

# IBD e SBD drives con function block in TIA portal

## Application note

Doc. TR512003  
Ed. 1.2 - Italiano



### Note legali

CMZ SISTEMI ELETTRONICI S.r.l. si riserva il diritto di apportare modifiche ai prodotti descritti in questo documento in qualsiasi momento e senza preavviso.

Il presente documento è stato preparato da CMZ SISTEMI ELETTRONICI S.r.l. esclusivamente per l'uso da parte dei propri clienti garantendo che esso costituisce, alla data di edizione, la documentazione più aggiornata relativa ai prodotti.

È inteso che l'uso della documentazione avviene da parte dell'utente sotto la propria responsabilità e che l'utilizzo di certe funzioni descritte in questo manuale, deve essere fatto con la dovuta cautela in modo da evitare pericolo per il personale e danneggiamenti alle macchine.

Nessuna ulteriore garanzia viene pertanto prestata da CMZ SISTEMI ELETTRONICI S.r.l., in particolare per eventuali imperfezioni, incompletezze e/o difficoltà operative.

Questo documento contiene informazioni confidenziali che sono di proprietà di CMZ SISTEMI ELETTRONICI S.r.l.. Né il documento né le informazioni in esso contenute devono essere divulgate o riprodotte in tutto o in parte, senza consenso scritto da parte di CMZ SISTEMI ELETTRONICI S.r.l..

1. Introduzione .....	2
2. Importazione FBs e struttura dati .....	2
3. Struttura Telegram200 .....	4
4. Esempi .....	5
5. Function blocks .....	9
5.1. Gestione dati ciclici e diagnostica dell' asse .....	9
5.2. Abilitazione .....	16
5.3. Reset .....	18
5.4. Homing .....	20
5.5. Movimenti .....	22
5.6. Stop .....	35
5.7. Ingressi e uscite digitali .....	39

## 1. Introduzione

Questo documento offre delle indicazioni su ciò che CMZ offre per la gestione degli azionamenti IBD e SBD PROFINET, tramite function blocks implementati utilizzando l'ambiente TIA portal e su come utilizzarli nel progetto.

CMZ offre due file da importare nel progetto per comandare gli azionamenti tramite function blocks:

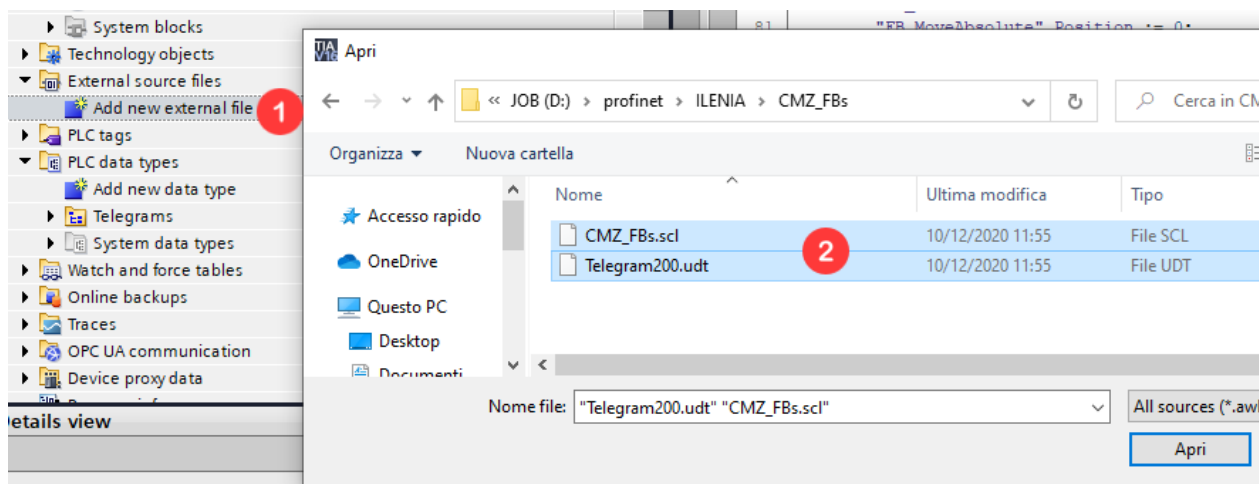
- CMZ\_FBs.scl : contenente i function blocks implementati per gestire l'azionamento.
- Telegram200.udt : contenente il tipo di dato utilizzato come IN/OUT di tutti i function blocks.

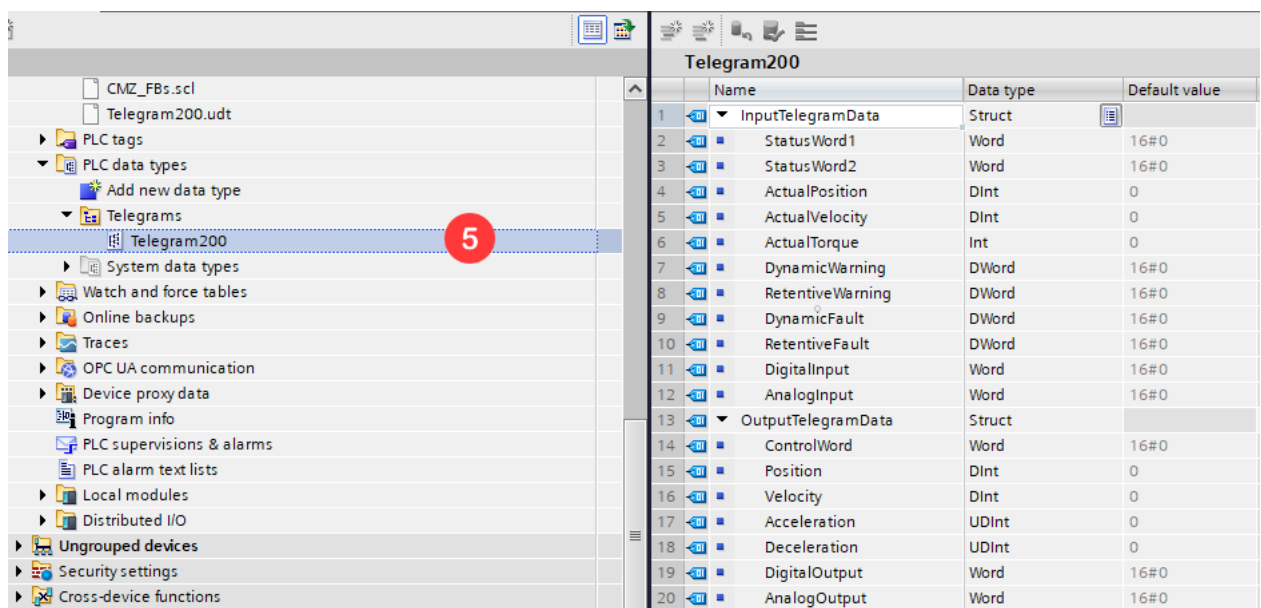
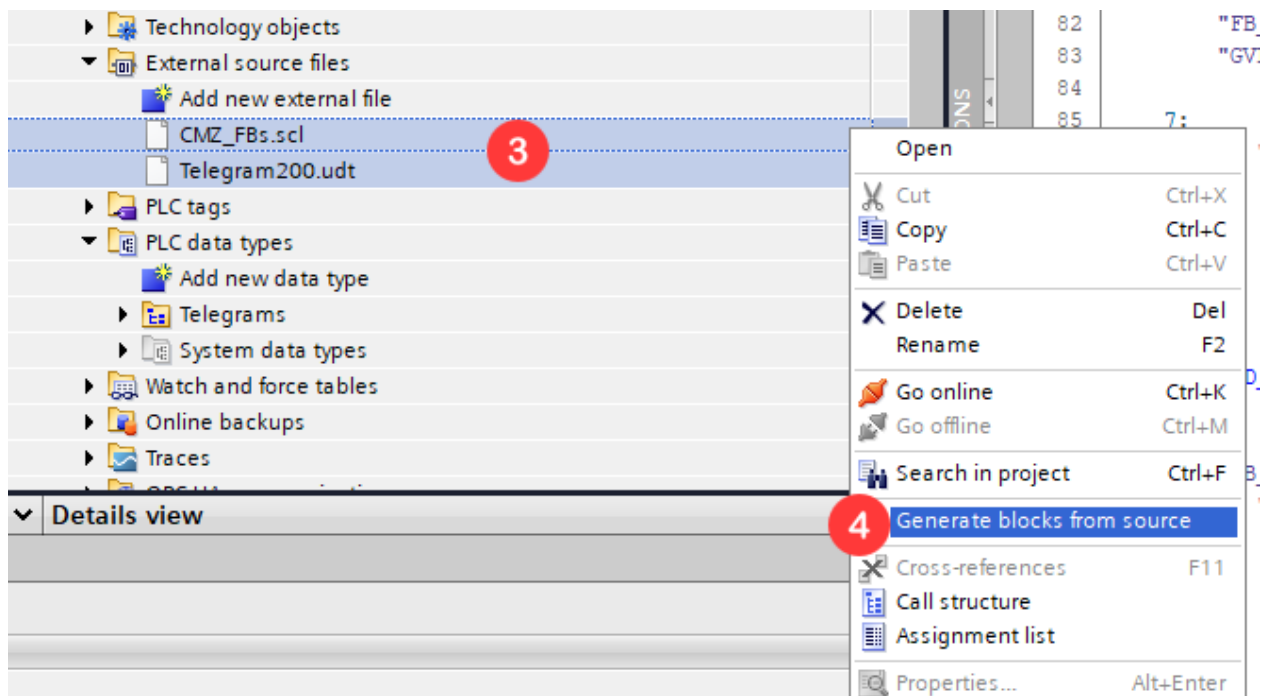
I capitoli in questo manuale spiegano gli step da seguire per utilizzare correttamente i function blocks in un progetto, quindi è necessario:

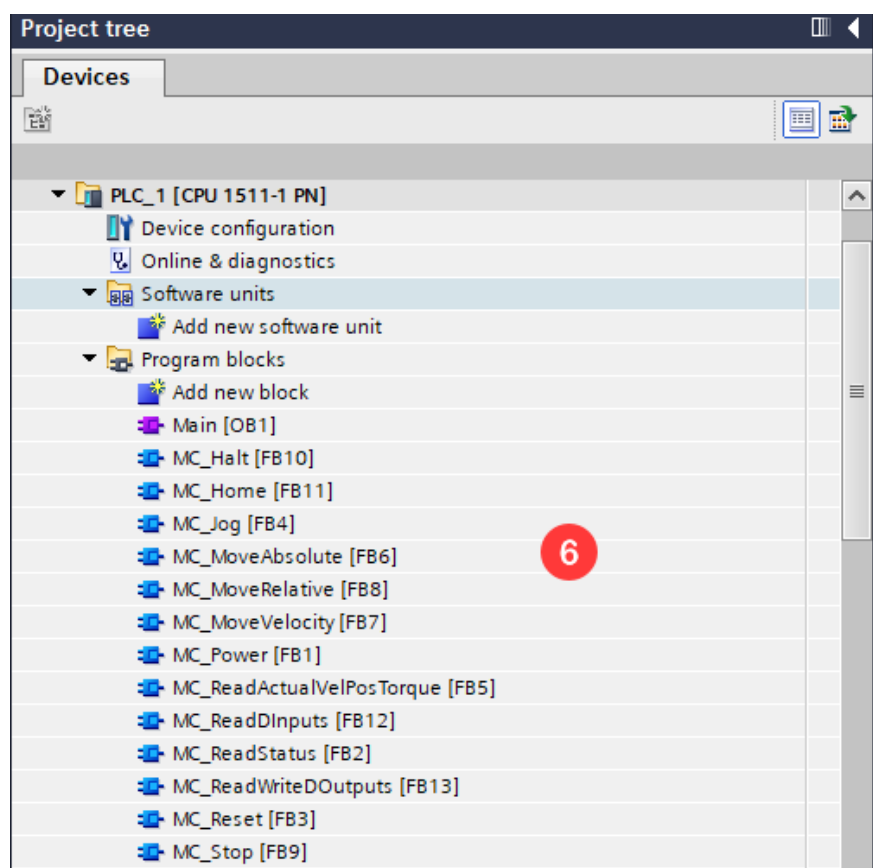
1. Importare nel progetto i function blocks tramite il file CMZ\_FBs.scl.
2. Importare nel progetto la struttura dati tramite il file Telegram200.udt.
3. Istanziare ed utilizzare correttamente nel programma i function blocks che servono nell'applicazione, seguendo quanto riportato nella [Sezione 4, «Esempi»](#), passando come IN/OUT di tutti i function block la struttura dati.

## 2. Importazione FBs e struttura dati

Per importare nel progetto i function blocks e la struttura da utilizzare, implementati da CMZ, che permettono di comandare l'azionamento è necessario seguire i passaggi qui sotto descritti:







- ❶ Dall'albero del progetto premere su *Add new external file* presente nella sezione *External source file*.
- ❷ Selezionare il function block (.scl) e la struttura (.udt) da importare nel progetto e premere *Apri*.
- ❸ Importato il file contenente i function blocks e il file contenente la struttura, selezionarli e fare tasto destro su di essi.
- ❹ Generare i blocchi FB e l'oggetto struttura da utilizzare poi nel programma, premendo su *Generate blocks from source*.
- ❺ Nella sezione *PLC data types* si potrà trovare la struttura *Telegram200* importata tramite il file *Telegram200.udt*.
- ❻ Nella sezione *Program blocks* si potranno trovare i function blocks importati tramite il file *CMZ\_Fbs.scl*.

### 3. Struttura Telegram200

La struttura dati implementata da CMZ si chiama *Telegram200* ed è importabile nel progetto tramite il file esterno *Telegram200.dut* distribuito da CMZ.

Questa struttura permette di gestire e comandare l'azionamento e contiene la struttura del telegramma 200 con il frame di input (Device → Controller) e il frame di output (Controller → Device).

Questa struttura dopo averla importata nel progetto viene gestita dal function block *MC\_ReadStatus* che permette di leggere e scrivere i dati ciclici tramite le istruzioni *DPRD\_DAT* e *DPWR\_DAT* e successivamente viene utilizzata come IN/OUT di tutti i function blocks implementati.

La struttura è la seguente:

```

TYPE
Telegram200                : STRUCT
  InputTelegramData         : STRUCT;
    statusWord1             : WORD;
    statusWord2             : WORD;
    actualPosition           : DINT;
    actualVelocity           : DINT;
    actualTorque             : INT;
    dynamicWarning          : DWORD;
    retentiveWarning        : DWORD;
    digitalInput            : WORD;
    analogInput             : WORD;
  OutputTelegramData        : STRUCT;
    controlWord             : WORD;
    position                : DINT;
    velocity                : DINT;
    acceleration            : DINT;
    deceleration            : INT;
    digitalOutput           : WORD;
    analogOutput            : WORD;
END_STRUCT;
END_TYPE

```

## 4. Esempi

In questo capitolo è presente un esempio che mostra come utilizzare i function blocks e la struttura dati in un programma, dopo averli importati nel progetto.

Questo programma di esempio permette di: mettere in coppia l'asse se non è in errore, eseguire la procedura di homing, eseguire un movimento in velocità con una velocità target, eseguire dei posizionamenti assoluti a quote assolute e alla fine un posizionamento relativo.

```

//istanziamento function block MC_ReadStatus
"FB_ReadStatus"(Enable:=TRUE, pHwTelegram200 := "IBD60-
PNT~DO_with_CMZ_telegram_200_1~CMZ_telegram_200");

```

```
//istanziamento function block MC_Power
"FB_Power"(Telegram_REF := "FB_ReadStatus".Telegram_REF,
           Enable := TRUE);

//istanziamento function block MC_Home
"FB_Home"(Telegram_REF := "FB_ReadStatus".Telegram_REF);

//istanziamento function block MC_MoveRelative
"FB_MoveRelative"(Telegram_REF := "FB_ReadStatus".Telegram_REF);

//istanziamento function block MC_MoveAbsolute
"FB_MoveAbsolute"(Telegram_REF := "FB_ReadStatus".Telegram_REF);

//istanziamento function block MC_Stop
"FB_Stop"(Telegram_REF := "FB_ReadStatus".Telegram_REF);

//istanziamento function block MC_Halt
"FB_Halt"(Telegram_REF := "FB_ReadStatus".Telegram_REF);

//istanziamento function block MC_MoveVelocity
"FB_MoveVelocity"(Telegram_REF := "FB_ReadStatus".Telegram_REF);

CASE "GVL".StepTest OF
  0:
    ;
  1:
    IF (NOT "FB_ReadStatus".ErrorStop) AND "FB_ReadStatus".Active
    THEN
      //if FB read status is active and axis isn't in errorstop
      //enable axis
      "FB_Power".Power := TRUE;
      "FB_Home".Execute := FALSE;
      "GVL".StepTest := "GVL".StepTest + 1;
    END_IF;
  2:
    IF "FB_Power".Status THEN
      //if axis is enabled
      //execute homing procedure
      "FB_Home".Execute := TRUE;
      "FB_MoveVelocity".Execute := FALSE;
      "GVL".StepTest := "GVL".StepTest + 1;
    END_IF;
```

```
3:
  IF "FB_Home".Done THEN
    //if homing has been executed correctly
    //parameterize velocity movement
    "FB_MoveVelocity".Velocity := 80000;
    "FB_MoveVelocity".Acceleration := 8000;
    "FB_MoveVelocity".Deceleration := 8000;
    //execute velocity movement
    "FB_MoveVelocity".Execute := TRUE;
    "FB_Stop".Execute := FALSE;
    "GVL".StepTest := "GVL".StepTest + 1;
  END_IF;

4:
  IF "FB_MoveVelocity".InVelocity THEN
    //if axis reached target velocity

    "FB_MoveVelocity".Execute := FALSE;
    //parametrize stop
    "FB_Stop".Deceleration := 50000;
    //execute stop
    "FB_Stop".Execute := TRUE;
    "GVL".StepTest := "GVL".StepTest + 1;
  END_IF;

5:
  IF "FB_Stop".Done THEN
    //if stop has been executed correctly
    "FB_Stop".Execute := FALSE;
    "FB_MoveAbsolute".Execute := FALSE;
    "GVL".StepTest := "GVL".StepTest + 1;
  END_IF;

6:
  //parametrize first move absolute
  "FB_MoveAbsolute".Velocity := 80000;
  "FB_MoveAbsolute".Deceleration := 80000;
  "FB_MoveAbsolute".Acceleration := 80000;
  "FB_MoveAbsolute".Position := 0;
  //execute first move absolute
  "FB_MoveAbsolute".Execute := TRUE;
  "GVL".StepTest := "GVL".StepTest + 1;
```

```
7:
  IF "FB_MoveAbsolute".Done THEN
    //if first move absolute has been executed correctly
    "FB_MoveAbsolute".Execute := FALSE;
    //parametrize second move absolute
    "FB_MoveAbsolute".Velocity := 160000;
    "FB_MoveAbsolute".Deceleration := 160000;
    "FB_MoveAbsolute".Acceleration := 160000;
    "FB_MoveAbsolute".Position := 160000;
    "GVL".StepTest := "GVL".StepTest + 1;
  END_IF;

8:
  //execute second move absolute
  "FB_MoveAbsolute".Execute := TRUE;
  //if second move absolute has been executed correctly
  IF "FB_MoveAbsolute".Done THEN
    "FB_MoveAbsolute".Execute := FALSE;
    //if count <= 2 executed absolute movements
    IF "GVL".Count <= 2 THEN
      "GVL".Count := "GVL".Count + 1;
      "GVL".StepTest := 6;
    ELSE
      "FB_MoveRelative".Execute := FALSE;
      "GVL".StepTest := "GVL".StepTest + 1;
    END_IF;
  END_IF;

9:
  //parametrize move relative
  "FB_MoveRelative".Distance := 80000;
  "FB_MoveRelative".Velocity := 8000;
  "FB_MoveRelative".Acceleration := 8000;
  "FB_MoveRelative".Deceleration := 8000;
  //execute move relative
  "FB_MoveRelative".Execute := TRUE;
  "GVL".StepTest := "GVL".StepTest + 1;

10:
  //if move relative has been executed correctly
  IF "FB_MoveRelative".Done THEN
    "FB_MoveVelocity".Execute := FALSE;
    "GVL".StepTest := "GVL".StepTest + 1;
  END_IF;
```



```
END_CASE;
```

## 5. Function blocks

In questo capitolo vengono descritti i function blocks che CMZ mette a disposizione per gestire gli azionamenti.

I function blocks sono importabili nel progetto tramite il file esterno CMZ\_FBs.scl che CMZ distribuisce.

I function block implementati permettono di:

- Abilitare e disabilitare l'azionamento
- Resettare eventuali errori dell'azionamento
- Dare all'azionamento i seguenti comandi:
  - Jog
  - Movimento in velocità
  - Posizionamento assoluto
  - Posizionamento relativo
  - Stop
  - Stop di emergenza
  - Procedura di homing
- Leggere lo stato dell'asse
- Leggere posizione, velocità, coppia attuale dell'asse
- Gestire ingressi e uscite digitali

### 5.1. Gestione dati ciclici e diagnostica dell' asse

Sono stati implementati due function block con funzione di diagnostica, tra cui uno gestisce anche i dati ciclici:

- MC\_ReadStatus: permette di gestire i dati ciclici e di leggere lo stato dell'asse.

- MC\_ReadPosVelTorque: permette di leggere posizione, velocità e coppia attuale dell'asse.

## MC\_ReadStatus

Permette di leggere e scrivere i dati ciclici tramite le funzioni *DPRD\_DAT* e *DPWR\_DAT* presenti all'interno del function block ed inoltre il function block permette di leggere lo stato dell'asse. Questo function block deve essere **sempre** richiamato con l'ingresso di Enable impostato fisso a TRUE poichè gestisce i dati ciclici.

## Interfaccia

```
FUNCTION_BLOCK MC_ReadStatus

VAR_INPUT
    pHwtelegram200      : HW_IO;
    Enable               : BOOL;
END_VAR

VAR_OUTPUT
    Telegram_REF        : Telegram200
    Active               : BOOL;
    xError               : BOOL;
    Disabled              : BOOL;
    Fault                : BOOL;
    Errorstop            : BOOL;
    Stopping             : BOOL;
    Standstill           : BOOL;
    DiscreteMotion       : BOOL;
    ContinuousMotion     : BOOL;
    SynchronizedMotion   : BOOL;
    Homing                : BOOL;
    ConstantVelocity     : BOOL;
    Accelerating          : BOOL;
    Decelerating          : BOOL;
END_VAR
```

## Parameter

### INPUT

Enable  
Tipo di dato HW\_IO

Etichetta identificativa del telegramma che viene assegnata quando si importa l'oggetto telegramma nel device.

Si trova in: *Devices e networks* → *Network view* → doppio click sull'azionamento → nella finestra *Device overview* selezionare *CMZ custom telegram 200* → dalla finestra *System constants* copiare l'etichetta.

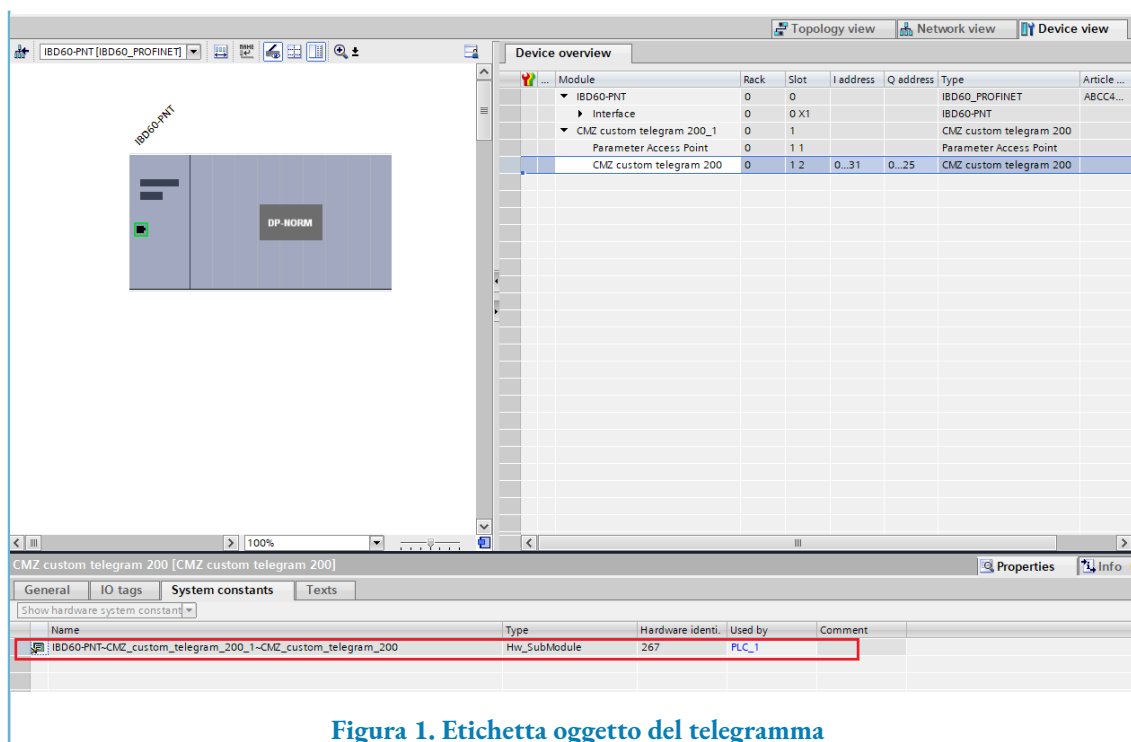


Figura 1. Etichetta oggetto del telegramma

## Enable

Tipo di dato BOOL

Flag per abilitare il function block.

## OUTPUT

## Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

Questa uscita sarà passata come IN/OUT di tutti gli altri function blocks.

## Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### xError

Tipo di dato BOOL

Errore gestione dati ciclici.

### Disabled

Tipo di dato BOOL

L'asse è disabilitato.

### Fault

Tipo di dato BOOL

Asse in fault.

### Errorstop

Tipo di dato BOOL

Asse nello stato *Errorstop*.

L'asse assume questo stato quando è in errore.

### Stopping

Tipo di dato BOOL

Asse nello stato *Stopping*.

L'asse assume questo stato quando è in esecuzione un comando di halt o di stop.

### StandStill

Tipo di dato BOOL

Asse nello stato *Standstill*.

L'asse assume questo stato quando è fermo e non è in esecuzione nessun comando.

### DiscreteMotion

Tipo di dato BOOL

Asse nello stato *Discrete motion*.

L'asse assume questo stato quando è in esecuzione un comando di posizionamento relativo o assoluto.

### ContinuousMotion

Tipo di dato BOOL

Asse nello stato *Continuous motion*.

L'asse assume questo stato quando è in esecuzione un comando di jog o un movimento in velocità.

### SynchronizedMotion

Tipo di dato BOOL

Asse nello stato *Synchronized motion*. Questo flag non è gestito poichè non sono previsti i modi sincronizzati (albero elettrico, camma, etc).

### Homing

Tipo di dato BOOL

Asse nello stato *Homing*.

### ConstantVelocity

Tipo di dato BOOL

L'asse si sta muovendo a velocità costante.

### Accelerating

Tipo di dato BOOL

Asse in accelerazione. Questo flag non è gestito.

### Decelerating

Tipo di dato BOOL

Asse in decelerazione. Questo flag non è gestito.

## Descrizione

Questo function block ritorna lo stato dell'asse, coerente con i comandi in corso e gestisce i dati ciclici.

## MC\_ReadActualVelPosTorque

Permette di leggere posizione, velocità, coppia attuale dell'asse.

### Interfaccia

```
FUNCTION_BLOCK MC_ReadActualVelPosTorque
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Enable                : BOOL;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    Position               : DINT;
    Velocity               : DINT;
    Torque                 : INT;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

#### INPUT

Enable

Tipo di dato BOOL

Flag per abilitare il function block.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### Position

Tipo di dato DINT

Posizione attuale dell'asse.

Espressa in incrementi.

### Velocity

Tipo di dato DINT

Velocità attuale dell'asse.

Espressa in incrementi/s.

### Torque

Tipo di dato INT

Coppia attuale dell'asse.

Espressa in % della corrente nominale (1000 = 100% corrente nominale).

## Descrizione

Questo function block ritorna posizione, velocità, coppia attuali dell'asse.

## 5.2. Abilitazione

Il function block utilizzato per abilitare e disabilitare l'asse è MC\_Power.



## MC\_Power

Questo function block permette di abilitare e disabilitare l'asse.

### Interfaccia

```
FUNCTION_BLOCK MC_Power
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Enable                : BOOL;
    Power                  : BOOL;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    Status                 : BOOL;
    xError                 : BOOL;
    StringError            : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

#### INPUT

Enable

Tipo di dato BOOL

Flag per abilitare il function block.

**Power**

Tipo di dato BOOL

Flag per abilitare l'asse.

**OUTPUT****Active**

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

**Status**

Tipo di dato BOOL

Quando questo flag è TRUE l'asse è abilitato.

**xError**

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

**StringError**

Tipo di dato BOOL

Stringa identificativa dell'errore.

**Descrizione**

Questo function block permette di abilitare l'asse.

**5.3. Reset**

Il function block utilizzato per resettare gli errori dell'asse è MC\_Reset.

## MC\_Reset

Questo function block permette di resettare gli errori dell'asse.

### Interfaccia

```
FUNCTION_BLOCK MC_Reset
  VAR_IN_OUT
    Telegram_REF          : Telegram200;
  END_VAR

  VAR_INPUT
    Execute               : BOOL;
  END_VAR

  VAR_OUTPUT
    Active                : BOOL;
    Done                  : BOOL;
    xError                 : BOOL;
    StringError           : STRING;
  END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

#### INPUT

Execute

Tipo di dato BOOL

Flag che permette di resettare gli errori.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### Done

Tipo di dato BOOL

Quando questo flag è TRUE il function block è terminato correttamente.

### xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

### StringError

Tipo di dato BOOL

Stringa identificativa dell'errore.

## Descrizione

Questo function block permette di resettare gli errori dell'asse.

## 5.4. Homing

Il function block utilizzato per la procedura di homing è MC\_Home.

## MC\_Home

Questo function block permette all'asse di eseguire la procedura di homing configurata all'interno dell'azionamento.

### Interfaccia

```
FUNCTION_BLOCK MC_Home
  VAR_IN_OUT
    Telegram_REF          : Telegram200;
  END_VAR

  VAR_INPUT
    Execute               : BOOL;
    Position               : DINT;
  END_VAR

  VAR_OUTPUT
    Active                : BOOL;
    Done                  : BOOL;
    CommandAborted        : BOOL;
    xError                 : BOOL;
    StringError           : STRING;
  END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

#### INPUT

Execute

Tipo di dato BOOL

Sul fronte di salita di questo flag inizia l'esecuzione del function block e l'asse esegue la procedura di homing.

**Position**

Tipo di dato DINT

Offset per la procedura di homing.

**OUTPUT****Active**

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

**Done**

Tipo di dato BOOL

Quando questo flag è TRUE il function block è terminato correttamente.

**CommandAborted**

Tipo di dato BOOL

Quando questo flag è TRUE il function block è stato abortito da un altro comando.

**xError**

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

**StringError**

Tipo di dato BOOL

Stringa identificativa dell'errore.

**Descrizione**

Questo function block permette all'asse di eseguire la procedura di homing configurata all'interno dell'azionamento.

**Nota**

Velocità, accelerazione, decelerazione e modalità di homing devono essere impostate all'interno dell'azionamento.

**5.5. Movimenti**

I movimento supportati sono:

- Jog con il function block MC\_Jog.
- Movimento in velocità con il function block MC\_MoveVelocity.
- Posizionamento assoluto con il function block MC\_MoveAbsolute.
- Posizionamento relativo con il function block MC\_MoveRelative.

## MC\_Jog

Questo function block permette di muovere l'asse quando uno degli ingressi tra JogForward e JogBackward è TRUE.

### Interfaccia

```
FUNCTION_BLOCK MC_Jog
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    JogForward            : BOOL;
    JogBackward           : BOOL;
    Velocity              : DINT;
    Acceleration          : UDINT;
    Deceleration          : UDINT;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    CommandAborted        : BOOL;
    xError                : BOOL;
    StringError           : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.



## INPUT

### JogForward

Tipo di dato BOOL

Quando questo flag è TRUE l'asse si muove in direzione positiva.

Se l'ingresso JogBackward è TRUE contemporaneamente non viene effettuato nessun movimento.

### JogBackward

Tipo di dato BOOL

Quando questo flag è TRUE l'asse si muove in direzione negativa.

Se l'ingresso JogForward è TRUE contemporaneamente non viene effettuato nessun movimento.

### Velocity

Tipo di dato DINT

Velocità per il movimento.

### Acceleration

Tipo di dato DINT

Accelerazione per il movimento.

### Deceleration

Tipo di dato DINT

Decelerazione per il movimento.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### CommandAborted

Tipo di dato BOOL

Quando questo flag è TRUE il function block è stato abortito da un altro comando.

### xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

StringError

Tipo di dato BOOL

Stringa identificativa dell'errore.

## Descrizione

Questo function block permette di muovere l'asse con una velocità, accelerazione e decelerazione specifica, in direzione positiva o negativa. Quando i flag che permettono il movimento vengono rilasciati l'asse si ferma.

## MC\_MoveVelocity

Questo function block permette all'asse di eseguire un movimento in velocità.

### Interfaccia

```
FUNCTION_BLOCK MC_MoveVelocity
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Execute               : BOOL;
    Velocity               : DINT;
    Acceleration           : UDINT;
    Deceleration           : UDINT;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    InVelocity             : BOOL;
    CommandAborted         : BOOL;
    xError                 : BOOL;
    StringError            : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

#### INPUT

Execute

Tipo di dato BOOL

Sul fronte di salita di questo flag inizia l'esecuzione del function block e l'asse esegue un movimento in velocità.

### Velocity

Tipo di dato DINT

Velocità per il movimento.



#### Note

Il segno della velocità stabilisce il verso del movimento.

### Acceleration

Tipo di dato DINT

Accelerazione per il movimento.

### Deceleration

Tipo di dato DINT

Decelerazione per il movimento.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### InVelocity

Tipo di dato BOOL

Questo flag indica che l'asse ha raggiunto la velocità target.

### CommandAborted

Tipo di dato BOOL

Quando questo flag è TRUE il function block è stato abortito da un altro comando.

### xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

### StringError

Tipo di dato ERROR

Stringa identificativa dell'errore.

## **Descrizione**

Questo function block comanda all'asse un movimento in velocità con una velocità, accelerazione, decelerazione specifica.

## MC\_MoveAbsolute

Questo function block permette all'asse di eseguire un posizionamento assoluto.

### Interfaccia

```
FUNCTION_BLOCK MC_MoveAbsolute
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Execute               : BOOL;
    Position               : DINT;
    Velocity               : UDINT;
    Acceleration           : UDINT;
    Deceleration           : UDINT;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    Done                   : BOOL;
    CommandAborted         : BOOL;
    xError                 : BOOL;
    StringError            : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

## INPUT

### Execute

Tipo di dato BOOL

Sul fronte di salita di questo flag inizia l'esecuzione del function block e l'asse esegue un posizionamento assoluto.

### Position

Tipo di dato DINT

Posizione assoluta per il posizionamento.

### Velocity

Tipo di dato DINT

Velocità per il movimento.

### Acceleration

Tipo di dato DINT

Accelerazione per il movimento.

### Deceleration

Tipo di dato DINT

Decelerazione per il movimento.



### Note

Velocità, accelerazione e decelerazione devono sempre essere positive.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### Done

Tipo di dato BOOL

Quando questo flag è TRUE il function block è terminato correttamente.

### CommandAborted

Tipo di dato BOOL

Quando questo flag è TRUE il function block è stato abortito da un altro comando.

xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

StringError

Tipo di dato STRING

Stringa identificativa dell'errore.

## Descrizione

Questo function block permette all'asse di muoversi in una posizione assoluta passata in ingresso al function block, con velocità, accelerazione e decelerazione specifica.



## MC\_MoveRelative

Questo function block permette all'asse di eseguire un posizionamento relativo.

### Interfaccia

```
FUNCTION_BLOCK MC_MoveRelative
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Execute               : BOOL;
    Distance               : DINT;
    Velocity               : DINT;
    Acceleration           : UDINT;
    Deceleration           : UDINT;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    Done                  : BOOL;
    CommandAborted        : BOOL;
    xError                 : BOOL;
    StringError            : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

##### Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

## INPUT

### Execute

Tipo di dato BOOL

Sul fronte di salita di questo flag inizia l'esecuzione del function block e l'asse esegue un posizionamento relativo.

### Distance

Tipo di dato DINT

Distanza relativa da raggiungere con il posizionamento.

### Velocity

Tipo di dato DINT

Velocità per il movimento.

### Acceleration

Tipo di dato DINT

Accelerazione per il movimento.

### Deceleration

Tipo di dato DINT

Decelerazione per il movimento.



### Note

Velocità, accelerazione e decelerazione devono essere sempre positive.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### Done

Tipo di dato BOOL

Quando questo flag è TRUE il function block è terminato correttamente.

### CommandAborted

Tipo di dato BOOL

Quando questo flag è TRUE il function block è stato abortito da un altro comando.

xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

StringError

Tipo di dato STRING

Stringa identificativa dell'errore.

## Descrizione

Questo function block permette all'asse di eseguire un movimento controllato ad una specifica distanza relativa alla posizione attuale, con una velocità, accelerazione e decelerazione target.

## 5.6. Stop

I function blocks utilizzati per fermare l'asse sono:

- MC\_Stop: permette di fermare l'asse in modo non interrompibile da altri comandi.
- MC\_Halt: classico comando di stop interrompibile da altri comandi.

## MC\_Stop

Questo function block permette di fermare l'asse in modo non interrompibile da altri comandi.

### Interfaccia

```
FUNCTION_BLOCK MC_Stop
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Execute               : BOOL;
    Deceleration          : UDINT;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    Done                  : BOOL;
    xError                : BOOL;
    StringError           : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

#### INPUT

Execute

Tipo di dato BOOL

Sul fronte di salita di questo flag inizia l'esecuzione del function block e l'asse inizia a fermarsi.

## Deceleration

Tipo di dato DINT

Decelerazione per il comando di stop.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### Done

Tipo di dato BOOL

Quando questo flag è TRUE il function block è terminato correttamente.

### xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

### StringError

Tipo di dato STRING

Stringa identificativa dell'errore.

## Descrizione

Questo function block permette di fermare l'asse.



### Nota

Questo tipo di stop non è interrompibile da nessun altro comando quindi nessun altro function block può essere mandato in esecuzione se questo function block è in esecuzione e non ha ancora dato il Done.

## MC\_Halt

Questo function block permette di fermare l'asse in modo interrompibile da altri comandi.

### Interfaccia

```
FUNCTION_BLOCK MC_Halt
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Execute               : BOOL;
    Deceleration          : UDINT;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    Done                  : BOOL;
    xError                : BOOL;
    StringError           : STRING;
END_VAR
```

### Parameter

#### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

#### INPUT

Execute

Tipo di dato BOOL

Sul fronte di salita di questo flag inizia l'esecuzione del function block e l'asse inizia a fermarsi.

## Deceleration

Tipo di dato DINT

Decelerazione per il comando di halt.

## OUTPUT

### Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

### Done

Tipo di dato BOOL

Quando questo flag è TRUE il function block è terminato correttamente.

### xError

Tipo di dato BOOL

Quando questo flag è TRUE c'è stato un errore all'interno del function block.

### StringError

Tipo di dato STRING

Stringa identificativa dell'errore.

## Descrizione

Questo function block permette di fermare l'asse.

## 5.7. Ingressi e uscite digitali

I function blocks utilizzati per gestire gli ingressi digitali e le uscite digitali sono

- MC\_ReadDInputs: permette di leggere gli ingressi digitali.
- MC\_ReadWriteDOutputs: permette di leggere e scrivere le uscite digitali.

## MC\_ReadInputs

Questo function block permette di leggere gli ingressi digitali.

### Interfaccia

```
FUNCTION_BLOCK MC_ReadDInputs
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Enable                : BOOL;
END_VAR

VAR_OUTPUT
    Active                : BOOL;
    DInputBankStatus      : WORD;
    StatusIn0             : BOOL;
    StatusIn1             : BOOL;
    StatusIn2             : BOOL;
    StatusIn3             : BOOL;
    StatusIn4             : BOOL;
    StatusIn5             : BOOL;
    StatusIn6             : BOOL;
    StatusIn7             : BOOL;
    StatusIn8             : BOOL;
    StatusIn9             : BOOL;
    StatusIn10            : BOOL;
    StatusIn11            : BOOL;
    StatusIn12            : BOOL;
    StatusIn13            : BOOL;
    StatusIn14            : BOOL;
    StatusIn15            : BOOL;
END_VAR
```



## Parameter

### IN\_OUT

Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

### INPUT

Enable

Tipo di dato BOOL

Flag per abilitare il function block.

### OUTPUT

Active

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

DInputStatus

Tipo di dato WORD

Immagine degli ingressi digitali.

StatusIn0

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 0 è attivo.

StatusIn1

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 1 è attivo.

StatusIn2

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 2 è attivo.

StatusIn3

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 3 è attivo.

**StatusIn4**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 4 è attivo.

**StatusIn5**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 5 è attivo.

**StatusIn6**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 6 è attivo.

**StatusIn7**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 7 è attivo.

**StatusIn8**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 8 è attivo.

**StatusIn9**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 9 è attivo.

**StatusIn10**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 10 è attivo.

**StatusIn11**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 11 è attivo.

**StatusIn12**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 12 è attivo.

**StatusIn13**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 13 è attivo.

**StatusIn14**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 14 è attivo.

**StatusIn15**

Tipo di dato BOOL

Quando questo flag è TRUE l'ingresso 15 è attivo.

**Descrizione**

Questo function block permette di leggere lo stato degli ingressi digitali.

## MC\_ReadWriteDOutputs

Questo function block permette di leggere e scrivere le uscite digitali.

### Interfaccia

```
FUNCTION_BLOCK MC_ReadWriteDOutputs
VAR_IN_OUT
    Telegram_REF          : Telegram200;
END_VAR

VAR_INPUT
    Enable                : BOOL;
    Out0                   : BOOL;
    Out1                   : BOOL;
    Out2                   : BOOL;
    Out3                   : BOOL;
    Out4                   : BOOL;
    Out5                   : BOOL;
    Out6                   : BOOL;
    Out7                   : BOOL;
    Out8                   : BOOL;
    Out9                   : BOOL;
    Out10                  : BOOL;
    Out11                  : BOOL;
    Out12                  : BOOL;
    Out13                  : BOOL;
    Out14                  : BOOL;
    Out15                  : BOOL;
END_VAR

VAR_OUTPUT
    Active                 : BOOL;
    DOutputBankStatus      : WORD;
    StatusOut0             : BOOL;
    StatusOut1             : BOOL;
    StatusOut2             : BOOL;
    StatusOut3             : BOOL;
    StatusOut4             : BOOL;
    StatusOut5             : BOOL;
    StatusOut6             : BOOL;
```

```
StatusOut7      : BOOL;  
StatusOut8      : BOOL;  
StatusOut9      : BOOL;  
StatusOut10     : BOOL;  
StatusOut11     : BOOL;  
StatusOut12     : BOOL;  
StatusOut13     : BOOL;  
StatusOut14     : BOOL;  
StatusOut15     : BOOL;  
END_VAR
```

## Parameter

### IN\_OUT

#### Telegram\_REF

Tipo di dato Telegram200

Struttura del telegramma 200 contenente il frame di input (Device → Controller) e il frame di output (Controller → Device).

E' l'uscita Telegram\_REF del function block MC\_ReadStatus.

### INPUT

#### Enable

Tipo di dato BOOL

Flag per abilitare il function block.

#### Out0

Tipo di dato BOOL

Flag per attivare l'uscita 0.

#### Out1

Tipo di dato BOOL

Flag per attivare l'uscita 1.

#### Out2

Tipo di dato BOOL

Flag per attivare l'uscita 2.

#### Out3

Tipo di dato BOOL

Flag per attivare l'uscita 3.

Out4

Tipo di dato BOOL

Flag per attivare l'uscita 4.

Out5

Tipo di dato BOOL

Flag per attivare l'uscita 5.

Out6

Tipo di dato BOOL

Flag per attivare l'uscita 6.

Out7

Tipo di dato BOOL

Flag per attivare l'uscita 7.

Out8

Tipo di dato BOOL

Flag per attivare l'uscita 8.

Out9

Tipo di dato BOOL

Flag per attivare l'uscita 9.

Out10

Tipo di dato BOOL

Flag per attivare l'uscita 10.

Out11

Tipo di dato BOOL

Flag per attivare l'uscita 11.

Out12

Tipo di dato BOOL

Flag per attivare l'uscita 12.

Out13

Tipo di dato BOOL

Flag per attivare l'uscita 13.

**Out14**

Tipo di dato BOOL

Flag per attivare l'uscita 14.

**Out15**

Tipo di dato BOOL

Flag per attivare l'uscita 15.

**OUTPUT****Active**

Tipo di dato BOOL

Quando questo flag è TRUE il function block è in esecuzione.

**DOutputStatus**

Tipo di dato WORD

Immagine delle uscite digitali.

**StatusOut0**

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 0 è attiva.

**StatusOut1**

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 1 è attiva.

**StatusOut2**

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 2 è attiva.

**StatusOut3**

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 3 è attiva.

**StatusOut4**

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 4 è attiva.

**StatusOut5**

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 5 è attiva.

#### StatusOut6

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 6 è attiva.

#### StatusOut7

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 7 è attiva.

#### StatusOut8

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 8 è attiva.

#### StatusOut9

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 9 è attiva.

#### StatusOut10

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 10 è attiva.

#### StatusOut11

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 11 è attiva.

#### StatusOut12

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 12 è attiva.

#### StatusOut13

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 13 è attiva.

#### StatusOut14

Tipo di dato BOOL

Quando questo flag è TRUE l'uscita 14 è attiva.

#### StatusOut15

Tipo di dato BOOL



Quando questo flag è TRUE l'uscita 15 è attiva.

## **Descrizione**

Questo function block permette di leggere e scrivere le uscite digitali.

