



IBD and SBD drives with function blocks in TIA portal

Application note

Doc. TR512003
Ed. 1.2 - English



IMPORTANT

CMZ SISTEMI ELETTRONICI S.r.l. reserves the right to make changes to the products described in this document at any time without notice.

This document has been prepared by CMZ SISTEMI ELETTRONICI S.r.l. solely for use by its customers, guaranteeing that at the date of issue it is the most up-to-date documentation on the products.

Users use the document under their own responsibility and certain functions described in this document should be used with due caution to avoid danger for personnel and damage to the machines.

No other guarantee is therefore provided by CMZ SISTEMI ELETTRONICI S.r.l., in particular for any imperfections, incompleteness or operating difficulties.

This document contains confidential information that is proprietary to CMZ SISTEMI ELETTRONICI S.r.l.. Neither the document nor the information contained therein should be disclosed or reproduced in whole or in part, without express written consent of CMZ SISTEMI ELETTRONICI S.r.l..

1. Introduction	2
2. FBs import and data structure	2
3. Telegram200 structure	4
4. Examples	5
5. Function blocks	9
5.1. Manage cyclic data and axis diagnostics	9
5.2. Enable	15
5.3. Reset	17
5.4. Homing	19
5.5. Movements	21
5.6. Stop	34
5.7. Digital inputs and outputs	38

1. Introduction

This document provides some instructions on what CMZ offers for the management of the IBD and SBD PROFINET drives, through function blocks implemented by using the TIA portal environment, and on how to use them in the project.

CMZ offers two files to be imported in the project to command the drives through function blocks:

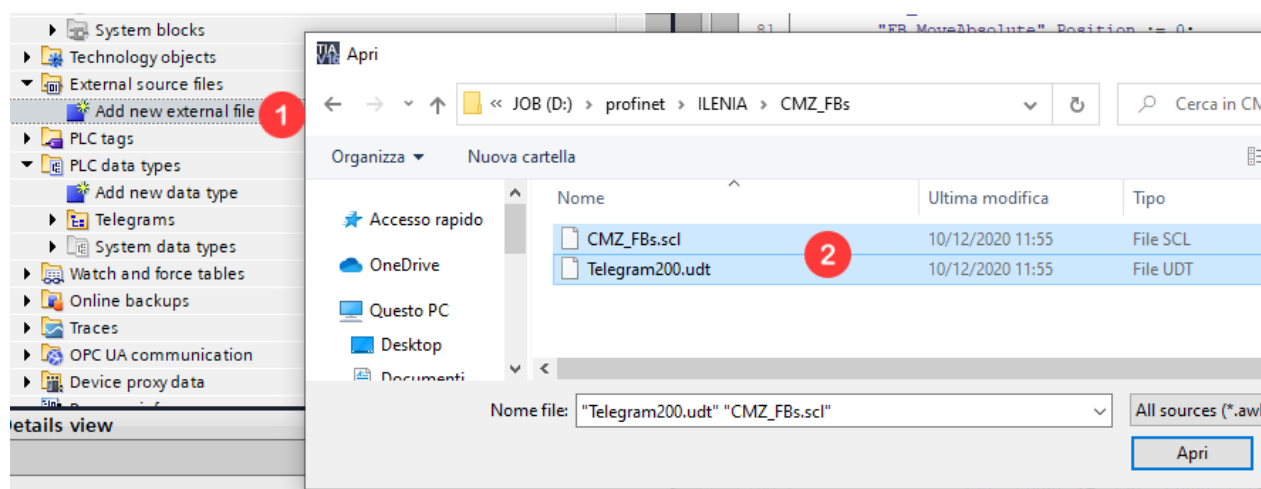
- CMZ_FBs.scl : contains the function blocks implemented to manage the drive.
- Telegram200.udt : contains the data type used as IN/OUT of all the function blocks.

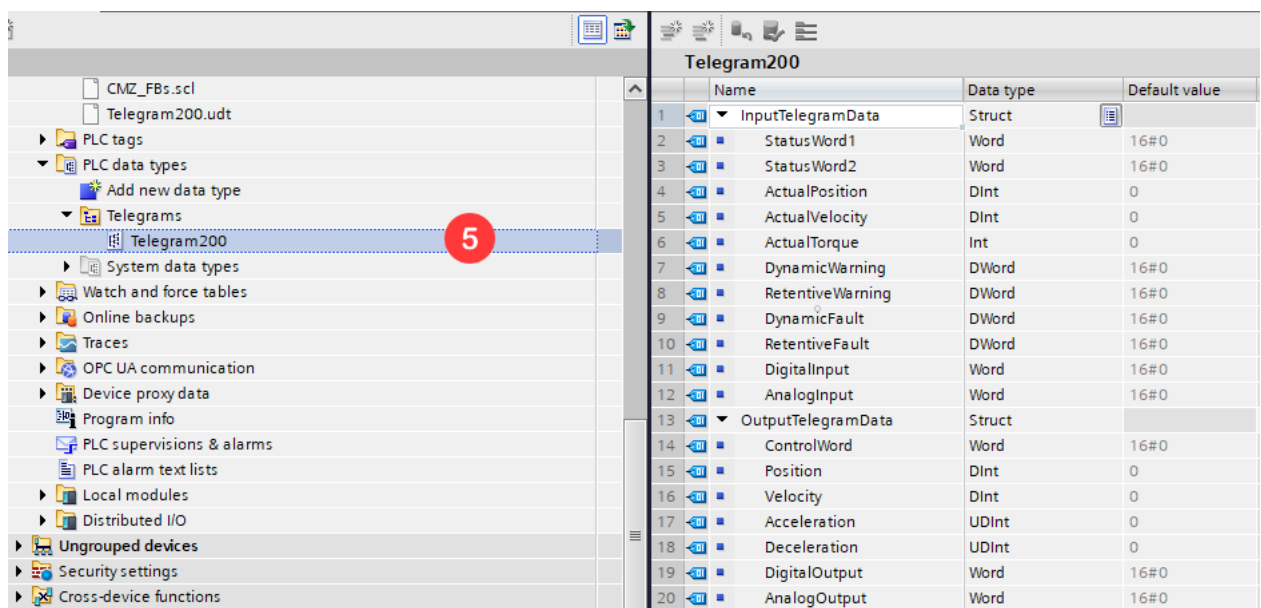
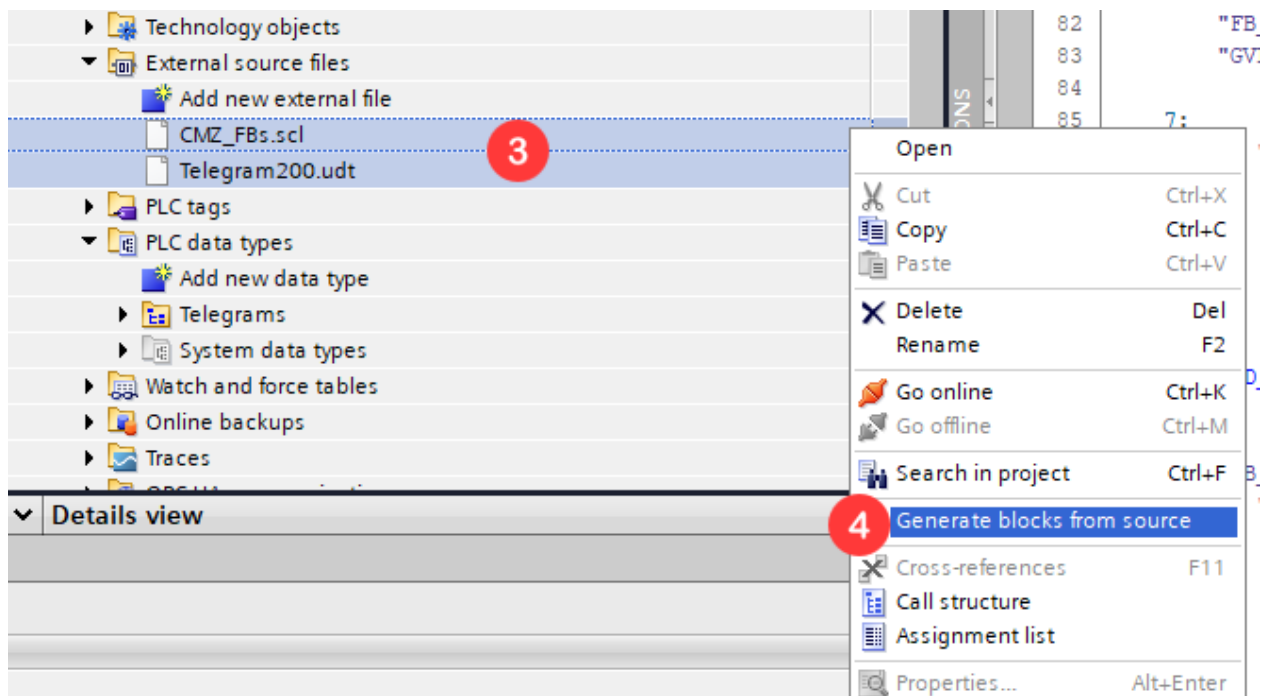
The sections in this manual describe the steps to be followed to correctly use the function blocks in a project, so it is necessary:

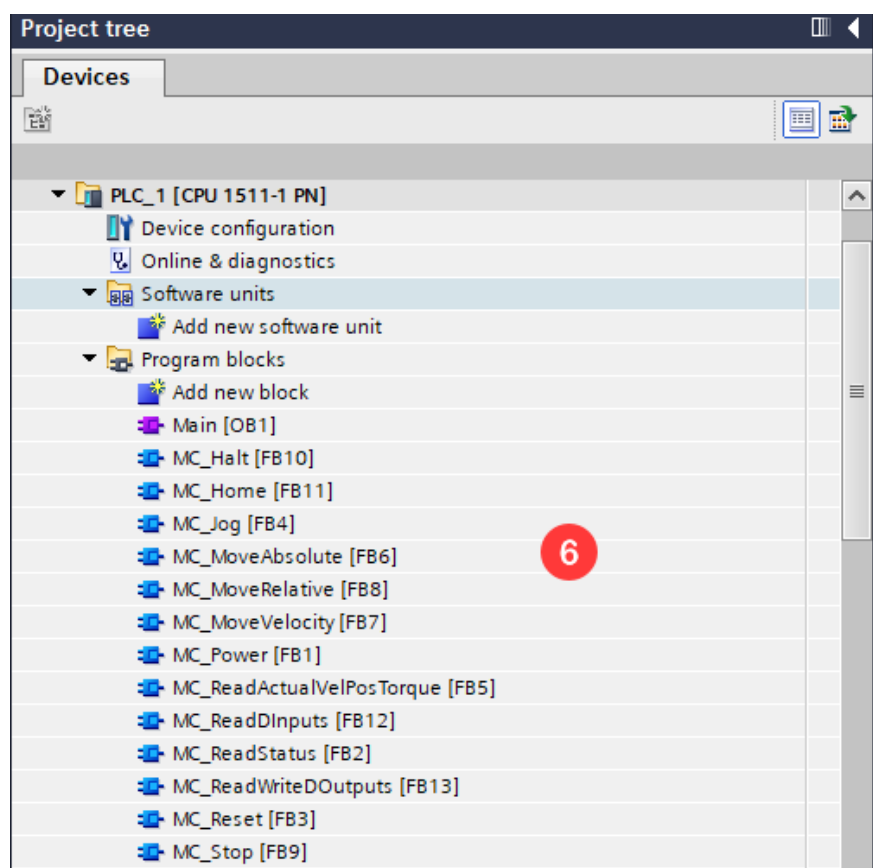
1. Import the function blocks in the project through the file CMZ_FBs.scl.
2. Import the data structure in the project through the file Telegram200.udt.
3. Instance and correctly use in the program the function blocks that the application needs, following what is reported in [Section 4, “Examples”](#), providing the data structure as IN/OUT of all the function block.

2. FBs import and data structure

In order to import the function blocks and the structure to be used in the project, implemented by CMZ, that allow to command the drive is necessary to execute the steps hereafter described:







- ① From the project tree click on *Add new external file* that is present in the *External source file* section.
- ② Select the function block (.scl) and the structure (.udt) to be imported in the project and click on *Apri*.
- ③ Once the file that contains the function blocks and the one that contains the structure has been imported, select them and right click on them.
- ④ Generate the FB blocks and the structure object to be used in the program, by clicking on *Generate blocks from source*.
- ⑤ In the *PLC data types* section the *Telegram200* structure imported through the file *Telegram200.udt* can be found.
- ⑥ In the *Program blocks* section the function blocks imported through the file *CMZ_Fbs.scl* can be found.

3. Telegram200 structure

The data structure implemented by CMZ is named *Telegram200* and can be imported in the project through the external file *Telegram200.dut* distributed by CMZ.

This structure allows to manage and command the drive and contains the telegram 200 structure with the input frame (Device → Controller) and the output frame (Controller → Device).

After being imported in the project, this structure is managed by the function block *MC_ReadStatus* that allows to read and write the cyclic data through the instructions *DPRD_DAT* and *DPWR_DAT* and then it is used as IN/OUT of all implemented function blocks.

The structure is the following:

```

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. Examples

In this chapter it is present an example that shows how to use the function blocks and the data structure in a program, after being imported in a project.

This example program allows to: enable the axis torque if it is not in error state, execute the homing procedure, execute a velocity movement with a target velocity, execute absolute positioning towards an absolute position and, at the end, a relative positioning.

```

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

```

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

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

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

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

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

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

//function block MC_MoveVelocity instance making
"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

This chapter describes the function blocks that are provided by CMZ to manage the drives.

The function blocks can be imported in the project through the external file `CMZ_FBs.scl` that is distributed by CMZ.

The implemented function blocks allow to:

- Enable and disable the drive
- Reset eventual errors of the drive
- Give the following commands to the drive:
 - Jog
 - Velocity movement
 - Absolute positioning
 - Relative positioning
 - Stop
 - Emergency stop
 - Homing procedure
- Read the axis status
- Read the axis actual position, velocity, torque
- Manage digital inputs and outputs

5.1. Manage cyclic data and axis diagnostics

Two function blocks have been implemented with diagnostics function, and one of them manages even the cyclic data:

- `MC_ReadStatus`: allows to manage cyclic data and to read the axis status.
- `MC_ReadPosVelTorque`: allows to read the axis actual position, velocity and torque.

MC_ReadStatus

Allows to read and write the cyclic data through the functions *DPRD_DAT* and *DPWR_DAT* that are present in the function block and, in addition, the function block allows to read the axis status. This function block must be **always** recalled with the Enable input steady set to TRUE because it manages the cyclic data.

Interface

```
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
HW_IO data type

Telegram identification label that is assigned when the telegram object is imported in the device.

It can be found in: *Devices e networks* → *Network view* → double click on the drive → in the window *Device overview* select *CMZ custom telegram 200* → from the window *System constants* copy the label.

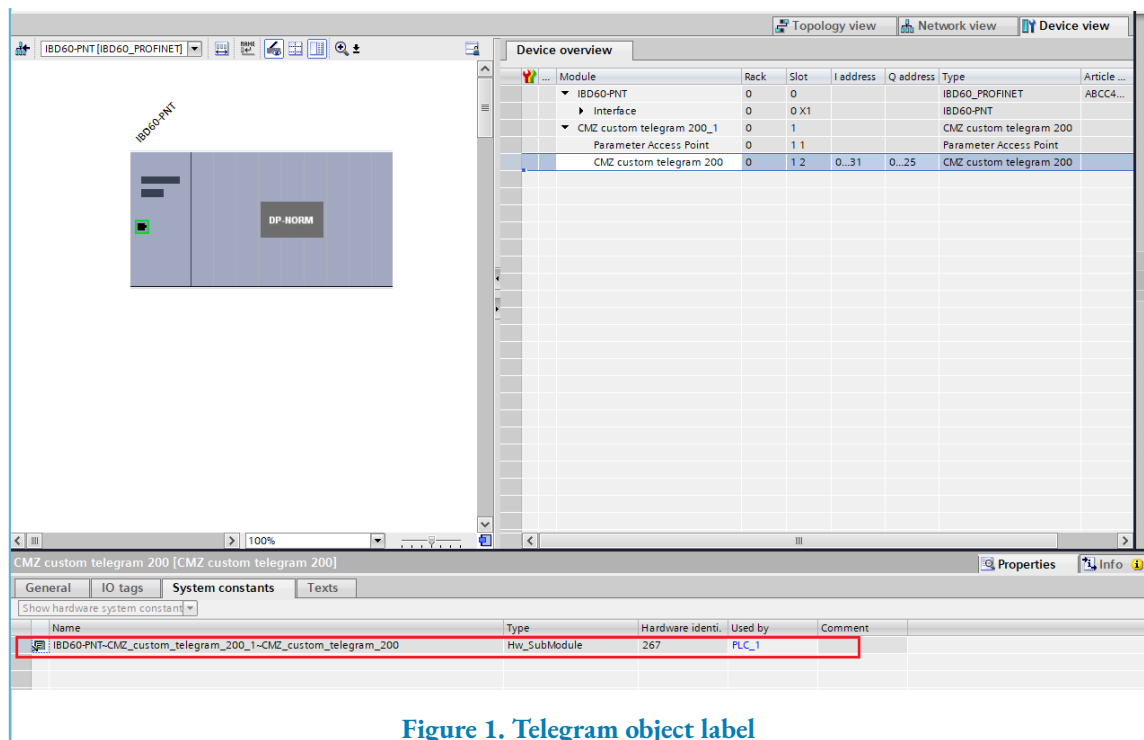


Figure 1. Telegram object label

Enable

BOOL data type

Flag to enable the function block.

OUTPUT

Telegram_REF

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

This output is passed as IN/OUT of all other function blocks.

Active

BOOL data type

When this flag is TRUE the function block is executing.

xError

BOOL data type

Cyclic data management error

Disabled

BOOL data type

The axis is disabled.

Fault

BOOL data type

Axis in fault.

Errorstop

BOOL data type

Axis in *Errorstop* status.

The axis takes this status when it is in error.

Stopping

BOOL data type

Axis in *Stopping* status.

The axis takes this status when an halt or stop command is executing.

StandStill

BOOL data type

Axis in *Standstill.* status.

The axis takes this status when it is standing still and no command is running.

DiscreteMotion

BOOL data type

The axis is in *Discrete motion* status.

The axis takes this status when an absolute or relative positioning command is executing.

ContinuousMotion

BOOL data type

Axis in *Continuous motion* status.

The axis takes this status when a jog or a velocity movement command is executing.

SynchronizedMotion

BOOL data type

Axis in *Synchronized motion* status. This flag is not managed because the synchronized modes are not provided (electric gear, cams, etc).

Homing

BOOL data type

Axis in *Homing* status.

ConstantVelocity

BOOL data type

The axis is moving with constant velocity.

Accelerating

BOOL data type

Axis in acceleration. This flag is not managed.

Decelerating

BOOL data type

Axis in deceleration. This flag is not managed.

Description

This function block returns the axis status, coherent with the commands in progress and manages the cyclic data.

MC_ReadActualVelPosTorque

It allows to read the actual position, velocity, torque of the axis.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Enable

BOOL data type

Flag to enable the function block.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Position

DINT data type

Axis actual position

Expressed in increments.

Velocity

DINT data type

Axis actual velocity.

Expressed in increments/s.

Torque

INT data type

Axis actual torque.

Expressed in % of the rated current (1000 = 100% nominal current).

Description

This function block returns the axis actual position, velocity, torque.

5.2. Enable

The function block used to enable and disable the axis is MC_Power.

MC_Power

This function block allows to enable and disable the axis.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Enable

BOOL data type

Flag to enable the function block.

Power

BOOL data type

Flag to enable the axis.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Status

BOOL data type

When this flag is TRUE the axis is enabled.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

BOOL data type

Error identification string.

Description

This function block allows to enable the axis.

5.3. Reset

The function block used to reset the errors of the axis is MC_Reset.

MC_Reset

This function block allows to reset the axis errors.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Execute

BOOL data type

Flag that allows to reset the errors.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Done

BOOL data type

When this flag is TRUE the function block has correctly ended.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

BOOL data type

Error identification string.

Description

This function block allows to reset the axis errors.

5.4. Homing

The function block used for the homing procedure is MC_Home.

MC_Home

This function block allows the axis to execute the homing procedure that is configured in the drive.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

INPUT

Execute

BOOL data type

On the rising edge of this flag the function block execution starts and the axis executes the homing procedure.

Position

DINT data type

Offset for the homing procedure.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Done

BOOL data type

When this flag is TRUE the function block has correctly ended.

CommandAborted

BOOL data type

When this flag is TRUE the function block has been aborted by another command.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

BOOL data type

Error identification string.

Description

This function block allows the axis to execute the homing procedure configured in the drive.



Note

Velocity, acceleration, deceleration and homing mode must be set inside the drive.

5.5. Movements

The supported movements are:

- Jog with the function block MC_Jog.
- Velocity movements with the function block MC_MoveVelocity.
- Absolute positioning with the function block MC_MoveAbsolute.
- Relative positioning with the function block MC_MoveRelative.

MC_Jog

This function block allows to move the axis when one input between JogForward and JogBackward is TRUE.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

JogForward

BOOL data type

When this flag is TRUE the axis moves towards positive direction.

If the JogBackward input is contemporary TRUE no movement is executed.

JogBackward

BOOL data type

When this flag is TRUE the axis moves towards negative direction.

If the JogForward input is contemporary TRUE no movement is executed.

Velocity

DINT data type

Velocity for the movement.

Acceleration

DINT data type

Acceleration for the movement.

Deceleration

DINT data type

Deceleration for the movement.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

CommandAborted

BOOL data type

When this flag is TRUE the function block has been aborted by another command.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

BOOL data type

Error identification string.

Description

This function block allows to move the axis with a specific velocity, acceleration and deceleration, towards positive or negative direction. When the flags that allow the movement are released the axis stops.

MC_MoveVelocity

This function block allows the axis to execute a velocity movement.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Execute

BOOL data type

On the rising edge of this flag the function block execution starts and the axis executes a velocity movement.

Velocity

DINT data type

Velocity for the movement.



Note

The velocity sign determines the movement direction.

Acceleration

DINT data type

Acceleration for the movement.

Deceleration

DINT data type

Deceleration for the movement.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

InVelocity

BOOL data type

This flag indicates that the axis has reached the target velocity.

CommandAborted

BOOL data type

When this flag is TRUE the function block has been aborted by another command.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

ERROR data type

Error identification string.

Description

This function block commands to the axis a velocity movement with a specific velocity, acceleration, deceleration.

MC_MoveAbsolute

This function block allows the axis to execute an absolute positioning.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Execute

BOOL data type

On the rising edge of this flag the function block execution starts and the axis executes an absolute positioning.

Position

DINT data type

Absolute position for the positioning.

Velocity

DINT data type

Velocity for the movement.

Acceleration

DINT data type

Acceleration for the movement.

Deceleration

DINT data type

Deceleration for the movement.



Note

Velocity, acceleration and deceleration must always be positive.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Done

BOOL data type

When this flag is TRUE the function block has correctly ended.

CommandAborted

BOOL data type

When this flag is TRUE the function block has been aborted by another command.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

STRING data type

Error identification string.

Description

This function block allows the axis to move towards an absolute position provided on the function block input, with a specific velocity, acceleration and deceleration.

MC_MoveRelative

This function block allows the axis to execute a relative positioning.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Execute

BOOL data type

On the rising edge of this flag the function block execution starts and the axis executes a relative positioning.

Distance

DINT data type

Relative distance to be reached with the positioning.

Velocity

DINT data type

Velocity for the movement.

Acceleration

DINT data type

Acceleration for the movement.

Deceleration

DINT data type

Deceleration for the movement.



Note

Velocity, acceleration and deceleration must always be positive.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Done

BOOL data type

When this flag is TRUE the function block has correctly ended.

CommandAborted

BOOL data type

When this flag is TRUE the function block has been aborted by another command.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

STRING data type

Error identification string.

Description

This function block allows the axis to execute a controlled movement to a specific distance that is relative to the actual position, with a target velocity, acceleration and deceleration.

5.6. Stop

The function blocks used to stop the axis are:

- MC_Stop: allows to stop the axis in a way that cannot be interrupted by other commands.
- MC_Halt: classic stop command that can be interrupted by other commands.

MC_Stop

This function block allows to stop the axis in a way that cannot be interrupted by other commands.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Execute

BOOL data type

On the rising edge of this flag the function block execution starts and the axis starts to stop.

Deceleration

DINT data type

Deceleration for the stop command.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Done

BOOL data type

When this flag is TRUE the function block has correctly ended.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

STRING data type

Error identification string.

Description

This function block allows to stop the axis.



Note

This stop type cannot be interrupted by any other command so no other function block execution can be launched if this function block is executing and has not yet returned the Done.

MC_Halt

This function block allows to stop the axis in a way that can be interrupted by other commands.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Execute

BOOL data type

On the rising edge of this flag the function block execution starts and the axis starts to stop.

Deceleration

DINT data type

Deceleration for the halt command.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

Done

BOOL data type

When this flag is TRUE the function block has correctly ended.

xError

BOOL data type

When this flag is TRUE an error is happened in the function block.

StringError

STRING data type

Error identification string.

Description

This function block allows to stop the axis.

5.7. Digital inputs and outputs

The function blocks used to manage the digital inputs and outputs are

- MC_ReadDInputs: allows to read the digital inputs.
- MC_ReadWriteDOutputs: allows to read and write the digital outputs.

MC_ReadInputs

This function block allows to read the digital inputs.

Interface

```
FUNCTION_BLOCK MC_ReadInputs
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Enable

BOOL data type

Flag to enable the function block.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

DInputStatus

WORD data type

Digital inputs image.

StatusIn0

BOOL data type

When this flag is TRUE the input 0 is active.

StatusIn1

BOOL data type

When this flag is TRUE the input 1 is active.

StatusIn2

BOOL data type

When this flag is TRUE the input 2 is active.

StatusIn3

BOOL data type

When this flag is TRUE the input 3 is active.

StatusIn4

BOOL data type

When this flag is TRUE the input 4 is active.

StatusIn5

BOOL data type

When this flag is TRUE the input 5 is active.

StatusIn6

BOOL data type

When this flag is TRUE the input 6 is active.

StatusIn7

BOOL data type

When this flag is TRUE the input 7 is active.

StatusIn8

BOOL data type

When this flag is TRUE the input 8 is active.

StatusIn9

BOOL data type

When this flag is TRUE the input 9 is active.

StatusIn10

BOOL data type

When this flag is TRUE the input 10 is active.

StatusIn11

BOOL data type

When this flag is TRUE the input 11 is active.

StatusIn12

BOOL data type

When this flag is TRUE the input 12 is active.

StatusIn13

BOOL data type

When this flag is TRUE the input 13 is active.

StatusIn14

BOOL data type

When this flag is TRUE the input 14 is active.

StatusIn15

BOOL data type

When this flag is TRUE the input 15 is active.

Description

This function block allows ti read the status of the digital inputs.

MC_ReadWriteDOutputs

This function block allows to read and write the digital outputs.

Interface

```
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

Telegram200 data type

Structure of the telegram 200 that contains the input frame (Device → Controller) and the output frame (Controller → Device).

It is the Telegram_REF output of the MC_ReadStatus function block.

INPUT

Enable

BOOL data type

Flag to enable the function block.

Out0

BOOL data type

Flag to activate the output 0.

Out1

BOOL data type

Flag to activate the output 1.

Out2

BOOL data type

Flag to activate the output 2.

Out3

BOOL data type

Flag to activate the output 3.

Out4

BOOL data type

Flag to activate the output 4.

Out5

BOOL data type

Flag to activate the output 5.

Out6

BOOL data type

Flag to activate the output 6.

Out7

BOOL data type

Flag to activate the output 7.

Out8

BOOL data type

Flag to activate the output 8.

Out9

BOOL data type

Flag to activate the output 9.

Out10

BOOL data type

Flag to activate the output 10.

Out11

BOOL data type

Flag to activate the output 11.

Out12

BOOL data type

Flag to activate the output 12.

Out13

BOOL data type

Flag to activate the output 13.

Out14

BOOL data type

Flag to activate the output 14.

Out15

BOOL data type

Flag to activate the output 15.

OUTPUT

Active

BOOL data type

When this flag is TRUE the function block is executing.

DOutputStatus

WORD data type

Image of the digital outputs.

StatusOut0

BOOL data type

When this flag is TRUE the output 0 is active.

StatusOut1

BOOL data type

When this flag is TRUE the output 1 is active.

StatusOut2

BOOL data type

When this flag is TRUE the output 2 is active.

StatusOut3

BOOL data type

When this flag is TRUE the output 3 is active.

StatusOut4

BOOL data type

When this flag is TRUE the output 4 is active.

StatusOut5

BOOL data type

When this flag is TRUE the output 5 is active.

StatusOut6

BOOL data type

When this flag is TRUE the output 6 is active.

StatusOut7

BOOL data type

When this flag is TRUE the output 7 is active.

StatusOut8

BOOL data type

When this flag is TRUE the output 8 is active.

StatusOut9

BOOL data type

When this flag is TRUE the output 9 is active.

StatusOut10

BOOL data type

When this flag is TRUE the output 10 is active.

StatusOut11

BOOL data type

When this flag is TRUE the output 11 is active.

StatusOut12

BOOL data type

When this flag is TRUE the output 12 is active.

StatusOut13

BOOL data type

When this flag is TRUE the output 13 is active.

StatusOut14

BOOL data type

When this flag is TRUE the output 14 is active.

StatusOut15

BOOL data type

When this flag is TRUE the output 15 is active.

Description

This function block allows ti read and write the digital outputs.