outpus 1
3   OUT2=INP2  ' copy input 2 on outpus 2
```

OUT0=INP0 **' copy input 0 on outpus 0**
OUT1=INP1 **' copy input 1 on outpus 1**
OUT2=INP2 **' copy input 2 on outpus 2**
DigInputs=**ng_di**(0) **' udpate digital inputs**
ng_do(0,DigOutputs) **' update digital outputs**

Scarica l' Esempio

11 Uscite analogiche 2 relè Su NGIO-NGPP

Le schede di espansione,NGIO e NGPP gestiscono 2 uscite analogiche +/- 10 V 12 bit.
La scheda NGIO può gestire anche 2 relè da 1 A.

11.1 NG_DAC - SCRITTURA USCITE ANALOGICHE NGIO-NGPP

Questa funzione permette di pilotare l'uscita analogica di ogni canale gestito dal controllo NGWARP tramite l' espansioni **NG-IO** e **NG-PP** (opzionale).
Le schede dispongono di convertitori digitale/analogico a 12 bit, su un fondo scala di 10V. Per cui con +2047 si ottengono 10V in uscita, con -2047 -10V in uscita.
La selezione del canale avviene con un indice che va da 0 a 7, ogni espansione gestisce due canali quindi avremo:

Indice Canale	Espansione Addr
0	Scheda 0
1	
2	Scheda 1
3	
4	Scheda 2
5	
6	Scheda 3
7	

Il numero massimo di canali analogici gestiti è di 8.

Sintassi

NG_DAC(Channel as Char, Val as Long) as void

Parametri
Channel Numero Canale (da 0 a 7)
val Valore dell'uscita (da -2047 a +2047)

11.2 NG_DAC_CAL - OFFSET USCITE ANALOGICHE NGIO-NGPP

Questa funzione permette di calibrare l' OFFSET dell'uscita analogica di ogni CANALE gestito dal controllo NGWARP. Vari fattori dovuti all' hardware fanno sì che, anche se viene pilotata con valori nulli, l'uscita risulti diversa da 0 nell'ordine di pochi mV (offset). Questa funzione permette di azzerare questo effetto.
Con questa funzione si sposta in maniera software lo 0V dell'uscita che comunque non potrà mai essere esattamente nulla. Considerando che per ogni unità si ottengono circa 4mV in uscita, i valori da passare dovrebbero essere nell'ordine di poche unità, in senso opposta all'uscita rilevata.

Sintassi

NG_DAC_CAL(Channel as Char, Offset as Long) as void

Parametri
Channel Numero Canale (da 0 a 7)
Offset Valore dell' OFFSET (da -2047 a +2047)

ATTENZIONE
IL VALORE DELL'OFFSET NON RIMANE MEMORIZZATO, PERTANTO AD OGNI ACCENSIONE OCCORRE IMPOSTARLO DI NUOVO
Salvare il valore offset in memoria NGWARP

11.3 NG_RELE - GESTIONE DEI RELE' NGIO

Questa funzione permette la gestione dei 2 RELE' presenti su ogni espansione **NG-IO**.
Normalmente questi RELE' servono per l' ABILITAZIONE DEL DRIVER del relativo ASSE ma possono essere utilizzate anche per scopi generici.

Indice Canale	Espansione Addr
0	**Scheda 0**
1	
2	**Scheda 1**
3	
4	**Scheda 2**
5	
6	**Scheda 3**
7	
.....	....
14	**Scheda 7**
15	

Sintassi

NG_RELE(Channel **as Char**, Stato **as Char**) **as void**

Parametri

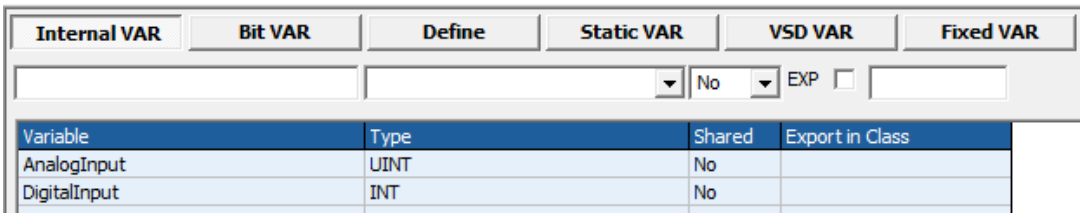
Channel Canale relativo all'attivazione del relè (da 0 a 16)

Stato Stato del relè
0 OFF(contatto aperto)
1 ON (contatto chiuso)

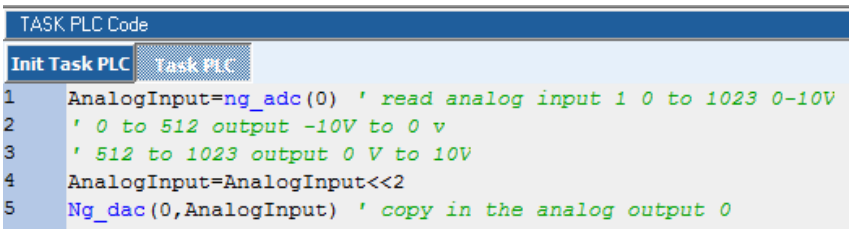
11.4 Esempio Utilizzo Uscite Analogiche e contatto Relè

Nell' esempio riportato vengono gestite le uscite Analogiche e il contatto del relè.
Nell' esempio, viene letto il canale analogico 0 e riportato sull' uscita analogica 0.
L' ingresso 1 viene ricopiato sul relè 0.

Dichiarare le variabili del progetto



Codice in Task PLC




```
AnalogInput=ng_adc(0) ' read analog input 1 0 to 1023 0-10V
' -512 0 output -10V to 0 v
' 0 512 output 0 V to 10V
AnalogInput=AnalogInput-1024
Ng_dac(0,AnalogInput) ' copy in the analog output 0
DigitalInput=ng_di(0) ' read digital input
if DigitalInput & 1 ' test input 1
        ng_rele(0,1) ' set rele'
else
        ng_rele(0,0) ' reset rele'
endif
```

Scarica l' Esempio

12 Lettura Encoder e Indice di ZERO su NGIO

Le schede di espansione,NGIO mette a disposizione 2 canali encoder incrementali con segnali LINE DRIVE 5 v. Oltre a questo può essere letto l' indice di zero del canale encoder (tacca di zero)

12.1 NG_ENC - GESTIONE ENCODER

Questa funzione permette di leggere l'ingresso encoder (quota incrementale attuale) di ogni canale gestito sulla scheda di espansione **NG-IO** con una risoluzione di 32 bit. Questa funzione legge soltanto l'incremento che sarà aggiunto alla variabile passata con un puntatore. Il contatore vero e proprio quindi sarà contenuto in una variabile definita dall'applicazione e che potrà essere azzerata in qualunque momento. Per una corretta gestione degli encoder raccomandiamo di utilizzare questa funzione esclusivamente nel TASK PLC e quindi utilizzare la variabile di appoggio all'occorrenza. La selezione del canale avviene con un indice che va da 0 a 15, ogni espansione gestisce due canali quindi avremo:

Indice Canale	Espansione Addr
0	Scheda 0
1	
2	Scheda 1
3	
4	Scheda 2
5	
6	Scheda 3
7	
.....	....
14	Scheda 7
15	

Sintassi

NG_ENC(Channel as Char, Value as *Long) as void

Parametri

Channel Numero canale encoder (da 0 a 15)

Value Puntatore alla variabile di tipo long in cui viene memorizzato il conteggio

ATTENZIONE

PER UN CORRETTO FUNZIONAMENTO INSERIRE QUESTA FUNZIONE NEL TASK_PLC

12.2 NG-T0 - LETTURA INDICE DI ZERO ENCODER

Questa funzione permette di leggere lo stato dell'ingresso tacca di zero di ogni canale encoder gestito sulla scheda di espansione **NG-IO**. Per la selezione del canale vale quanto detto per la lettura encoder.

Sintassi

NG_T0(Channel **as Char**) **as char**

Parametri

Channel Numero canale encoder (da 0 a 15)

Valore di ritorno

Stato della tacca:

0 OFF

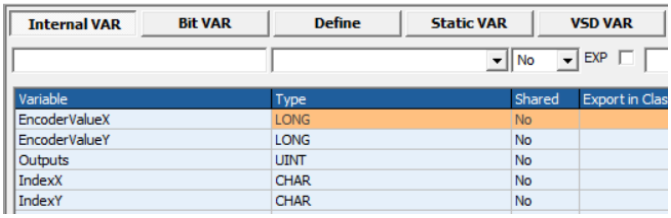
1 ON

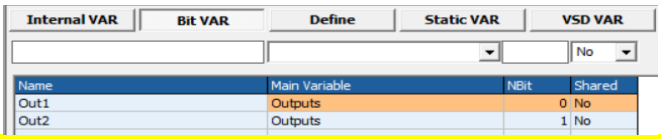
SU HARDWARE NGIO 2.0, E' CONSIGLIABILE LEGGERE LA TACCA DI ZERO, TRAMITE GLI INGRESSI FAST INPUT vedi CAP 12

12.3 Esempio Lettura Encoder NGIO e TACCA DI ZERO

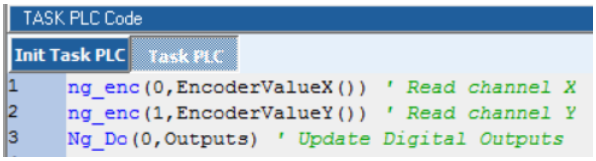
Nell' esempio riportato vengono letti due canali encoder e le due tacche di zero su NGIO Addr 0

Dichiarare le variabili del progetto



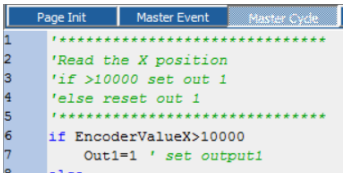


Codice in Task PLC




```
ng_enc(0,EncoderValueX()) ' Read channel X
ng_enc(1,EncoderValueY()) ' Read channel Y
Ng_Do(0,Outputs) ' Update Digital Outputs
```

Codice in Task MAIN




```
'*******************************
'Read the X position
'if >10000 set out 1
'*******************************
if EncoderValueX>10000
        Out1=1 ' set output1
else
        Out1=0 ' reset output1
endif
'*******************************
'Read the Y position
'if >5000 set out 2
'*******************************
if EncoderValueY>5000
        Out2=1 ' set output2
else
        Out2=0 ' reset output2
endif
IndexX=ng_t0(0) 'read index X
IndexY=ng_t0(1) 'read index Y
```

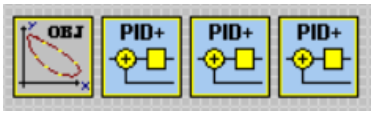
Scarica l' Esempio

12.4 Esempio Utilizzo Assi Interpolati Analogici

Nell' esempio riportato, vengono gestiti TRE assi Analogici con filtro PID in modo interpolazione lineare.

ATTENZIONE:
Tutte le velocità vengono inserite in mm/min se impostato correttamente i parametri:
RapX,RapY,RapZ
Tutte le quote assi vengono inserite in micron (0.001 mm) se impostato correttamente i parametri:
RapX,RapY,RapZ

Oggetti utilizzati:



Motor Control → CobjInterpola → Interpolatore

Project Explorer

Project | Objects | Functions | Properties | Tables

Interp

Property | Events

Property	Value
Nome	Interp
Left	15
Top	10
N.assi	3
N.tratti	16
Vper	1024
Div. Vper	1024
Abilita arcto	1

Motor Control Plus→ CPidPlus → Pid NG

Property	Value
Nome	PidX
Left	55
Top	10
EnablePid	False
Kp	10
Ki	0
Kv	0
Kd	0
Err_Sat	10000
NG ENC CHANNEL	0
NG DAC CHANNEL	0
ENABLE KP	True
ENABLE KI	True
ENABLE KV	True
ENABLE KD	False
Divisore	100
Dir	1
ServoErr	10000
TServoErr	1000
EnableDelay	50

Property	Value
Nome	PidY
Left	100
Top	10
EnablePid	False
Kp	10
Ki	0
Kv	0
Kd	0
Err_Sat	10000
NG ENC CHANNEL	1
NG DAC CHANNEL	1
ENABLE KP	True
ENABLE KI	True
ENABLE KV	True
ENABLE KD	False
Divisore	100
Dir	1
ServoErr	10000
TServoErr	1000
EnableDelay	50

Property	Value
Nome	PidZ
Left	145
Top	10
EnablePid	False
Kp	10
Ki	0
Kv	0
Kd	0
Err_Sat	10000
NG ENC CHANNEL	2
NG DAC CHANNEL	2
ENABLE KP	True
ENABLE KI	True
ENABLE KV	True
ENABLE KD	False
Divisore	100
Dir	1
ServoErr	10000
TServoErr	1000
EnableDelay	50

Vengono gestite le seguenti funzioni:

Wait_Move – Stato movimento assi
Parametri ***Nessuno***
Ritorna ***1 Assi in movimento***
0 Assi fermi
Move_Axes – Interpolazione lineare
Parametri ***Vel → Velocità assi in millimetri/min***
Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)
Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)
Px,Py,Pz → Quote Assi in 0.001 mm
Ritorna ***0 Movimento inserito nel buffer***
1 Buffer pieno (occorre ripetere Move_Axes fino a che il buffer è libero)
Acc_Axes – Set Accelerazione di Interpolazione
Parametri ***Value → Valore Accelerazione di interpolazione in count per TAU***
Ritorna ***Nessuno***

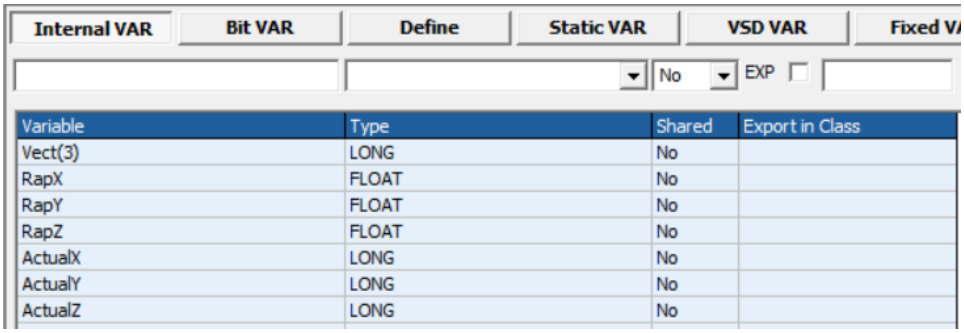
Stop_Axes – Stop Movimento
Parametri ***Nessuno***
Ritorna ***Nessuno***

Enable_Axis_X_Y_Z – Abilita controllo Asse
Parametri ***Nessuno***
Ritorna ***Nessuno***

Disable_Axis_X_Y_Z – Disabilita controllo Asse
Parametri ***Nessuno***
Ritorna ***Nessuno***

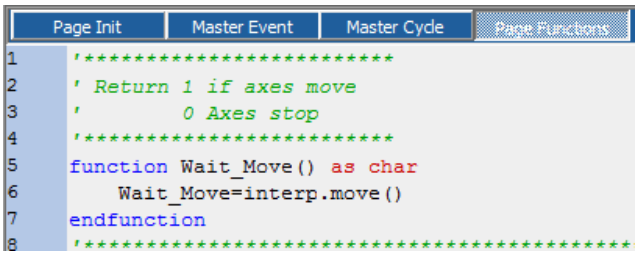
Test_Following_Error – Controlla errore inseguimento assi
Disabilita in caso di errore tutti gli assi
Parametri ***Nessuno***
Ritorna ***Nessuno***

Dichiarare le variabili del progetto



Variable	Type	Shared	Export in Class
Vect(3)	LONG	No	
RapX	FLOAT	No	
RapY	FLOAT	No	
RapZ	FLOAT	No	
ActualX	LONG	No	
ActualY	LONG	No	
ActualZ	LONG	No	

Codice in Funzioni di Pagina Main




```
'*************************
' Return 1 if axes move
'      0 Axes stop
'*************************
function Wait_Move() as char
        Wait_Move=interp.move()
endfunction

'*********************************************************
' Move Axes
' Vel= interp vel Axes in mm/min
' Flg if 1 move without buffer
'      0 move in buffer mode
' Px,Py,Pz Axes value in 0.001 mm
'Return 1 if movement is inserted in the buffer
'     0 The movement is not inserted in the buffer
'       in this case, is necessary reload the movement
'*********************************************************
function Move_Axes(Vel as long, Flg as char, Px as long, Py as long,Pz as long) as char
        Vel=Vel*TAU/60 ' Transform in mm/min
        Vect(0)=Px
        Vect(1)=Py
        Vect(2)=Pz
        Move_Axes=interp.moveto(Vel, Flg, Vect())
endfunction

'*********************************************************
' Set ACC
' Value Acc value in count
'*********************************************************
function Acc_Axes(Value as long) as void
```

```
        interp.acc=Value
endfunction



'*********************************************************
' Stop Axes
'*********************************************************
function Stop_Axes() as void
        interp.stop()
endfunction
'*******************
' Axis X enable
'*******************
function Enable_X() as void
PidX.enablepid=0
'Preset Axis X 0, not change y,z
Vect(0)=0
Vect(1)=interp.pc(1)
Vect(2)=interp.pc(2)
interp.preset(Vect())
PidX.posr=0
'enable axis
PidX.enablepid=1
PidX.enable() ' closes the rele' on NGIO
endfunction
'*******************
' Axis X Disable
'*******************
function Disable_X() as void
Pidx.disable()
endfunction
'*******************
' Axis Y enable
'*******************
function Enable_Y() as void
PidY.enablepid=0
'Preset Axis Y 0, not change X,z
Vect(0)=interp.pc(0)
Vect(1)=0
Vect(2)=interp.pc(2)
interp.preset(Vect())
PidY.posr=0
'enable axis
PidY.enablepid=1
PidY.enable() ' closes the rele' on NGIO
endfunction
'*******************
' Axis Y Disable
'*******************
function Disable_Y() as void
PidY.disable()
endfunction
'*******************
' Axis Z enable
```

```
'*******************
function Enable_Z() as void
PidZ.enablepid=0
'Preset Axis Z 0, not change X,Y
Vect(0)=interp.pc(0)
Vect(1)=interp.pc(1)
Vect(2)=0
interp.preset(Vect())
PidZ.posr=0
'enable axis
PidZ.enablepid=1
PidZ.enable() ' closes the rele' on NGIO
endfunction
'*******************
' Axis Z Disable
'*******************
function Disable_Z() as void
PidZ.disable()
endfunction
'***************************
' Test following error
' Disable all Axes if error
'***************************
function Test_Following_Error()as void
dim Error as char
error=0
if PidX.err=1 ' test X
        error=1
endif
if PidY.err=1 ' test Y
        error=1
endif
if PidZ.err=1 ' test Z
        error=1
endif
if error=1 'if serror disable all motor
        Disable_X()
        Disable_Y()
        Disable_Z()
endif
endfunction
```

Codice in Init Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1  '************************************************
2  'Ex: Motor Encoder Revolution = 10000 i/rev
3  'Motor inserted directly in the Screw 5 mm step
4  'Rap=10000/5000=2
5  '************************************************
6  Rapx=1
7  Rapy=1
8  Rapz=1
```

```
'************************************************
```

```
'Ex: Motor Encoder Revolution = 10000 i/rev
'Motor inserted directly in the Screw 5 mm step
'Rap=10000/5000=2
'************************************************
Rapx=1
Rapy=1
Rapz=1
```

Codice in Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1   'Write the PID Axes
2   PidX.post=interp.pc(0)*RapX
3   PidY.post=interp.pc(1)*RapY
4   PidZ.post=interp.pc(2)*RapZ
```

```
'Write the PID Axes
PidX.post=interp.pc(0)*RapX
PidY.post=interp.pc(1)*RapY
PidZ.post=interp.pc(2)*RapZ
'read analog 0 and set the Vper %
interp.vper=ng_adc(0)
' copy the axes values
' for ex: display in HMI
' value in 0.001 mm
ActualX=PidX.posr ' read actual position X
ActualY=PidY.posr ' read actual position Y
ActualZ=PidZ.posr ' read actual position Z
```

Scarica l' Esempio

12.5 Esempio Utilizzo Asse Posizionato Analogico

Nell' esempio riportato, viene gestito un asse Analogico posizionato utilizzando un oggetto VTB
Vedere il manuale relativo agli oggetti di VTB per ulteriori spiegazioni

ATTENZIONE:

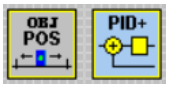
Tutte le velocità vengono inserite in mm/min se impostato correttamente i parametri:

MSOF e DSOF

Tutte le quote assi vengono inserite in micron (0.001 mm) se impostato correttamente i parametri:

MSOF e DSOF

Oggetti utilizzati:



Motor Control Plus → CobjPos → Posizionatore

Property	Value
Nome	Pos1
Left	25
Top	30
N.TRATTI	8
Vper	1024
Div. Vper	1024
AccQstop	10
Acc	5
RzeroMode	1
RzeroOffset	0
RzeroPreset	0
RzeroVel	10
RzeroVelf	5
RzeroAcc	10
Msof	10000
Dsof	5000
LimitN	-99999999
LimitP	99999999
Gioco	0
Vgioco	1
MsofV	1
DsofV	1
RZERO ENABLE	True
AXIS TYPE	4
VTB AXIS OBJECT	PidX
PDO NAME	0
STEP CHANNEL	0
STEP NODE	1

Motor Control Plus→ CPidPlus → Pid NG

Property	Value
Nome	PidX
Left	75
Top	30
EnablePid	False
Kp	10
Ki	0
Kv	0
Kd	0
Err_Sat	10000
NG ENC CHANNEL	0
NG DAC CHANNEL	0
ENABLE KP	True
ENABLE KI	True
ENABLE KV	True
ENABLE KD	False
Divisore	100
Dir	1
ServoErr	10000
TServoErr	1000
EnableDelay	50

Vengono gestite le seguenti funzioni:

Wait_Move – Stato movimento asse

Parametri ***Nessuno***

Ritorna ***1 Asse in movimento***

0 Asse fermi

Move_Axis – Interpolazione lineare

Parametri ***Vel → Velocità assi in millimetri/min***

Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)

Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)

Px → Quota Asse in 0.001 mm

Ritorna ***0 Movimento inserito nel buffer***

1 Buffer pieno (occorre ripetere Move_Axes fino a che il buffer è libero)

Preset – Preset quota Asse

Parametri ***Px → Quota Asse in 0.001 mm per preset***

Ritorna ***Nessuno***

Acc_Axis – Set Accelerazione/decelrazione di posizionamento

Parametri ***Value → Valore Accelerazione di posizionamento in count per TAU***

Ritorna ***Nessuno***

Stop – Stop Movimento

Parametri ***Nessuno***

Ritorna ***Nessuno***

Enable – Abilita controllo Asse

Parametri ***Nessuno***

Ritorna ***Nessuno***

Disable – Disabilita controllo Asse

Parametri ***Nessuno***

Ritorna ***Nessuno***

StartHome – Start ricerca homing Vel in pos1.rzerovel e pos1.rzerovelf

Parametri ***Nessuno***

Ritorna ***Nessuno***

CheckHome – Check stato homing in corso

Parametri ***Nessuno***

Ritorna ***1 homing terminato***

StopHome – Stop homing in corso

Parametri ***Nessuno***

Ritorna ***Nessuno***

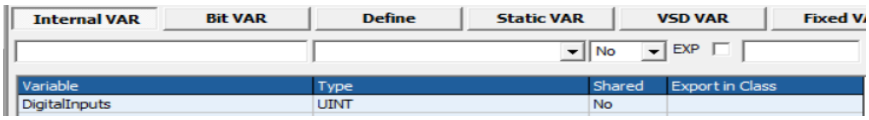
Test_Following_Error – Controlla errore inseguimento asse

Disabilita in caso di errore l' asse

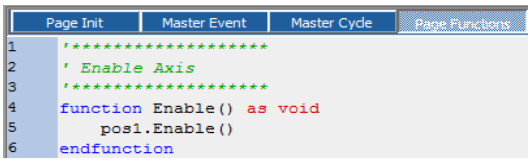
Parametri ***Nessuno***

Ritorna ***Nessuno***

Dichiarare le variabili del progetto



Codice in Funzioni di Pagina Main




```
'*******************
' Enable Axis
'*******************
function Enable() as void
        pos1.Enable()
endfunction
'*******************
' Disable Axis
'*******************
function Disable() as void
        pos1.Disable()
endfunction
'*******************
' Preset Axis
'*******************
function Preset(Val as long) as void
        pos1.Preset(Val)
endfunction
'*************************
' Return 1 if axis move
'      0 Axis stop
'*************************
function Wait_Move() as char
        Wait_Move=pos1.move()
endfunction
'*******************
' Axis Stop Move
'*******************
function Stop() as void
        pos1.Stop()
endfunction
'*****************************
' Start Homing
' Homing input see in task plc
'*****************************
function StartHome() as void
        pos1.StartHome()
endfunction
'*****************************
' Check if homing finished
```

```
' Return 1 if finished
'*****************************
function CheckHome() as char
        CheckHome=pos1.status_home
endfunction
'*****************************
' Stop home function
'*****************************
function StopHome() as void
        pos1.StopHome()
endfunction
'*************************************************************
' Move Axis
' Vel=  vel Axis in mm/min
' Flg if 1 move without buffer
'       0 move in buffer mode
' Px  Axis value in 0.001 mm
'Return 1 if movement is inserted in the buffer
'      0 The movement is not inserted in the buffer
'       in this case, is necessary reload the movement
'*************************************************************
function Move_Axis(Vel as long, Flg as char, Px as long) as char
        Vel=Vel*TAU/60 ' Transform in mm/min
        Move_Axis=pos1.moveto(Vel, Flg, Px)
endfunction
'*************************************************************
' Set ACC
' Value Acc value in count
'*************************************************************
function Acc_Axis(Value as long) as void
        pos1.acc=Value
endfunction

'***************************
' Test following error
' Disable Axis
'***************************
function Test_Following_Error() as void
if PidX.err=1 ' test Axis
        Disable()
endif
endfunction
```

Codice in Init Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1   pos1.msof=10000 ' motor 10000 i/rev
2   pos1.ext_fcz=Fc_Home ' home input
```

pos1.msof=10000 ' **motor 10000 i/rev**
pos1.dsof=5000 ' **5 mm per revolution motor**

Codice in Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1    DigitalInputs=ng_di(0)  ' read digital inputs
2    pos1.ext_fcz=Fc_Home  ' home input
```

DigitalInputs=ng_di(0) **' read digital inputs**
pos1.ext_fcz=Fc_Home **' home input**

Scarica l' Esempio

13 FAST INPUTS Interrupt NGIO-NGPP

Le schede di espansione,NGIO e NGPP mettono a disposizione 2 (per NGIO su tacche di zero) 4 (per NGPP su FAST INPUTS 1-4) ingressi che possono essere gestiti in modo ad INTERRUPT dall' hardware.
Questo modo aumenta il tempo di risposta (lettura ingresso) rispetto ad una normale lettura.
Gli ingressi veloci della scheda NGPP (fast inputs 1-4) possono essere letti solo nel modo ad INTERRUPT.
Gli ingressi veloci della scheda NGIO (tacche di zero) oltre che nel modo ad INTERRUPT possono essere letti tramite la funzione NG_T0() Vedi Cap 11.2

13.1 OGGETTO FAST INPUT - GESTIONE INTERRUPT INGRESSI

Per leggere gli ingressi in modo INTERRUPT, viene utilizzato l 'oggetto:

General → Fast Input → NGWARP Finput

Proprietà principali

Card Index Indice della scheda NGIO o NGPP nel bus locale:
da 0 a 7

Channel Numero di canale di ingresso da associare
NGIO da 0 a 1 - tacche di 0
NGPP da 0 a 3 – Fast Inputs

Metodi

Nome.get() as void

Update nelle registri di appoggio dei latch relativi al fronte di salita o discesa
(metodo da chiamare sempre prima di ogni lettura dei fronti UP e DN)

Nome.clear() as void

Resetta i registri di appoggio dei latch relativi al fronte di salita o discesa
Questo metodo una volta chiamato, azzera gli stati dei registri UP e DN, ma se lo stato dell' ingresso si trova per es. ALTO, il relativo latch UP viene immediatamente settato. (viceversa per DN)

Variabili di sola lettura

Nome.stato Contiene lo stato attuale (0 o 1) dell' ingresso selezionato
(questa non è gestita ad INTERRUPT)

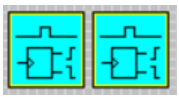
Nome.up Contiene lo stato del LATCH del fronte di salita.
Questa funzione è gestita in modo HARDWARE INTERRUPT dal sistema operativo

Nome.dn Contiene lo stato del LATCH del fronte di discesa.
Questa funzione è gestita in modo HARDWARE INTERRUPT dal sistema operativo

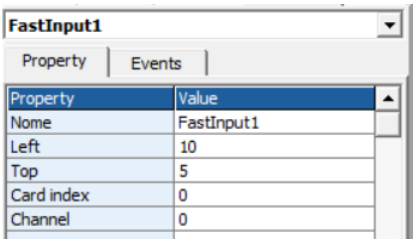
13.2 Esempio Lettura ingressi ad Interrupt

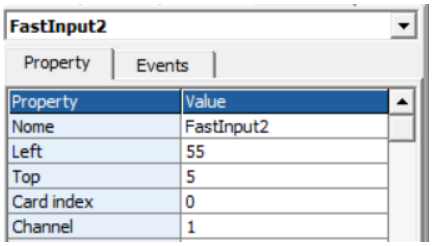
Nell' esempio riportato, vengono letti gli ingressi ad interrupt 1 e di una scheda NGIO-NGPP

Oggetti utilizzati:

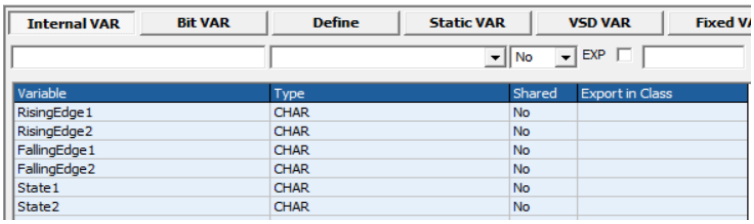


General → Fast Input → NGWARP Finput





Dichiarare le variabili del progetto



Codice in Task Main – Master ciclo


```
'***********************
'FAST INPUT 1
'***********************
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
'***********************
'FAST INPUT 2
'***********************
Fastinput2.get()              'get fastinput2
RisingEdge2=FastInput2.up     'ceck the rising edge
FallingEdge2=FastInput2.dn    'ceck the falling edge
State2=FastInput2.inp         'read the state
FastInput2.clear()            'reset latch up e dn
```

Scarica l' Esempio

14 Gestione canali STEP su NGPP

Le schede di espansione,NGPP mette a disposizione 4 canali STEP/DIR gestibili da funzioni VTB, in interpolazione o posizionamento

14.1 PP_STEP – GENERAZIONE DEI SEGNALI STEP/DIR

Questa funzione **PP_STEP** è la PRIMITIVA che permette la generazione dei PASSI STEP sul canale indicato. Generalmente il suo utilizzo è abbinato a generatori di RAMPA e POSIZIONE derivati dai relativi OGGETTI.

Sintassi

PP_STEP(Channel as Char, Value as Long) as void

Parametri

Channel Numero del canale STEP/DIR
da 0 a 3 canali prima espansione NGPP
4 a 7 canali seconda espansione NGPP
da 28 a 31 canali ultima espansione NGPP

Value Valore assoluto della posizione dell'asse step/dir

ATTENZIONE
PER UN CORRETTO FUNZIONAMENTO INSERIRE QUESTA FUNZIONE NEL TASK_PLC

14.2 PP_PRESET – PRESET DEL GENERATORE STEP/DIR

Questa funzione consente di cambiare la posizione corrente di un generatore step/dir.

Sintassi

PP_PRESET(Channel as Char, Value as Long) as void

Parametri

Channel Numero del canale STEP/DIR
Value Valore della posizione che assumerà il l'asse step/dir

ATTENZIONE
PER UN CORRETTO PRESET DEGLI ASSI vedere gli esempi riportati per INTERPOLATORE e POSIZIONATORE

14.3 PP_GETPOS – LETTURA POSIZIONE ATTUALE

Questa funzione consente di leggere la posizione corrente di un generatore step/dir. **Il valore letto corrisponderà al DOPPIO dei passi generati**.

Sintassi

PP_GETPOS(Channel as Char) as long

Parametri

Channel Numero del canale STEP/DIR

Valore di ritorno
Long *Posizione attuale x 2*

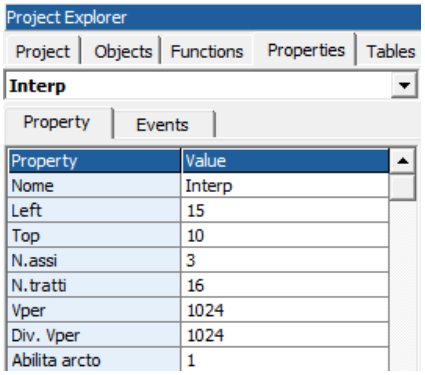
14.4 Esempio Utilizzo Assi Interpolati STEP/DIR

Nell' esempio riportato, vengono gestiti TRE assi STEP/DIR in modo interpolazione lineare.

ATTENZIONE:
Tutte le velocità vengono inserite in mm/min se impostato correttamente i parametri:
RapX,RapY,RapZ
Tutte le quote assi vengono inserite in micron (0.001 mm) se impostato correttamente i parametri:
RapX,RapY,RapZ

Oggetti utilizzati:

Motor Control → CobjInterpola → Interpolatore



Vengono gestite le seguenti funzioni:

Wait_Move – Stato movimento assi
Parametri ***Nessuno***
Ritorna ***1 Assi in movimento***
0 Assi fermi

Move_Axes – Interpolazione lineare
Parametri ***Vel → Velocità assi in millimetri/min***
Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)
Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)
Px,Py,Pz → Quote Assi in 0.001 mm
Ritorna ***0 Movimento inserito nel buffer***
1 Buffer pieno (occorre ripetere Move_Axes fino a che il buffer è libero)

Acc_Axes – Set Accelerazione di Interpolazione
Parametri ***Value → Valore Accelerazione di interpolazione in count per TAU***
Ritorna ***Nessuno***

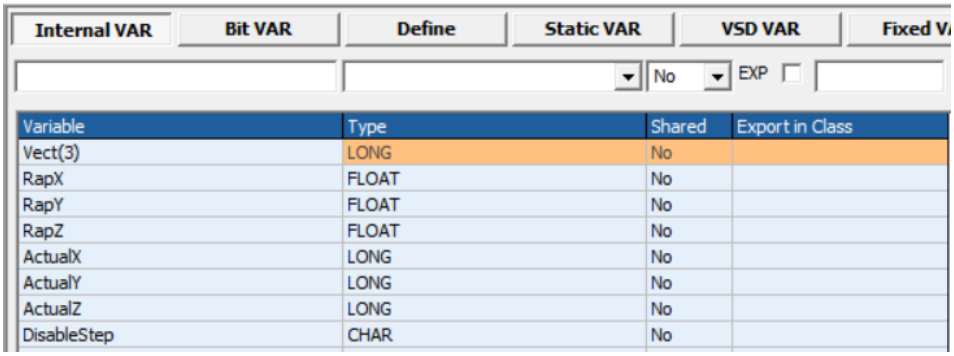
Stop_Axes – Stop Movimento
Parametri ***Nessuno***
Ritorna ***Nessuno***

Enable_Axis_X_Y_Z – Abilita controllo Asse

Parametri ***Nessuno***

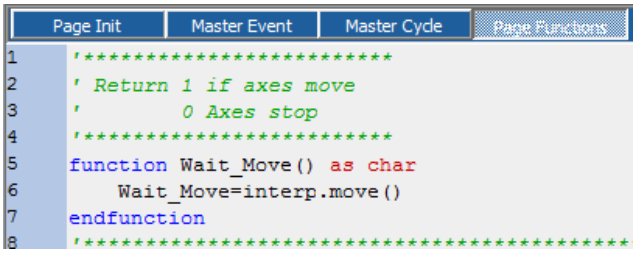
Ritorna ***Nessuno***

Dichiarare le variabili del progetto



Variable	Type	Shared	Export in Class
Vect(3)	LONG	No	
RapX	FLOAT	No	
RapY	FLOAT	No	
RapZ	FLOAT	No	
ActualX	LONG	No	
ActualY	LONG	No	
ActualZ	LONG	No	
DisableStep	CHAR	No	

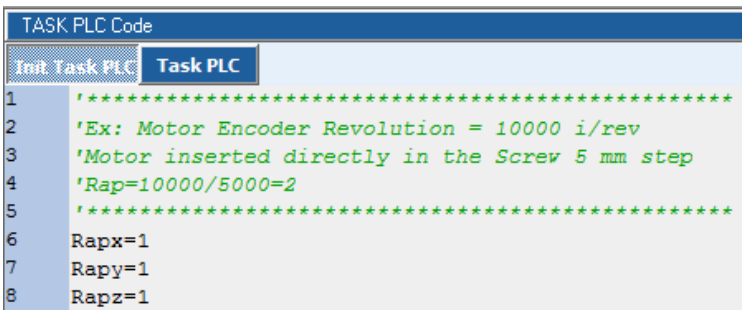
Codice in Funzioni di Pagina Main




```
'*************************
' Return 1 if axes move
'      0 Axes stop
'*************************
function Wait_Move() as char
        Wait_Move=interp.move()
endfunction

'*************************************************************
' Move Axes
' Vel= interp vel Axes in mm/min
' Flg if 1 move without buffer
'      0 move in buffer mode
' Px,Py,Pz Axes value in 0.001 mm
'Return 1 if movement is inserted in the buffer
'      0 The movement is not inserted in the buffer
'       in this case, is necessary reload the movement
'*************************************************************
function Move_Axes(Vel as long, Flg as char, Px as long, Py as long,Pz as long) as char
        Vel=Vel*TAU/60 ' Transform in mm/min
        Vect(0)=Px
        Vect(1)=Py
        Vect(2)=Pz
        Move_Axes=interp.moveto(Vel, Flg, Vect())
endfunction
```

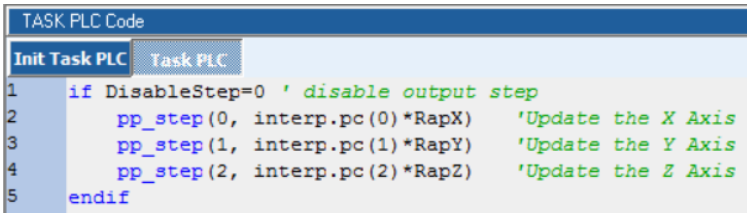
```
'*********************************************************
' Set ACC
' Value Acc value in count
'*********************************************************
function Acc_Axes(Value as long) as void
        interp.acc=Value
endfunction

'*********************************************************
' Stop Axes
'*********************************************************
function Stop_Axes() as void
        interp.stop()
endfunction
'*******************
' Axis X enable
'*******************
function Enable_X() as void
DisableStep=1
'Preset Axis X 0, not change y,z
Vect(0)=0
Vect(1)=interp.pc(1)
Vect(2)=interp.pc(2)
interp.preset(Vect())
'enable axis
DisableStep0=0
endfunction
'*******************
' Axis Y enable
'*******************
function Enable_Y() as void
DisableStep=1
'Preset Axis Y 0, not change X,z
Vect(0)=interp.pc(0)
Vect(1)=0
Vect(2)=interp.pc(2)
interp.preset(Vect())
'enable axis
DisableStep=0
endfunction
'*******************
' Axis Z enable
'*******************
function Enable_Z() as void
DisableStep=1
'Preset Axis Z 0, not change X,Y
Vect(0)=interp.pc(0)
Vect(1)=interp.pc(1)
Vect(2)=0
interp.preset(Vect())
PidZ.posr=0
'enable axis
DisableStep=0
endfunction
```

Codice in Init Task PLC




```
'************************************************
'Ex: Motor Encoder Revolution = 10000 i/rev
'Motor inserted directly in the Screw 5 mm step
'Rap=10000/5000=2
'************************************************
Rapx=1
Rapy=1
Rapz=1
```

Codice in Task PLC




```
if DisableStep=0 ' disable output step
        pp_step(0, interp.pc(0)*RapX)      'Update the X Axis
        pp_step(1, interp.pc(1)*RapY)      'Update the Y Axis
        pp_step(2, interp.pc(2)*RapZ)      'Update the Z Axis
endif
'read analog 0 and set the Vper %
interp.vper=ng_adc(0)
' copy the axes values
' for ex: display in HMI
' value in 0.001 mm
ActualX=interp.pc(0) ' read actual position X
ActualY=interp.pc(1) ' read actual position Y
ActualZ=interp.pc(2) ' read actual position Z
```

Scarica l' Esempio

14.5 Esempio Utilizzo Asse Posizionato STEP/DIR

Nell' esempio riportato, viene gestito un asse Step/Dir posizionato utilizzando un oggetto VTB
Vedere il manuale relativo agli oggetti di VTB per ulteriori spiegazioni
ATTENZIONE:
Tutte le velocità vengono inserite in mm/min se impostato correttamente i parametri:
MSOF e DSOF
Tutte le quote assi vengono inserite in micron (0.001 mm) se impostato correttamente i parametri:
MSOF e DSOF

Oggetti utilizzati:

Motor Control Plus → CobjPos → Posizionatore

Property	Value
Nome	Pos1
Left	25
Top	30
N.TRATTI	8
Vper	1024
Div. Vper	1024
AccQstop	10
Acc	5
RzeroMode	1
RzeroOffset	0
RzeroPreset	0
RzeroVel	10
RzeroVelf	5
RzeroAcc	10
Msof	10000
Dsof	5000
LimitN	-99999999
LimitP	99999999
Gioco	0
Vgioco	1
MsofV	1
DsofV	1
RZERO ENABLE	True
AXIS TYPE	2
VTB AXIS OBJECT	0
PDO NAME	0
STEP CHANNEL	0
STEP NODE	0

Vengono gestite le seguenti funzioni:

Wait_Move – Stato movimento asse

Parametri ***Nessuno***
Ritorna ***1 Asse in movimento***
0 Asse fermi

Move_Axis – Interpolazione lineare

Parametri ***Vel → Velocità assi in millimetri/min***
Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)
Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)
Px → Quota Asse in 0.001 mm
Ritorna ***0 Movimento inserito nel buffer***
1 Buffer pieno (occorre ripetere Move_Axes fino a che il buffer è libero)

Preset – Preset quota Asse

Parametri ***Px → Quota Asse in 0.001 mm per preset***
Ritorna ***Nessuno***

Acc_Axis – Set Accelerazione/decelrazione di posizionamento

Parametri ***Value → Valore Accelerazione di posizionamento in count per TAU***
Ritorna ***Nessuno***

Stop – Stop Movimento

Parametri ***Nessuno***
Ritorna ***Nessuno***

Enable – Abilita controllo Asse

Parametri ***Nessuno***
Ritorna ***Nessuno***

Disable – Disabilita controllo Asse

Parametri ***Nessuno***
Ritorna ***Nessuno***

StartHome – Start ricerca homing Vel in pos1.rzerovel e pos1.rzerovelf

Parametri ***Nessuno***
Ritorna ***Nessuno***

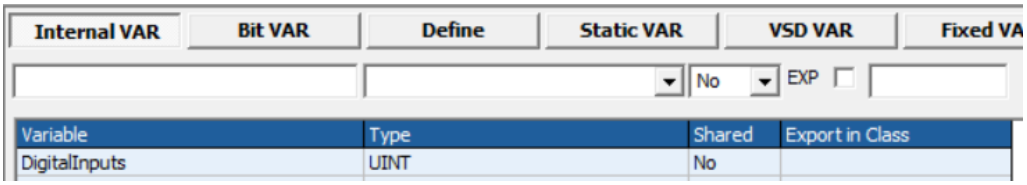
CheckHome – Check stato homing in corso

Parametri ***Nessuno***
Ritorna ***1 homing terminato***

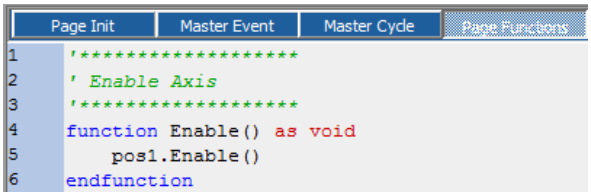
StopHome – Stop homing in corso

Parametri ***Nessuno***
Ritorna ***Nessuno***

Dichiarare le variabili del progetto



Codice in Funzioni di Pagina Main




```
'********************
' Enable Axis
'********************
function Enable() as void
        pos1.Enable()
endfunction
'********************
' Disable Axis
'********************
function Disable() as void
        pos1.Disable()
endfunction
'********************
' Preset Axis
'********************
function Preset(Val as long) as void
        pos1.Preset(Val)
endfunction
'*************************
' Return 1 if axis move
'      0 Axis stop
'*************************
function Wait_Move() as char
        Wait_Move=pos1.move()
endfunction
'********************
' Axis Stop Move
'********************
function Stop() as void
        pos1.Stop()
endfunction
'******************************
' Start Homing
' Homing input see in task plc
'******************************
function StartHome() as void
        pos1.StartHome()
endfunction
```

```
'*****************************
' Check if homing finished
' Return 1 if finished
'*****************************
function CheckHome() as char
        CheckHome=pos1.status_home
endfunction
'*****************************
' Stop home function
'*****************************
function StopHome() as void
        pos1.StopHome()
endfunction
'************************************************************
' Move Axis
' Vel=  vel Axis in mm/min
' Flg if 1 move without buffer
'       0 move in buffer mode
' Px  Axis value in 0.001 mm
'Return 1 if movement is inserted in the buffer
'      0 The movement is not inserted in the buffer
'       in this case, is necessary reload the movement
'************************************************************
function Move_Axis(Vel as long, Flg as char, Px as long) as char
        Vel=Vel*TAU/60 ' Transform in mm/min
        Move_Axis=pos1.moveto(Vel, Flg, Px)
endfunction
'************************************************************
' Set ACC
' Value Acc value in count
'************************************************************
function Acc_Axis(Value as long) as void
        pos1.acc=Value
endfunction
```

Codice in Init Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1   pos1.msof=10000  ' motor 10000 i/rev
2   pos1.ext_fcz=Fc_Home  ' home input
```

pos1.msof=10000 **' motor 10000 i/rev**
pos1.dsof=5000 **' 5 mm per revolution motor**

Codice in Task PLC

TASK PLC Code

Init Task PLC | Task PLC

```
1   DigitalInputs=ng_di(0)  ' read digital inputs
2   pos1.ext_fcz=Fc_Home  ' home input
```

DigitalInputs=ng_di(0) **' read digital inputs**
pos1.ext_fcz=Fc_Home **' home input**

Scarica l' Esempio

15 Gestione Memoria permanente

La scheda NGWARP, mette a disposizione 2 tipologie di memoria permanente:

MEMORIA TIPO STATIC

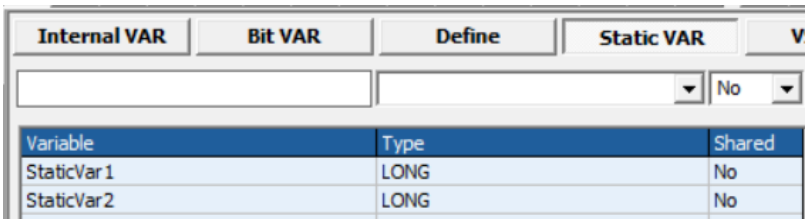
Utilizza una RAM con batteria

MEMORIA TIPO FLASH INTERNA

Utilizza la memoria FLASH di sistema (condivisa con codice applicazione)

15.1 Memoria tipo Static

Questa viene direttamente gestita dal Sistema Operativo, La sua dimensione standard è di 32Kb, ed è resa permanente grazie all' ausilio di una batteria di durata 10 anni circa.
In questo tipo di memoria, possono essere dichiarate delle variabili di sistema, come fossero in RAM, ed utilizzate in modo normale. La particolarità è che il loro valore rimane anche dopo lo spegnimento del sistema.
Per utilizzare questo tipo di memoria è sufficiente dichiarare le Variabili nel campo STATIC:



Queste sono poi utilizzate come normali variabili

15.2 Gestione Memoria FLASH Interna

La memoria FLASH interna, è condivisa con il codice dell' applicazione.
Normalmente sono disponibili 4 Mb di memoria interna, difficilmente le applicazioni NGWARP usano oltre 1 Mb di codice, pertanto quasi sicuramente 3 Mb e oltre rimangono liberi per il salvataggio dei dati applicazione.

15.2.1 IMS_READ – Lettura memoria flash

Legge da Memoria interna i dati a partire da ADDR, per una lunghezza di NBYTE e l' inserisce nel vettore puntato da PUNT.

Sintassi

IMS_READ(Punt as *char, Addr as long, Nbyte as long) as char

Parametri

Punt	Puntatore al buffer di dati di lettura
Addr	Indirizzo di partenza della MEMORIA
Nbyte	Numero di byte da leggere

Valore di ritorno

Char	0	Nessun errore
	<>0	Errore lettura

15.2.2 IMS_WRITE – Scrittura memoria flash

Scrive nella FLASH interna all'indirizzo indicato da ADDR, i dati indirizzati dal puntatore PUNT per un totale di NBYTE dati.
La FLASH viene gestita a BLOCCHI di 256 Bytes, pertanto si consiglia sempre di scrivere a MULTIPLI di 256 come lunghezza dati. Questo perché se vengono scritti meno di 256 Bytes, viene comunque cancellato l'intero BLOCCO, per non perdere i dati occorrerebbe quindi leggere prima tutto il blocco, salvare i dati interessati e sovrascrivere il tutto.

Sintassi

IMS_WRITE(Punt as *char, Addr as long, Nbyte as long) as char

Parametri

Punt	Puntatore al buffer di dati
Addr	Indirizzo di partenza della MEMORIA
Nbyte	Numero di byte da scrivere

Valore di ritorno

Char	0	Nessun errore
	<>0	Errore scrittura

ATTENZIONE
NELLA FLASH VIENE SEMPRE SCRITTO UN MULTIPLO DI 256 BYTES QUINDI SE IL NUMERO DI BYTES DA SCRIVERE E' MINORE, VERRANNO SCRITTI DEI VALORI CASUALI

15.3 Esempio save/load in FLASH di valori in un vettore

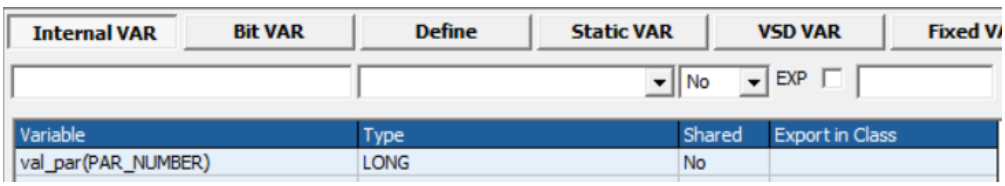
Nell' esempio riportato, vengono salvati e caricati in FLASH i valori contenuti in un vettore di LONG. Questo può essere utilizzato come gestione di parametri macchine.
Viene gestita un CheckSum (somma dei valori dei parametri) e salvata nell' ultima posizione del vettore. Questa garantisce l' integrità dei valori.

Vengono gestite le seguenti funzioni:

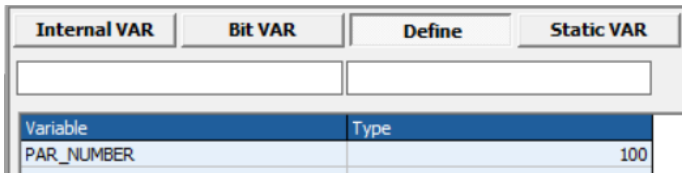
LoadPar – Carica i valori da FLASH
Parametri ***Nessuno***
Ritorna ***0 OK***
1 Errore FLASH
2 Errore checksum valori

SavePar – Salva i valori da FLASH
Parametri ***Nessuno***
Ritorna ***0 OK***
1 Errore FLASH

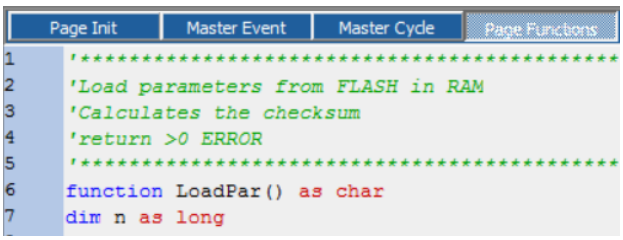
Dichiarare le variabili del progetto



Dichiarare le DEFINE del progetto



Codice in Funzioni di Pagina Main




```
'*************************************************
'Load parameters from FLASH in RAM
'Calculates the checksum
'return >0 ERROR
'*************************************************
function LoadPar() as char
dim n as long
```

```
dim ckl as long
dim ck as long
dim Ret as char
'PAR_NUMBER is number of parameters
'all parameters are in long
Ret=ims_read(val_par(),0,PAR_NUMBER*4) ' reads parameters from flash and 'puts in val_par vector



if Ret<>0
        'LOAD ERROR !!!!
        LoadPar=1 'return ERROR 1
        return
endif
ck=val_par(PAR_NUMBER) 'gets the check sum in last position
ckl=0
for n=0 to n<(PAR_NUMBER-1) 'calculates the checksum
        ckl=ckl+val_par(n)
next n
if ckl=0            'if all parameters are ZERO - chekcsum error
        ckl=ck+1
endif
if ckl<>ck
        'CheckSum ERROR
        LoadPar=2 'return ERROR 2
else
        LoadPar=0 'return OK
endif
endfunction

'*********************************
'Save the parameters in FLASH
'Return >0 ERROR
'*********************************
function SavePar() as char
dim ck as long
dim n as long
dim Ret as char
ck=0
for n=0 to n<(PAR_NUMBER)-1 'calculates the checksum
        ck=ck+val_par(n)
next n
val_par(PAR_NUMBER-1)=ck 'put the checksum
Ret=ims_write(val_par(),0,PAR_NUMBER*4) 'save the parameters
if Ret<>0
        'SAVE ERROR !!!!
        SavePar=1 'return ERROR 1
else
        SavePar=0 'return OK
endif
endfunction
```

Scarica l' Esempio

16 System Utility

La Scheda NGWARP mette a disposizione delle utiltiy presenti nel sistema operativo.
Queste possono essere richiamate da applicazione VTB tramite la funzione **System_Utility(…)**

16.1 User Led

Normalmente lo stato dei LED ST1-ST2-ST3 viene gestito dal sistema operativo Interno:

ST1 – Ser2 – CanOpen Slave
ST2 – Ethercat
ST3 – TaskPlc Time Burst

E' possibile gestire manualmente lo stato di questo led tramite la funzione System_Utility(…)

System_Utility(60,0,0,0)	→	Ritorna la gestione dei Led al sistema operativo
System_Utility(61,State,0,0)	→	**State=1** Accende il LED **ST1** **State=0** Spenge il LED **ST1**
System_Utility(62,State,0,0)	→	**State=1** Accende il LED **ST2** **State=0** Spenge il LED **ST2**
System_Utility(63,State,0,0)	→	**State=1** Accende il LED **ST3** **State=0** Spenge il LED **ST3**

Una volta attivata la funzione **61,62,63** il relativo LED non viene più gestito dal sistema operativo.
Per ritornare all normale Gestione, chiamare la funzione **60**.

16.2 Lettura dimensione memoria di Salvatggio IMS

Questa funzione riporta ilo numero di Bytes disponibili nella memoria IMS.
Questo valore si riferisce a Bytes totali e non a quelli liberi o occupati

Long Bytes=*System_Utility(101,0,0,0)* → Ritorna Il numero di bytes della memoria IMS

16.3 Abilitazione Canali Analogici a 10/12 bit

Normalmente per compatibilità delle applicazioni NG35, NGWARP inizializza i Canali analogici 1- 8 a 10 BIT di risoluzione (0-1023).
Tramite utility di sistema è però possibile abilitare la massima risoluzione disponibile 12 BIT (0-4095)

System_Utility(132,0,0,0)	→	Abilita i canali analogici a 10 bit
System_Utility(132,1,0,0)	→	Abilita i canali analogici a 12 bit

Non esiste una Utility per riportare i Canali Analogici a 10 bit, pertanto è necessari togliere la chiamata **132** alle utility dall' applicazione VTB

16.4 Lettura Tempo Utilizzo Task Plc

Con questa funzione è possibile leggere il Tempo di ciclo del TASK PLC
Il valore che ritorna sono i Microsecondi dell' effettivo tempo di ciclo del TASK PLC, in pratica misura la deviazione effettiva del Task Plc rispetto a quella impostata.

Long uSec = System_Utility(1601,0,0,0) → Legge il tempo effettivo

Il tempo letto, può non essere sempre lo stesso, in quanto questo dipende dal ciclo del codice del task plc

16.5 Lettura Can Synk Time

Con questa funzione è possibile leggere il Tempo (in microsecondi) di SYNK impostato per il CanOpen

Long uSec = System_Utility(1602,0,0,0) → Legge il tempo di SYNK

16.6 Lettura DC Ethercat Synk Time

Con questa funzione è possibile leggere il Tempo (in nanosecondi) del DC per Ethercat

Long nSec = System_Utility(1703,0,0,0) → Legge il tempo di DC

16.7 Lettura Errore DC Ethercat Synk Time

Con questa funzione è possibile leggere l' errore (in nanosecondi) del DC per Ethercat

Long nSec = System_Utility(1706,ID,0,0) → Legge l' errore di DC

ID=0 → ***Lettura Master Sync Error***
ID=n → ***Lettura slave "n" Sync Error dove "n" è il numero dello slave***

Sommario