NG QUARK

www.promax.it

VTB Software Resources

Rev. 1.0.1

© 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 NGQ 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 RS232/RS485

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

2.1 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

2.2 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)

2.3 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

2.4 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

2.5 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

2.6 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

2.7 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

2.8 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 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

2.9 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

2.10 Esempio

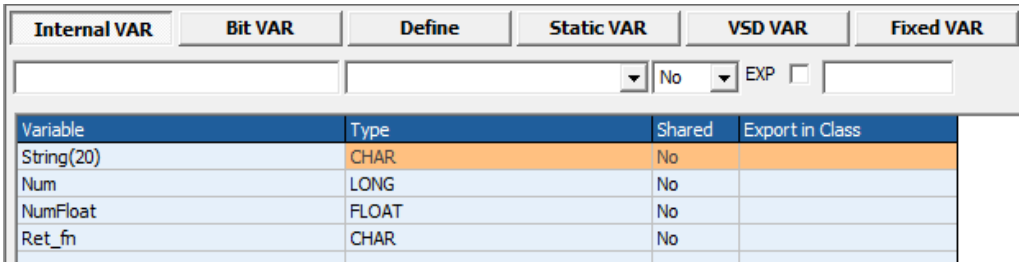
Nell' esempio riportato, viene chiamata la funzione Read_Data() in polling dal task main.
Seriale SER2 programmata con i seguenti parametri:

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

Questa controlla se ricevuto un carattere ed effettua un risposta:

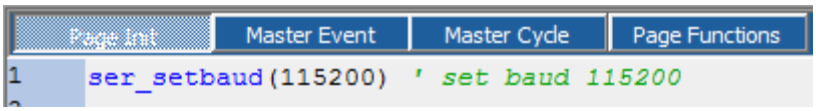
Carattere ricevuto=1 → ***Eco del carattere stesso (1) con ser_putchar***
Carattere ricevuto=2 → ***Invio testo "Test String" con Ser_puts***
Carattere ricevuto=3 → ***Print formattato di variabile Num (Long numero caratteri ricevuti)***
Carattere ricevuto=4 → ***Print formattato di variabile NumFloat (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



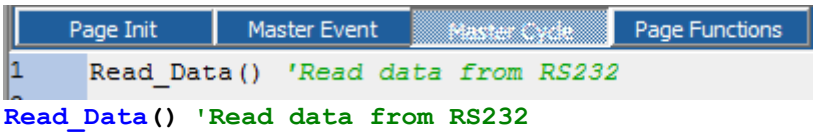
Variable	Type	Shared	Export in Class
String(20)	CHAR	No	
Num	LONG	No	
NumFloat	FLOAT	No	
Ret_fn	CHAR	No	

Codice in Init Main



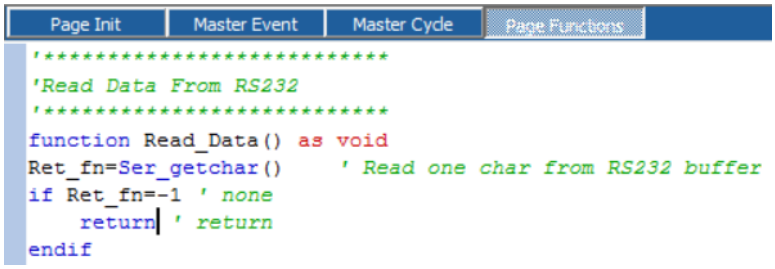

```
ser_setbaud(115200) ' set baud 115200
```

Codice in Master Ciclo Main




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

Codice in Funzioni di Pagina Main




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

3 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

3.1 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

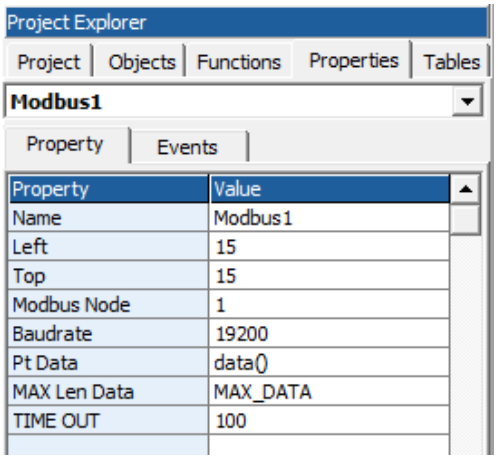
3.2 Esempio ModBus slave

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 :

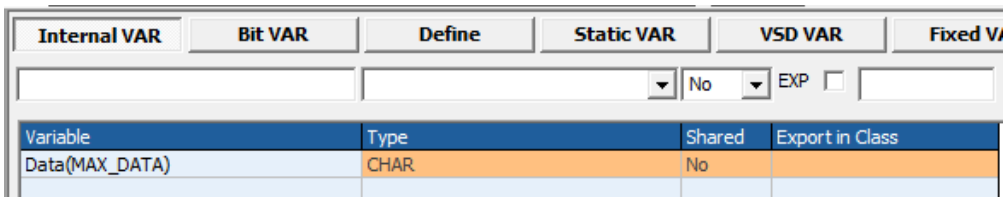
Lettura/Scrittura da Modbus registri Nr.1 → Data(0)
Lettura/Scrittura da Modbus registri Nr.2 → Data(1)
ecc.
L' esempio legge il registro 2 - Data(1) e scrive nel registro 1 Data(0)

Oggetti utilizzati:

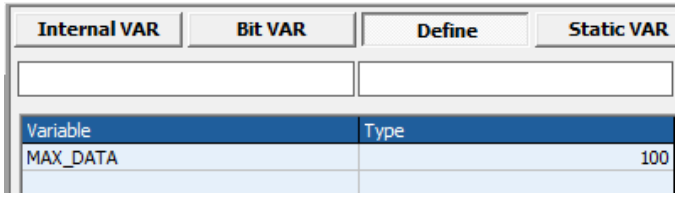
Modbus → Cmodbus → ModBus Protocol



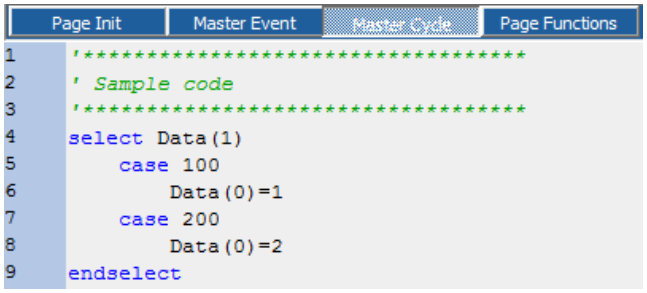
Dichiarare le variabili del progetto



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

3.3 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

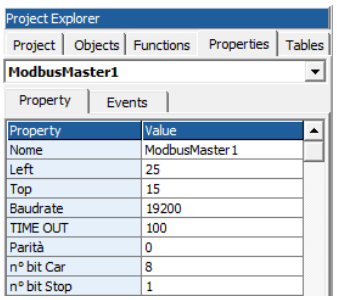
0	Scrittura OK
1	Risposta errata dallo slave
2	Time Out
3	lunghezza dati > di 127
4	errore checksum

3.4 Esempio ModBus Master

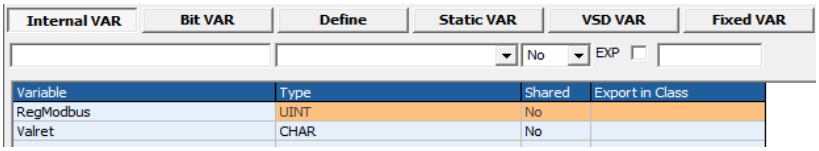
Nell' esempio riportato, vengono scritti e letti dei registri 16 bit di uno slave al nodo

Oggetti utilizzati:

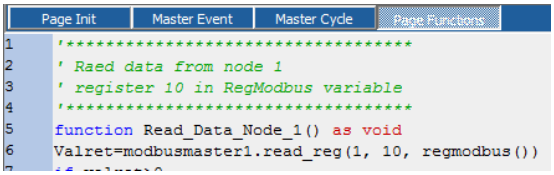
Modbus → CmodbusMaster → ModBus Master Protocol



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

4 Lettura Ingressi Analogici

La CPU NGQ integra 4 ingressi analogici disponibili da funzioni VTB.
Attualmente gli ingressi sono a 12 bit (valore da 0 a 4096)

4.1 Lettura Ingressi

Sintassi

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

Parametri

Channel Numero del canale (da 0 a 3)

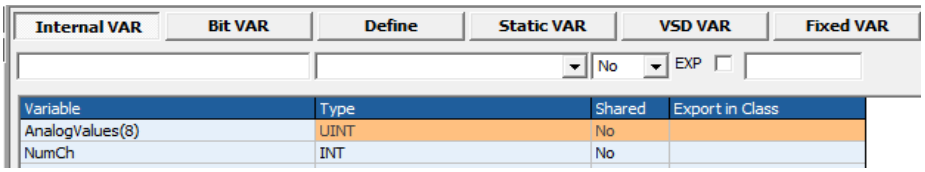
Valore di ritorno

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

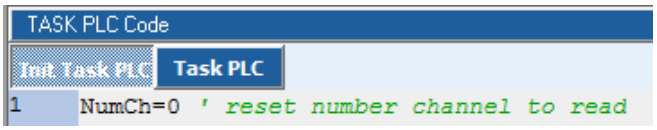
4.2 Esempio Lettura canali analogici

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

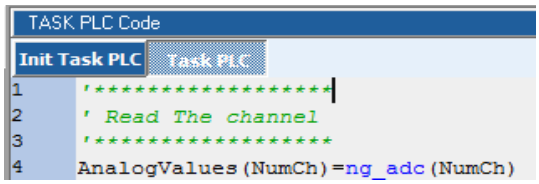


Codice su Init Task Plc




```
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=4 ' limit
      NumCh=0
endif
```

Scarica l' Esempio

5 Gestione CanOpen

Nella scheda NGQ, è presente una Linea CanOpen configurabile Master o Slave, a seconda del Firmware utilizzato. 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

5.1 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

5.2 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

5.3 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
```

5.4 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

5.5 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

5.6 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**				

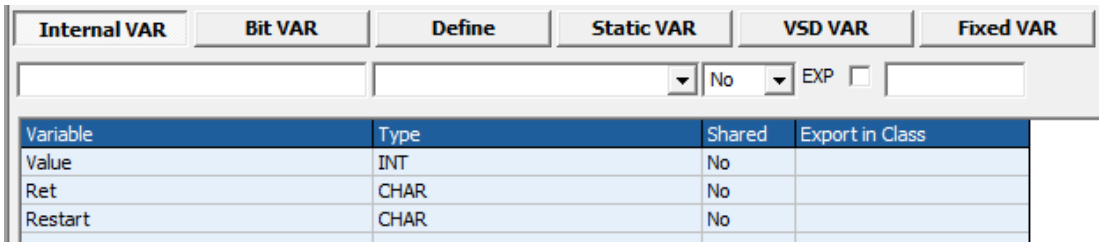
ATTENZIONE

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

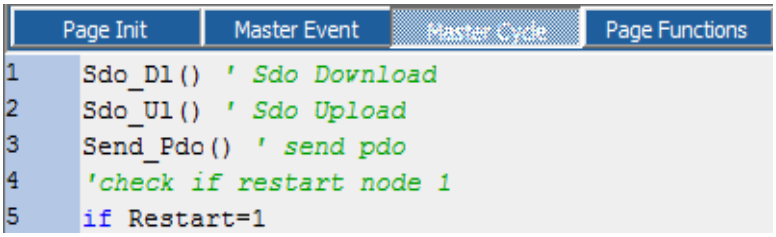
5.7 Esempio Utilizzo delle funzioni CanOpen

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

Dichiarare le variabili del progetto



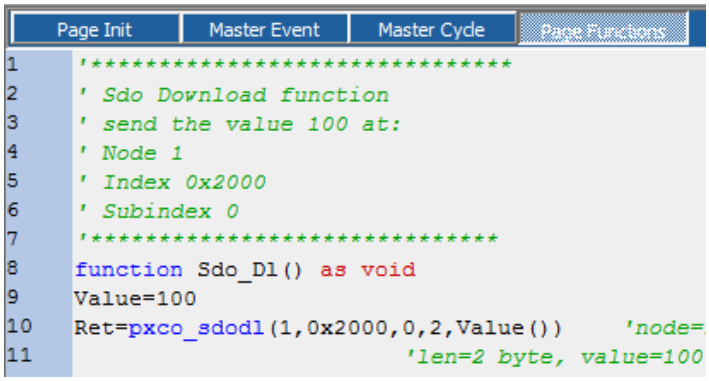
Inserire il seguente codice nella Master Ciclo – Main




```
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

5.8 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***

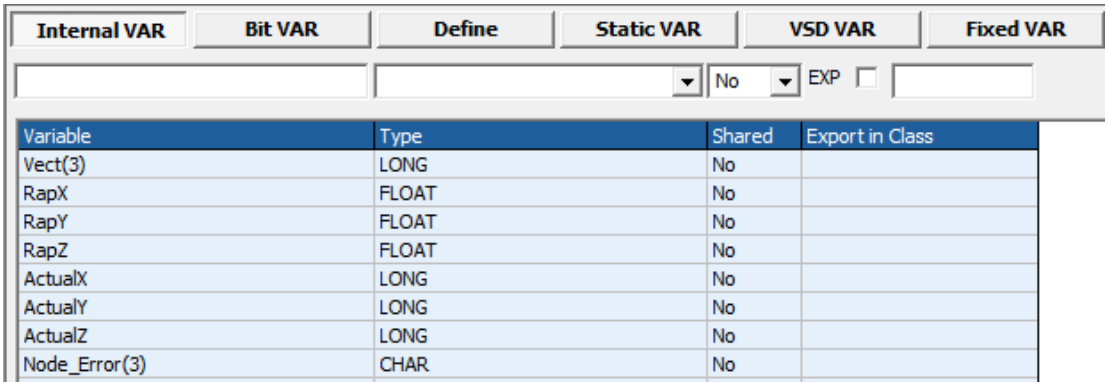
Stop_Axes – Stop Movimento
Parametri ***Nessuno***
Ritorna ***Nessuno***

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***

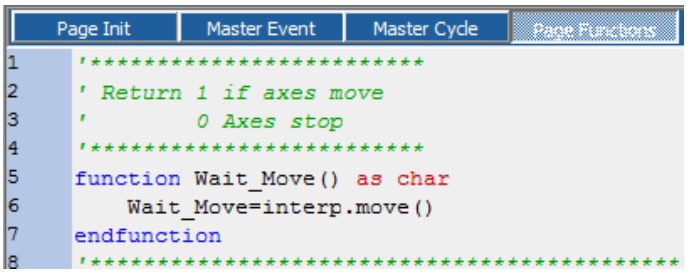
Close_cancfgerr - Errori Custom CanOpen questa funzione viene chiamata alla fine della scansione di tutti i nodi presenti in configurazione
Parametri ***Nessuno***
Ritorna ***Nessuno***
Open_cancfgerr - Errori Custom CanOpen questa funzione viene chiamata quando inizia la fase di inizializzazione di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

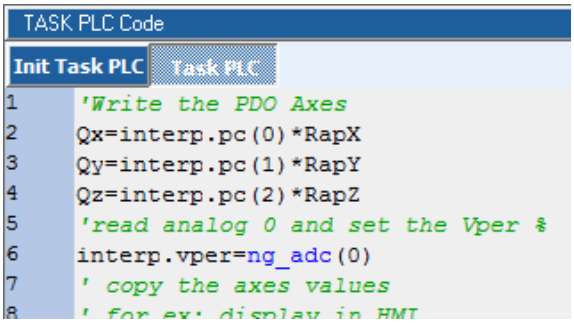
TASK PLC Code

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




```
'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

5.9 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

Project Explorer

Project | Objects | Functions | Properties

CanPos1

Property | Events

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***

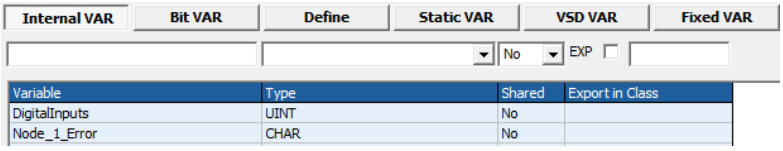
Stop – Stop Movimento
Parametri ***Nessuno***
Ritorna ***Nessuno***

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

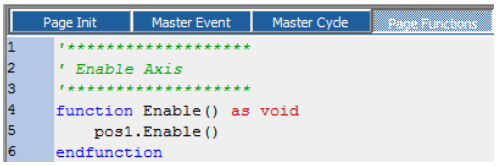
Disable – Disabilita controllo Asse
Parametri ***Nessuno***
Ritorna ***Nessuno***
StartHome – Start ricerca homing Vel in pos1.rzerovel e pos1.rzerovelf
Parametri ***Nessuno***
Ritorna ***Nessuno***

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

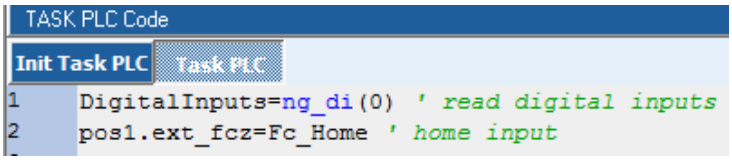
TASK PLC Code

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




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

Scarica l' Esempio

6 I/O Digitali

Nella scheda NGQ sono presenti 11 ingressi digitali e 8 uscite digitali gestibili da funzioni VTB.

6.1 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 10

Ingresso	11	10	9	8	7	6	5	4	3	2	1
Bit	10	9	8	7	6	5	4	3	2	1	0

Sintassi

NG_DI(CardNumber **as Char**) **as uint**

Parametri

CardNumber 0

Valore di ritorno

Uint Valore degli 11 ingressi gestiti a BIT
bit = 1 → ingresso ON
bit = 0 → ingresso OFF

6.2 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 7

Uscita	8	7	6	5	4	3	2	1
Bit	7	6	5	4	3	2	1	0

Sintassi

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

Parametri

CardNumber 0

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

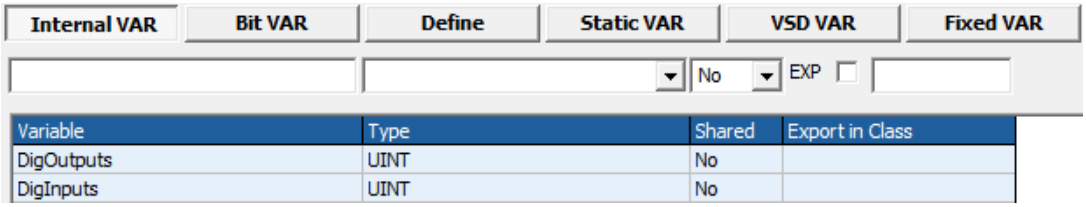
6.3 Esempio Utilizzo I/O Digitali

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

AGGIORNAMENTO DELLO STATO DEGLI INGRESSI E USCITE FATTO NEL TASK PLC

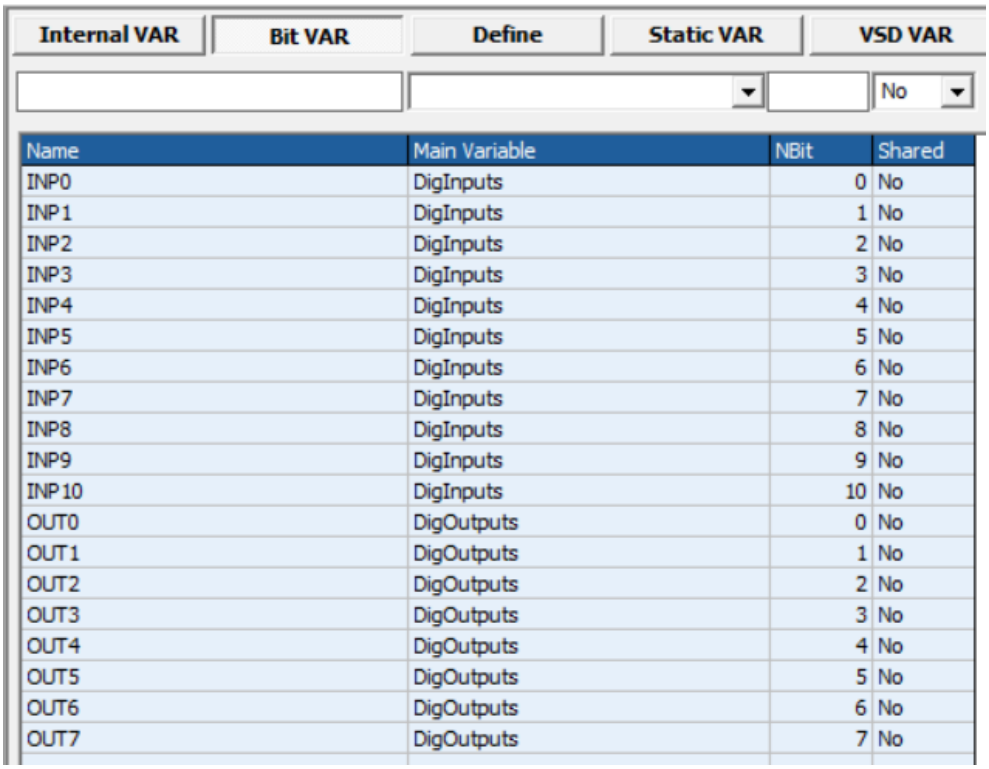
Gestione a bit delle I/O. I primi TRE ingressi ricopiano lo stato sulle prime 3 Uscite

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
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

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

7 Uscite analogiche

Sulla scheda NGQ possono essere montate 2 uscite analogiche +/- 10 V 12 bit.

7.1 NG_DAC - SCRITTURA USCITE ANALOGICHE

Questa funzione permette di pilotare l'uscita analogica di ogni canale gestito dal controllo NGQ (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 1,

Il numero massimo di canali analogici gestiti è di 2

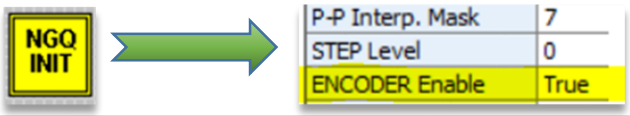
Sintassi

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

Parametri

Channel Numero Canale (da 0 a 1)
val Valore dell'uscita (da -2047 a +2048)

ATTENZIONE
LE USCITE ANALOGICHE SI TROVANO SU UN PICCOLA SCHEDA DI ESPANSIONE. PER ABILITARE QUESTE USCITE ANALOGICHE, OCCORRE ABILITARE LA PROPRIETÀ ENCODER ENABLE A TRUE DELL' OGGETTO NGQ INIT



7.2 NG_DAC_CAL - OFFSET USCITE ANALOGICHE

Questa funzione permette di calibrare l' OFFSET dell'uscita analogica di ogni CANALE gestito dal controllo NGQ. 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

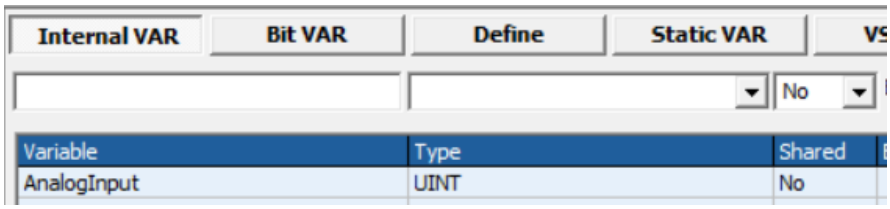
Channel Numero Canale (da 0 a 1 **)**
Offset Valore dell' OFFSET (da -2047 a +2048)

ATTENZIONE
IL VALORE DELL'OFFSET NON RIMANE MEMORIZZATO, PERTANTO AD OGNI ACCENSIONE OCCORRE IMPOSTARLO DI NUOVO
SALVARE IL VALORE OFFSET IN MEMORIA FRAM NGQ

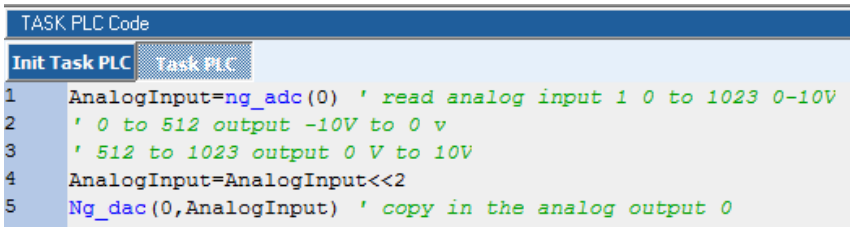
7.3 Esempio Utilizzo Uscite Analogiche

Nell' esempio riportato vengono gestite le uscite Analogiche
Nell' esempio, viene letto il canale analogico 0 e riportato sull' uscita analogica 0.

Dichiarare le variabili del progetto



Codice in Task PLC




```
AnalogInput=ng_adc(0) ' read analog input 1 0 to 1023 0-10V
' -2047 0 output -10V to 0 v
' 0 2048 output 0 V to 10V
AnalogInput=AnalogInput-2048
Ng_dac(0,AnalogInput) ' copy in the analog output 0
```

Scarica l' Esempio

8 Gestione canali STEP su NGQ

La scheda NGQ porta 4 canali STEP/DIR sulla CPU e fino a 6 canali STEP/DIR su espansione NGMsX. La differenza tra i canali su NGQ e NGMsX è nella frequenza massima uìin uscita:

NGQ clock position mode 125 Khz totali
NGQ clock interpolation mode 30 Khz totali

8.1 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
Value	Valore assoluto della posizione dell'asse step/dir

ATTENZIONE
PER UN CORRETTO FUNZIONAMENTO INSERIRE QUESTA FUNZIONE NEL TASK_PLC

8.2 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 da 0 a 3
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

8.3 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*

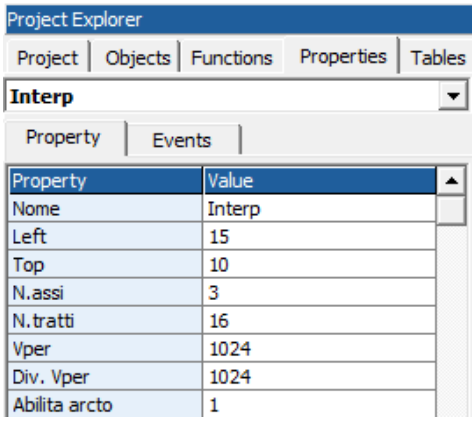
8.4 Esempio Utilizzo Assi Interpolati STEP/DIR

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

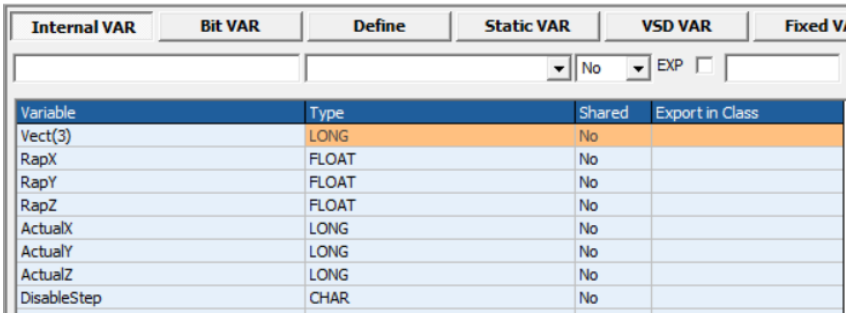
Move_Axes – Interpolazione lineare
Parametri ***Vel → Velocità assi in millimetri/min***
Flg → Impostare ad 1 per disabilitare il buffer movimenti (fermata sul tratto)
Impostare a 0 per abilitare buffer movimenti (fermata sullo spigolo >SGLP)
Px,Py,Pz → Quote Assi in 0.001 mm
Ritorna ***0 Movimento inserito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

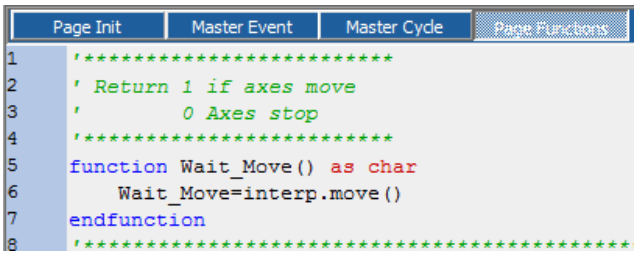


Internal VAR | Bit VAR | Define | Static VAR | VSD VAR | Fixed V

No | EXP

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



Page Init | Master Event | Master Cycle | Page Functions

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

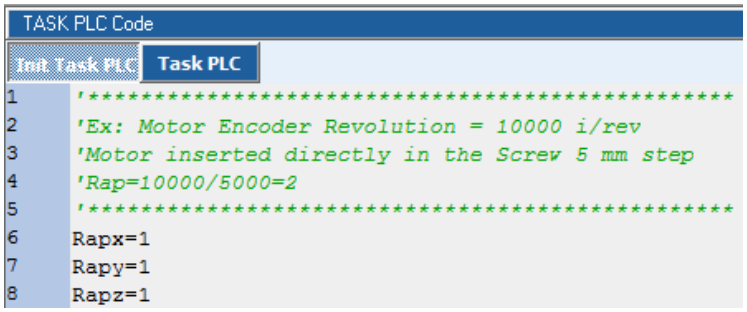
```
'************************
' Return 1 if axes move
'        0 Axes stop
'************************
function Wait_Move() as char
      Wait_Move=interp.move()
endfunction

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

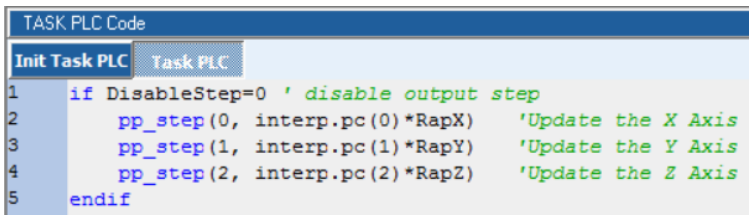
```
'*************************************************************
' Stop Axes
'*************************************************************
function Stop_Axes() as void
      interp.stop()
endfunction
'*******************
' Axis X enable
'*******************
function Enable_X() as void
DisableStep=1
'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

8.5 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***

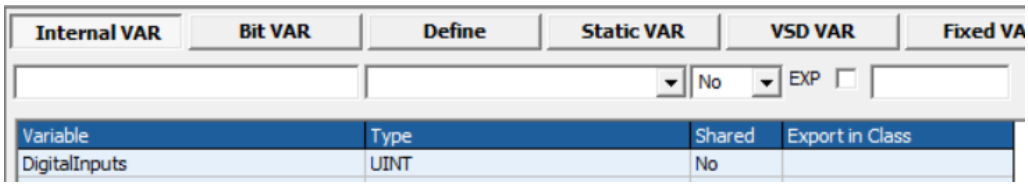
CheckHome – Check stato homing in corso

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

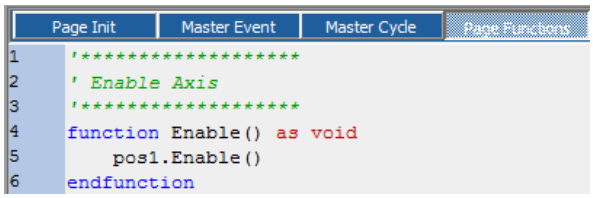
StopHome – Stop homing in corso

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

9 Gestione Memoria permanente

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

Interna tipo FLASH 1024 Bytes
Esterna tipo FRAM 16 Kb (opzionale)

La differenza tra le due memorie è sostanziale:

La memoria interna deve essere scritta con un unico blocco di 1024 Bytes
La memoria esterna può essere gestita a Byte

La selezione tra memoria interna ed esterna viene automaticamente effettuata adll' indirizzo di LETTURA o SCRITTURA utilizzato:

Addr da 0 a 1023 Memoria Interna FLASH
Addr da 1024 a 17407 Memoria esterna FRAM

9.1 Gestione Memoria permanente

La memoria permanente , può avere una dimensione di 2 o 16 Kb (seconda dell' opzione scelta)

9.1.1 IMS_READ – Lettura memoria permanente

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

9.1.2 IMS_WRITE – Scrive nella memoria permanente

Scrive nella FRAM interna all'indirizzo indicato da ADDR, i dati indirizzati dal puntatore PUNT per un totale di NBYTE dati.

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
LA MEMORIA ESTERNA FRAM, PUO' ESSERE GESTITA BYTE PER BYTE
LA MEMORIA INTERNA FLASH VIENE GESTITA A BLOCCHI DI 1024 BYTES

9.2 Esempio save/load in FRAM ESTERNA di valori in un vettore

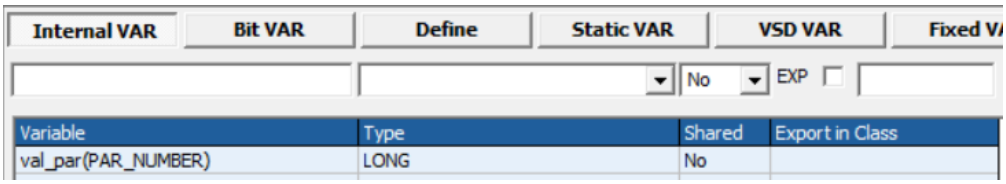
Nell' esempio riportato, vengono salvati e caricati in FRAM i valori contenuti in un vettore di LONG. Questo può essere utilizzato come gestione di parametri macchine.
Viene gestita un CheckSum (somma dei valori dei parametri) e salvata nell' ultima posizione del vettore. Questa garantisce l' integrità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

Vengono gestite le seguenti funzioni:

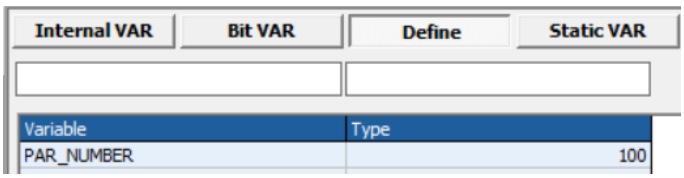
LoadPar – Carica i valori da FRAM
Parametri ***Nessuno***
Ritorna ***0 OK***
1 Errore FRAM
2 Errore checksum valori

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

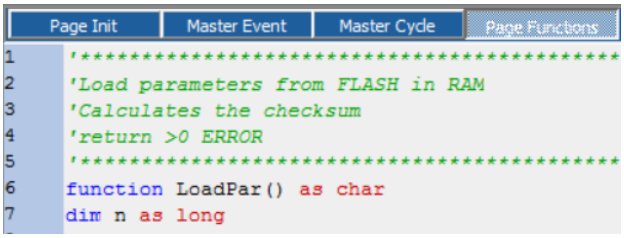
Dichiarare le variabili del progetto



Dichiarare le DEFINE del progetto



Codice in Funzioni di Pagina Main




```
'*****************************************************
'Load parameters from FRAM 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(),1024,PAR_NUMBER*4) ' reads parameters from FRAM
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 FRAM
'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(),1024,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

ATTENZIONE

NELL' ESEMPIO RIPORTATO VIENE UTILIZZATO UN INDIRIZZO DI PARTENZA

1024

QUESTO DEFINISCE DI UTILIZZARE LA MEMORIA ESTERNA DI TIPO FRAM

Sommario