



IBD and SBD drives with control word and status word in TIA portal

Application note

Doc. TR512004
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	1
2. Creation of a new project	2
3. GSDML file import	3
4. Import and identification of the devices	4
5. Management of cyclic data	8
5.1. Telegram 200	12
6. Management of the acyclic data	23

1. Introduction

This manual provides some instructions on the first use of the TIA portal environment with our PROFINET drives of the IBD and SBD drives, managed through the telegram 200.

The various sections of this document follow a sequence according to how the project must be created and managed:

1. Creation of a new project;
2. Setting of the GSDML file;
3. Configuration of the hardware section;
4. Configuration of the software section and program development.



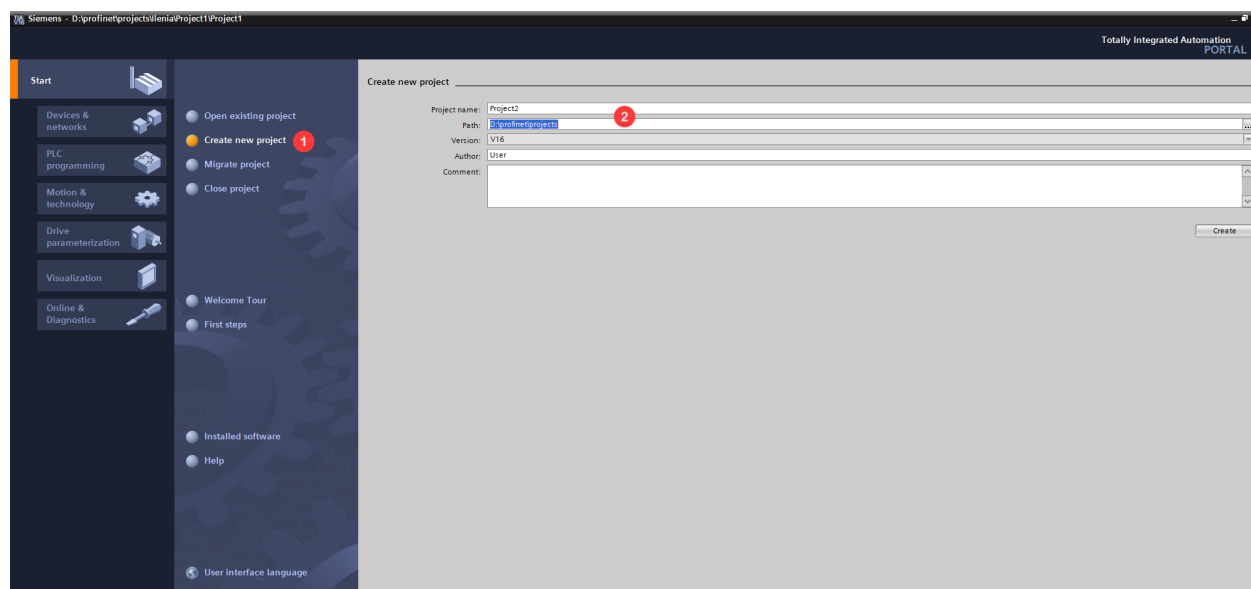
Note

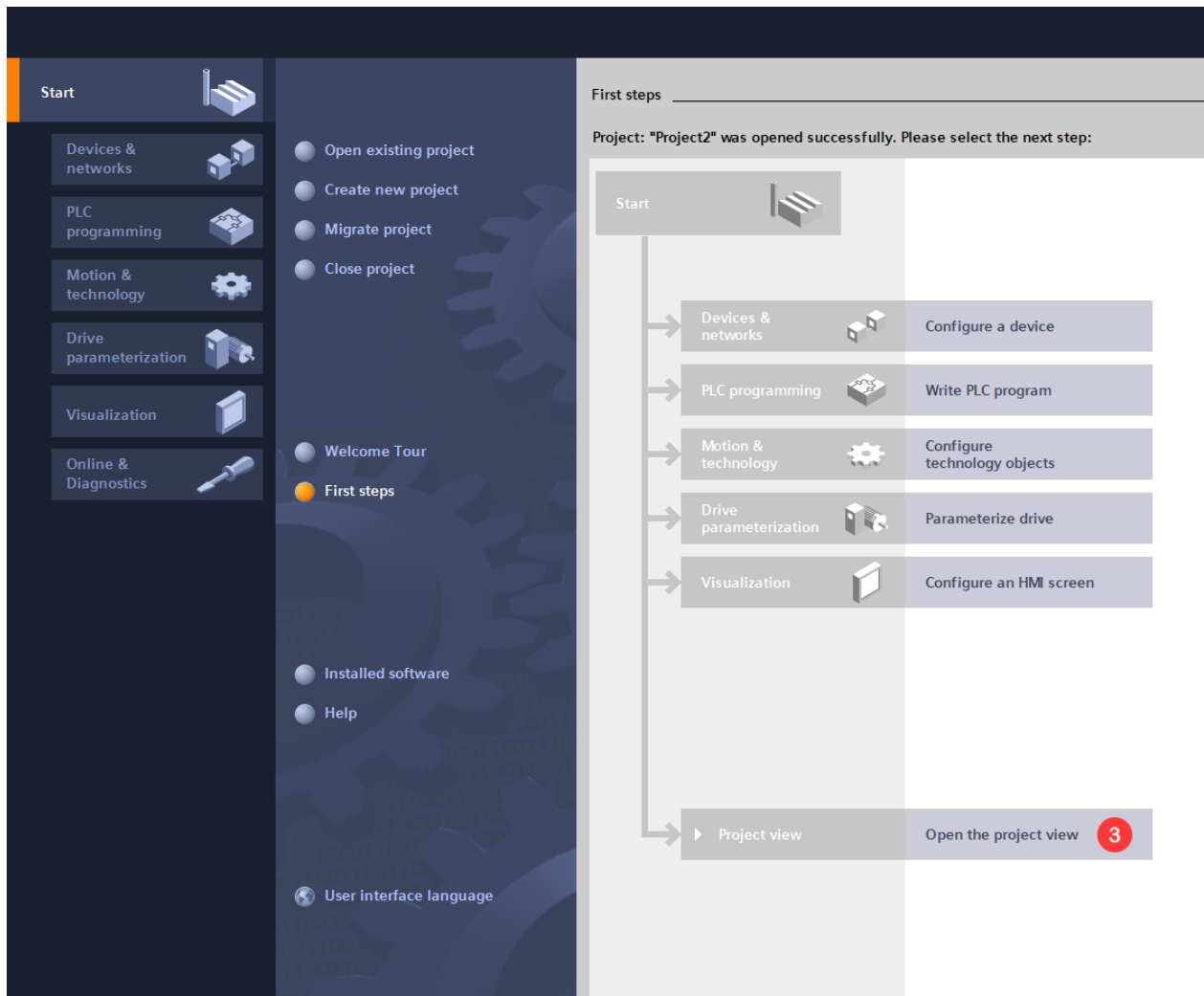
The configuration of the project and the development of the software section present in this document are implemented by using:

- The controller Siemens SIMATIC S7-1500 CPU 1511-1 PN model 6ES7 511-1AK02-0AB0;
- the development environment TIA Portal v16.

2. Creation of a new project

To create a new project with TIA portal follow the hereafter described steps:



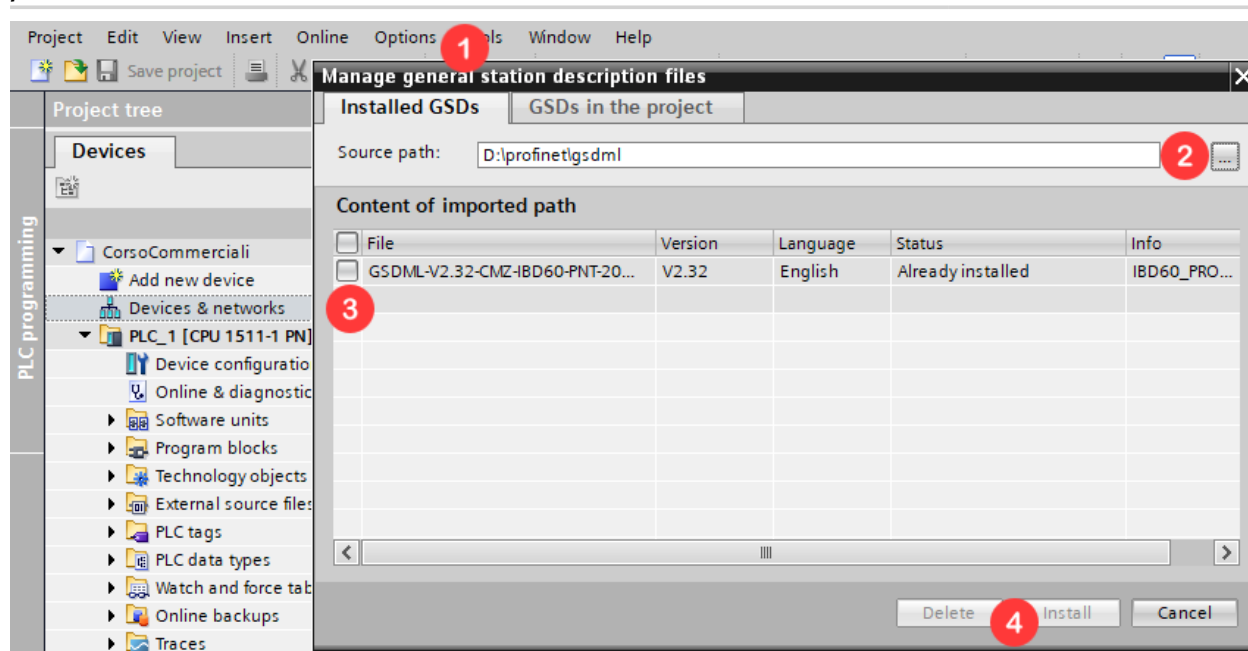


- ① Press on *Create new project*.
- ② Insert the name of the project and the path in which save it.
- ③ When the project will be created, press *Open the project view* to start the hardware configuration and the software writing.

3. GSDML file import

The GSDML file is a file that describes the device that has to be used in the application and is distributed by CMZ.

To import the GSDML file it is necessary, after the project creation:



- 1 From the menu bar press *Option* → *Manage general station description files(GSD)* .
- 2 Select the path where the GSDML file that has to be imported is saved.
- 3 Select the GSDML file to be imported.
- 4 Click on *Install* to install the selected GSDML file.

4. Import and identification of the devices

The hardware configuration of the project is related to the insertion and the configuration in the project of the devices that are used in the application.

To configure the hardware section follow the hereafter described steps:

1. From the project tree double click on *Device e networks* and 3 tabs will appear: *Topology view*, *Network view* and *Device view*.
2. In *Topology view* drag, from the hardware catalogue that is in the window on the right, the devices that the application needs:
 - Controller : Controller → select the controller correct model.

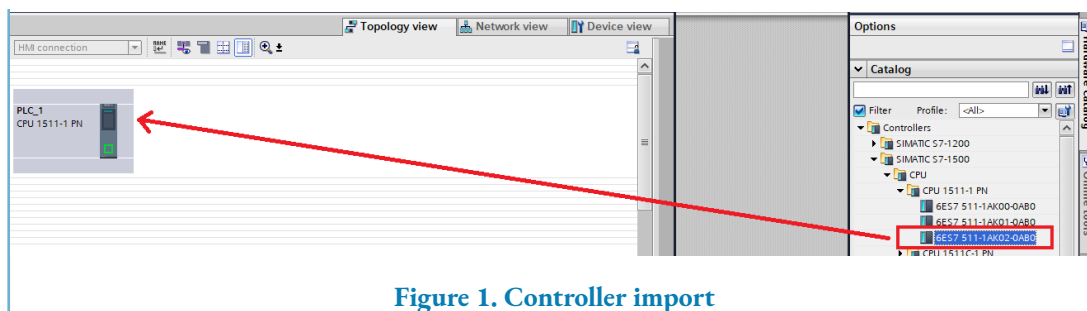


Figure 1. Controller import

- Drive: Other filed devices → PROFINET IO → Drives → CMZ Sistemi Elettronici → BD drives → IBD60-PROFINET (for IBD drive) or SBD-SSD (for SBD drive).

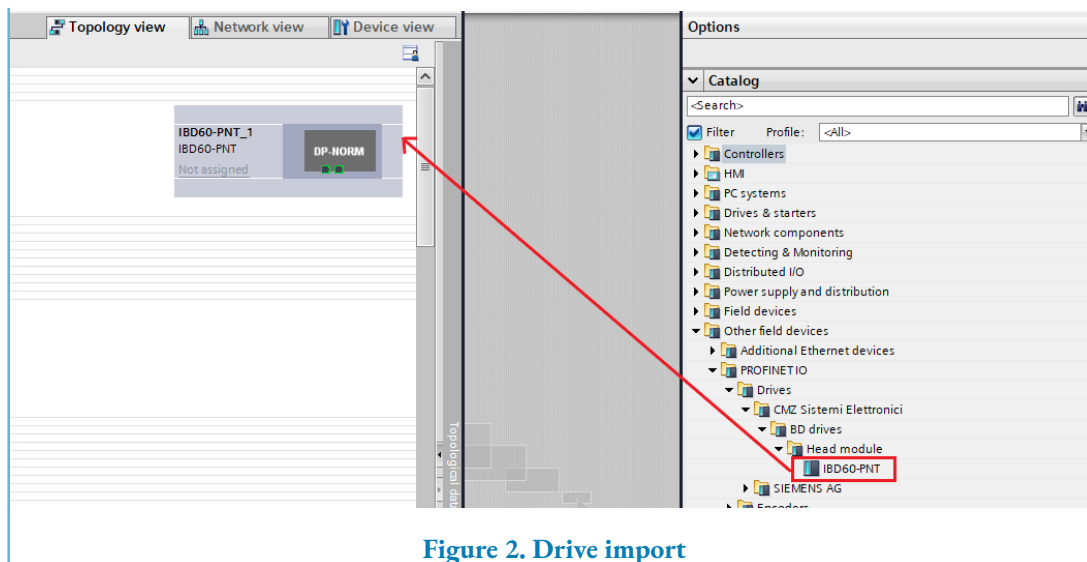


Figure 2. Drive import

3. In *Topology view* connect the controller and the drive by wiring the cable between the correct port of the controller to the correct one of the drive, according to how they are physically connected.

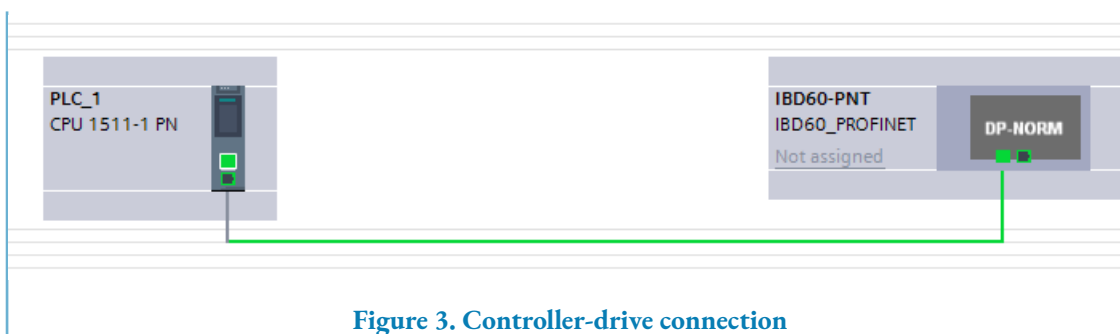


Figure 3. Controller-drive connection

4. In *Network view* select the PROFINET interface (name of the controller from which the drive is commanded) by clicking on the blue writing *Not assigned* that appears in the device of the drive and by selecting the correct interface.

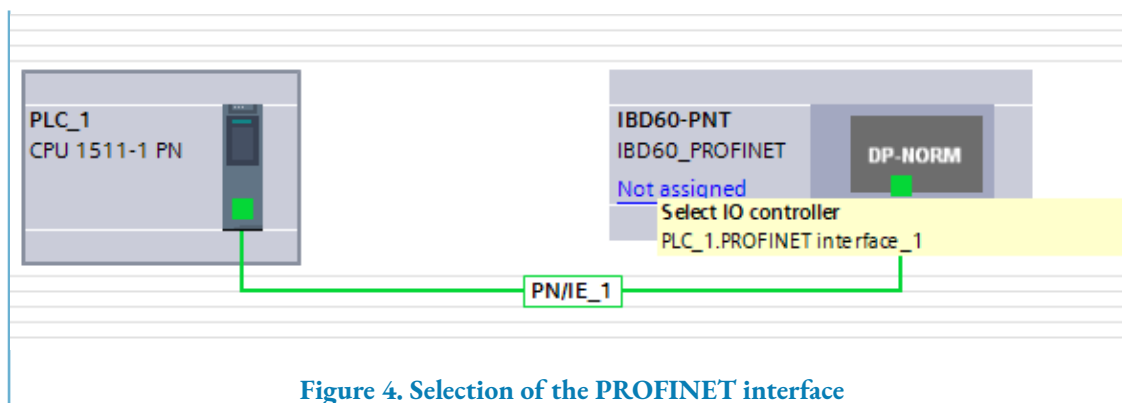


Figure 4. Selection of the PROFINET interface

5. In *Network view* double click on the controller device and on the tab *General*:
 - From the window *Ethernet addresses* set in the slot *Ip protocol* the address to be assigned to the controller.
 - From the window *Cycle* set the maximum and the minimum cycle time. The cycle time of the cyclic objects (OBs) will be the lowest set cycle time.
6. In *Network view* double click on the drive device and import the telegram to be used, by dragging it. The telegram is in the hardware catalog under the category *Module*.

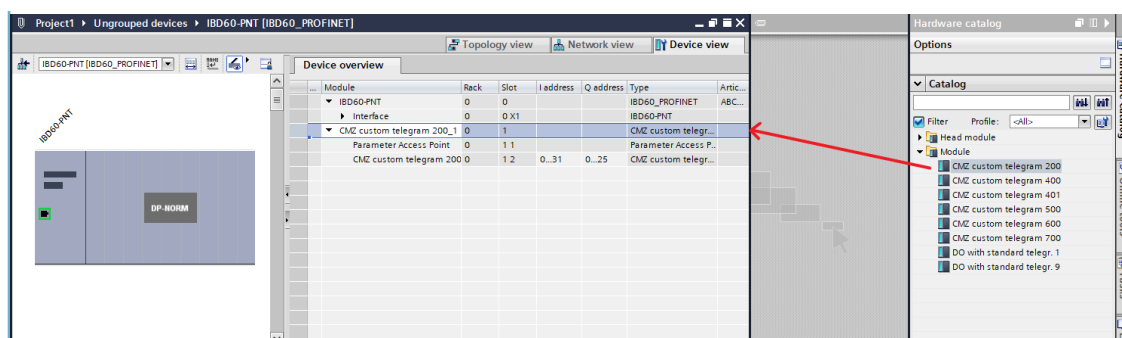
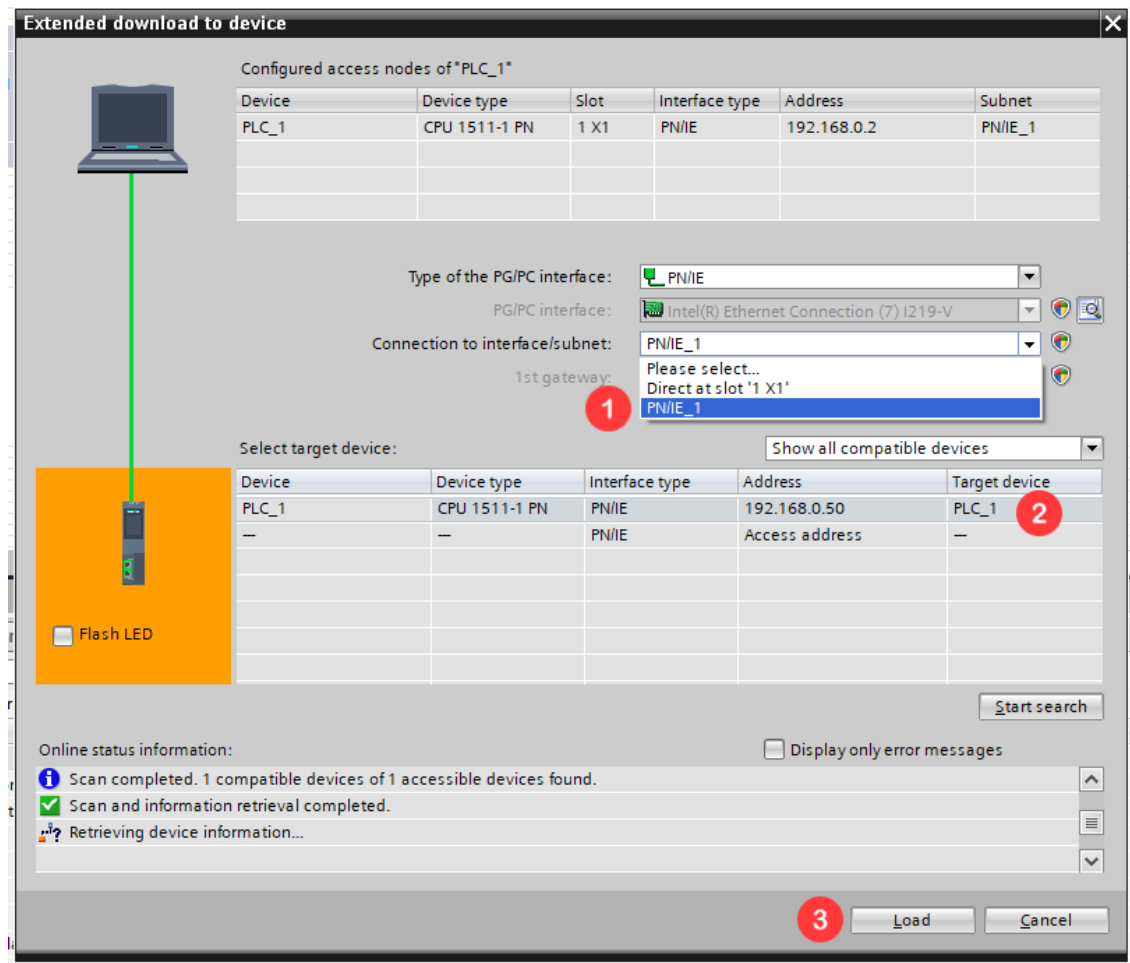


Figure 5. Telegram import

7. From the project tree click with the right button on *PLC_1[CPU1511-1 PN]* → *Compile* → *Hardware* to compile the hardware configuration. Repeat the same procedure for the software section.
8. From the project tree click with the right button on *PLC_1[CPU 1511-1 PN]* → *Download to device* → *Hardware configuration* to download the hardware configuration in the controller. Repeat the same procedure for the software section.



- ① Select the interface.
- ② Press *Start search* and select the device target on which the download must be done.
- ③ Press *Load*.

If the configuration is correct and the download has been correctly made both for the hardware and the software sections, when the mode switches to online (through the button *Go online* in the menu bar), in the project tree all the dots should be green as showed in the picture [Figure 6, "Successful download"](#).

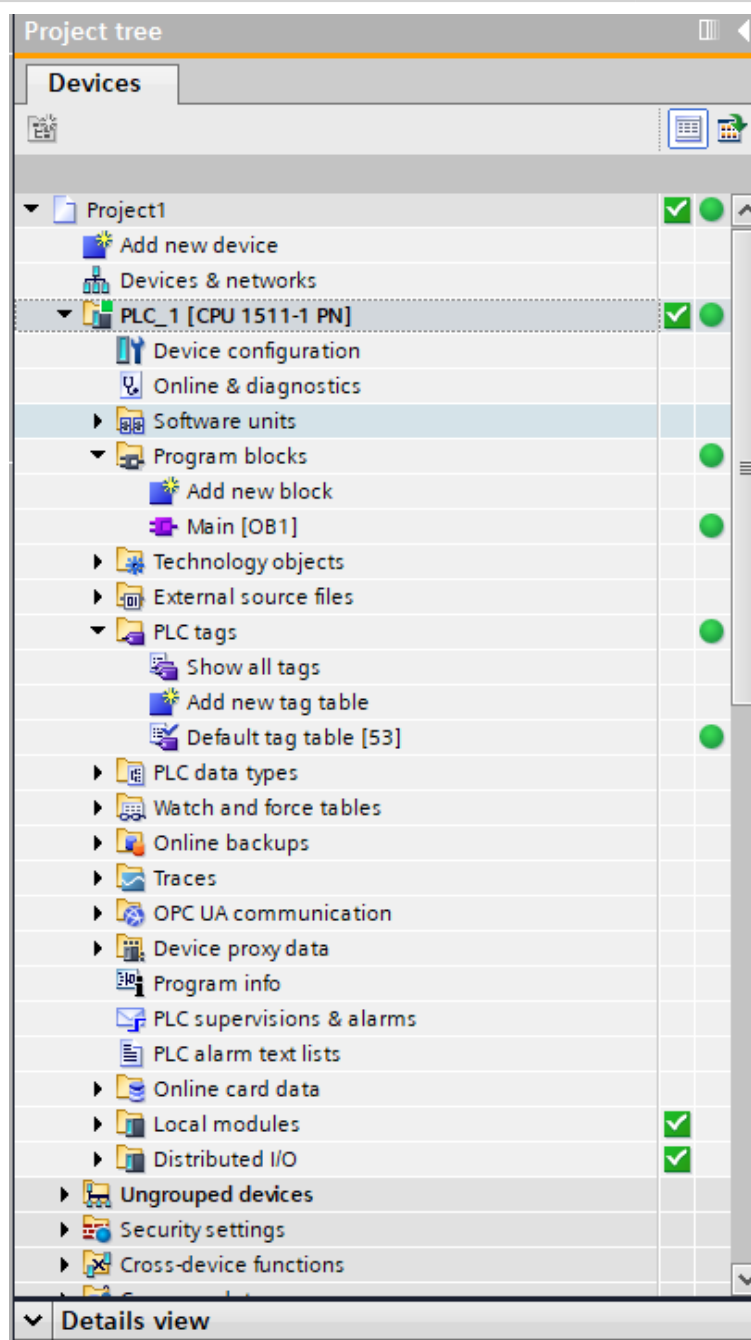


Figure 6. Successful download

5. Management of cyclic data

The cyclic data are periodic messages that are exchanged through the telegrams.

The telegrams are exchanged with periodic frequency between master (Controller) and slave (Device), allowing to manage the drive in real time mode.

In the IBD and SBD drives for the cyclic data management, the telegram 200 has been implemented, see [Section 5.1, “Telegram 200”](#).

Thanks to the telegram 200 is possible to manage and command the drive by managing the bits in the telegram.

To manage the cyclic data in the program:

1. Create a data block (right click on *Program blocks* → *Add new block* → create a *Data block*) that will include the variables and structures that are shared by all the programs and declare inside it:
 - the output frame structure (Controller → Device) that is *SentTelegramData* in this example project:

	Name	Data type	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Set...
1	Static								
2	pHwTelegram	HW_IO	*IBD60-PNT-D...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	ReceivedTelegramData	Struct				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	ErrorCodeR	Int	0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	SentTelegramData	Struct				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	controlWord	Word	16#4000			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	position	DInt	50000			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
8	velocity	DInt	50000			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
9	acceleration	UDInt	50000			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
10	deceleration	UDInt	100000			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
11	digitalOutput	Word	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
12	analogOutput	Word	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figure 7. Structure of the output telegram

- the input frame structure (Device → Controller) that is *ReceivedTelegramData* in this example project:

	Name	Data type	Start value	Monitor value	Retain	Accessible f...	Writa...	Visible in
1	Static								
2	pHwTelegram	HW_IO	*IBD60-PNT-D...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	ReceivedTelegramData	Struct				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	statusWord1	Word	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	statusWord2	Word	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	actualPosition	DInt	0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	actualVelocity	DInt	0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
8	actualTorque	Int	0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
9	dynamicWarning	DWord	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
10	retentiveWarning	DWord	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
11	dynamicFault	DWord	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
12	retentiveFault	DWord	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
13	digitalInput	Word	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
14	analogInput	Word	16#0			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figure 8. Structure of the input telegram

2. It is possible to access the telegram in two ways:

- a. Through hardware ID of the telegram, by following the hereafter described steps:
 - add in the data block (created at the point 1) the hardware ID of the telegram for the cyclic data: it is the telegram ID that has automatically assigned to the object when it is imported in the device. This ID will be used as input of the instructions that allow to read and write the cyclic data.

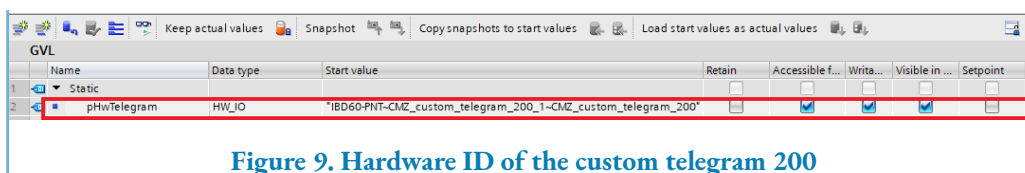


Figure 9. Hardware ID of the custom telegram 200

The *start value* is the label of the object that is 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.

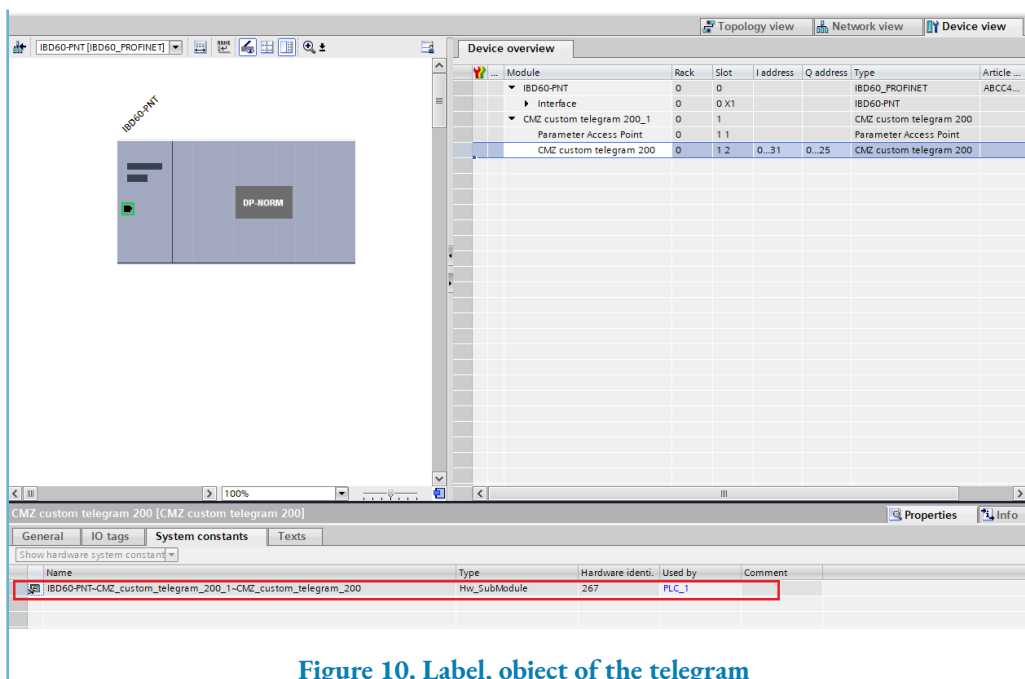


Figure 10. Label, object of the telegram

- To read the cyclic data (input frame) use the instruction *DPRD_DAT* that requires as input the hardware ID of the telegram (pHwTelegram) and as output the structure to be filled with the read data (ReceivedTelegramData).

```
"GVL".ErrorCodeR := DPRD_DAT(LADDR := "GVL".pHwTelegram, RECORD => "GVL".ReceivedTelegramData);
```

Figure 11. Instruction DPRD_DAT

- To send the cyclic data (output frame) use the instruction *DPWD_DAT* that requires as input the hardware ID of the telegram (pHwTelegram) and as output the structure with the data to be sent (ReceivedTelegramData).

```
"GVL".ErrorCodeW := DPWR_DAT(LADDR := "GVL".pHwTelegram, RECORD := "GVL".SentTelegramData);
```

Figure 12. Instruction DPWR_DAT

- Through the addresses, by following the hereafter described steps:
 - Among the properties of the data block that has been created at the point 1 (right click on the data block object → *Properties*) remove the data optimization inside the blobk by removing the attribute *Optimized block data*. If this attribute is removed it is possible to see the addressing of the data that are present in the data block.
 - Use the Siemens instruction *MOVE BLOCK* for:
 - move the input frame content of the telegram in the input frame structure that has been created in the data block;
 - move the content of the output frame created in the data block in the output frame of the telegram.

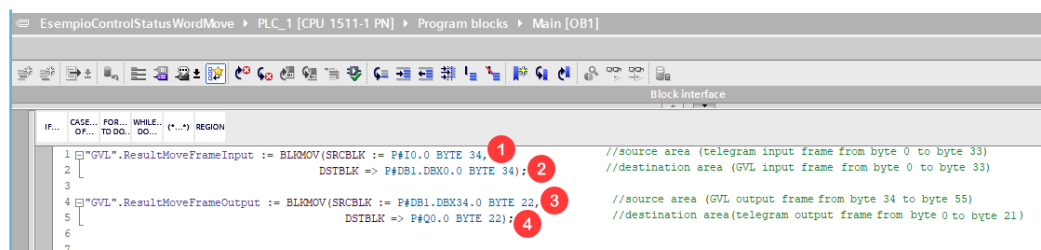


Figure 13. Instruction MOVE BLOCK on input (1,2) and output(3,4) frame.

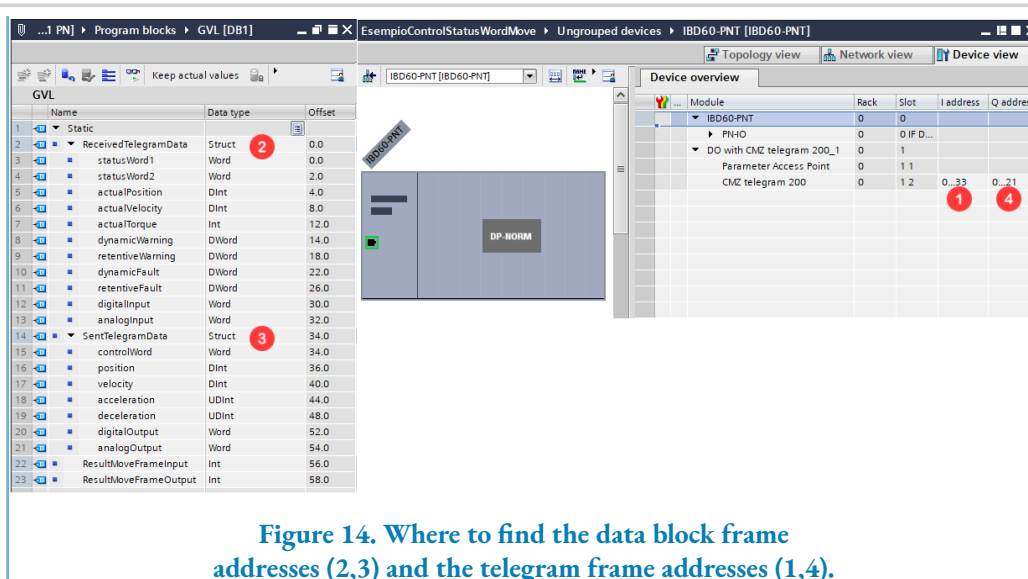


Figure 14. Where to find the data block frame addresses (2,3) and the telegram frame addresses (1,4).

5.1. Telegram 200

The telegram 200 is a cyclic message that allows to manage the drive in the real time mode and is made of :

- **Output frame** through which the PLC Controller can manage the digital and analog outputs and can get the drive to execute the following commands:
 - Enable the torque.
 - Reset.
 - Homing procedure parametrized in the drive.
 - Relative and absolute positioning.
 - Velocity movement.
 - Stop.
 - Emergency stop.
- **Input frame** through which the PLC Controller can read in the drive:
 - Status
 - Actual position

- Actual velocity
- Actual torque
- Dynamic and retentive warnings
- Dynamic and retentive faults
- Digital inputs
- Analog input

5.1.1. Output frame (Controller → Device)

The output frame is made of:

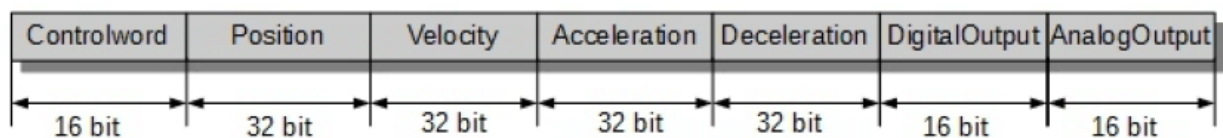


Figure 15. Output frame (Controller → Device).

- **ControlWord:** is the word that allows to give commands of the drive, by parametrizing them through the frame fields.

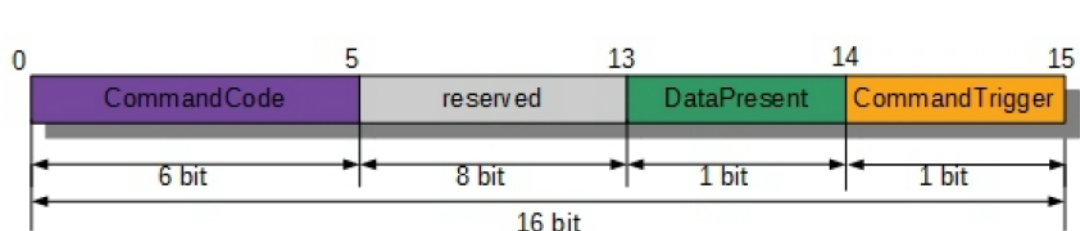


Figure 16. ControlWord structure.

- Bit 0–5 : **CommandCode**: code of the command to be executed. The implemented commands are:

Value	Name	Description
1	EmergencyStop	Executes a movement stop that cannot be interrupted.
2	Stop	Executes a movement stop that can be interrupted.
4	PowerOn	Enables the axis.
5	PowerOff	Disables the axis.
6	Home	Executes an axis homing.
7	MoveAbsolute	Executes an absolute movement.
8	MoveRelative	Executes a relative movement.
9	MoveVelocity	Executes a velocity movement.
17	Reset	Executes a faults and warnings reset.

Table 1. Commands

- Bit 6-13: reserved for future use (to be left to 0).
 - Bit 14: **DataPresent**: indicates that the Controller is sending the valid data to the Device. It must always be written to 1. If it is written to 0, the drive returns a fault.
 - Bit 15: **CommandTrigger**: on the rising edge (0 → 1) of this bit the command indicated in *CommandCode* is sent.
-
- **Position**: target position for the commands *MoveRelative* and *MoveAbsolute* and offset for the command *Home*.
Expressed in centimeters.
 - **Velocity**: target velocity for the commands *MoveRelative*, *MoveAbsolute* and *MoveVelocity*.
Expressed in increment/s.
 - **Acceleration**: acceleration for all the movement commands except for the command *Home*.
Expressed in increment/s².
 - **Deceleration**: deceleration for all the movement commands except for the command *Home*.
Expressed in increment/s².

- **DigitalOutput:** digital outputs image.
- **AnalogOutput:** analog output image.

5.1.1.1. Example of use of output frame (Controller → Device)

Some example on how to use the telegram 200 output frame to execute the commands through controlword:

- Axis enable:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //write 4 in the CommandCode field and set to 1 the bit
    //CommandTrigger of the control word to enable the axis
    "GVL".SentTelegram.controlWord := 16#C004;
END_CASE;
```

- Axis disable:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //write 5 in the CommandCode field and set to 1 the bit
    //CommandTrigger of the control word to disable
    //the axis
    "GVL".SentTelegram.controlWord := 16#C005;
END_CASE;
```

- Reset:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //write 17 in the CommandCode field and set to 1 the
    //bit CommandTrigger of the control word to
    //reset the axis
    "GVL".SentTelegram.controlWord := 16#C011;
END_CASE;
```

- Absolute position:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //positioning parametrization
    "GVL".SentTelegram.position := 0;
    "GVL".SentTelegram.velocity := 8000;
    "GVL".SentTelegram.acceleration := 8000;
    "GVL".SentTelegram.deceleration := 8000;
    //write 7 in the CommandCode field and set to 1 the
    //bit CommandTrigger of the control word to
    //execute an absolute positioning
    "GVL".SentTelegram.controlWord := 16#C007;
END_CASE;
```

- Relative positioning:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //positioning parametrization
    "GVL".SentTelegram.position := -8000;
    "GVL".SentTelegram.velocity := 8000;
    "GVL".SentTelegram.acceleration := 8000;
    "GVL".SentTelegram.deceleration := 8000;
    //write 8 in the CommandCode and set to 1 the
    //bit CommandTrigger of the control word to
    //execute an absolute positioning
    "GVL".SentTelegram.controlWord := 16#C008;
END_CASE;
```

- Velocity movement:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //parametrization of the velocity movement
    "GVL".SentTelegram.velocity := 8000;
    "GVL".SentTelegram.acceleration := 8000;
    "GVL".SentTelegram.deceleration := 8000;
    //write 9 in the CommandCode field and set to 1 the
    //bit CommandTrigger of the control word to
    //execute an absolute positioning
    "GVL".SentTelegram.controlWord := 16#C009;
END_CASE;
```

- Stop:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //parametrize the stop command
    "GVL".SentTelegram.deceleration := 8000;
    //write 2 in the CommandCode field and set to 1 the
    //bit CommandTrigger of the control word to
    //execute an axis stop
    "GVL".SentTelegram.controlWord := 16#C002;
END_CASE;
```

- Emergency stop:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //reset the bit CommandTrigger (15) of the
    //control word by writing it to 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //parametrize the stop command
    "GVL".SentTelegram.deceleration := 8000;
    //write 1 in the CommandCode field and set to 1 the
    //bit CommandTrigger of the control word to
    //execute an axis emergency stop
    "GVL".SentTelegram.controlWord := 16#C001;
END_CASE;
```

- Homing:

```

CASE "GVL".Step OF
0:
;
1:
//reset the bit CommandTrigger (15) of the
//control word by writing it to 0
"GVL".SentTelegram.controlWord := 16#4000;
"GVL".Step := "GVL".Step + 1;
2:
//set the offset for the homing procedure
"GVL".SentTelegram.position := 8000;
//write 6 in the CommandCode field and set to 1 the
//bit CommandTrigger of the control word to
//execute the homing procedure
"GVL".SentTelegram.controlWord := 16#C006;
END_CASE;

```

5.1.2. Frame di input (Device → Controller)

The input frame is made by:



Figure 17. Input frame (Device → Controller).

- **StatusWord1:** first word that summarize the drive status.

The meaning of the bit is the following:

Bit	Name
0	ReadyToSwitchOn
1	SwitchedOn
2	Run
3	Fault
4	ErrorStop
5	Stopping

Bit	Name
6	StandStill
7	DiscreteMotion
8	ContinuousMotion
9	SynchronizedMotion
10	Homing
11	Initialization
12	ConstantVelocity
13-14	Future use
15	CommandTriggerEcho: it is the copy (in reading) of the bit 15 of the Controlword. It is useful as feedback to verify if the <i>CommandTrigger</i> has been received by the drive.

Table 2. StatusWord1

- **StatusWord2:** second word that summarize the drive status.

The meaning of the bit is the following:

Bit	Name
0	Accelerating (not managed)
1	Decelerating (not managed)
2-15	Future use

Table 3. StatusWord2

- **ActualPosition:** axis actual position.
Expressed in increments.
- **ActualVelocity:** axis actual velocity.
Expressed in increment/s.
- **ActualTorque:** axis actual torque.
Expressed in thousandths of the nominal current.
- **DynamicWarning:** bit mask of the dynamic warnings.
- **RetentiveWarning:** bit mask of the retentive warnings.
- **DynamicFault:** bit mask of the dynamic faults.
- **RetentiveFault:** bit mask of the retentive faults.
- **DigitalInput:** image of the digital inputs.

- **AnalogInput:** image of the analog input.

5.1.2.1. Input frame use example (Device → Controller)

Some example on how to use the telegram 200 input frame:

- Status word1 bit reading:

```
CASE "GVL".Step OF
  1:
    //reset control bit (bit 15 control word)
    "GVL".SentTelegramData.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //if the control bit has been reset correctly
    //(bit 15 status word1 = 0)
    IF NOT "GVL".ReceivedTelegramData.statusWord1.%X15 THEN
      IF NOT "GVL".ReceivedTelegramData.statusWord1.%X4 AND
      NOT "GVL".ReceivedTelegramData.statusWord1.%X3 THEN
        //if axis isn't in errorstop, enable the axis
        "GVL".SentTelegramData.controlWord := 16#C004;
        "GVL".Step := "GVL".Step + 1;
      ELSE
        "GVL".Step := 100;
      END_IF;
    END_IF;
  3:
    //if the command has reached the drive
    //(bit 15 status word1 = 1)
    IF "GVL".ReceivedTelegramData.statusWord1.%X15 THEN
      IF "GVL".ReceivedTelegramData.statusWord1.%X1
      AND "GVL".ReceivedTelegramData.statusWord1.%X2 THEN
        //if the axis is enabled
        "GVL".SentTelegramData.controlWord := 16#4000;
        //reset control bit (bit 15 control word)
        "GVL".Step := "GVL".Step + 1;
      END_IF;
    END_IF;
  4:
    //if the control bit has been reset correctly
    //(bit 15 status word1 = 0)
```

```
IF NOT "GVL".ReceivedTelegramData.statusWord1.%X15 THEN
  //set offset homing
  "GVL".SentTelegramData.position := 0;
  //execute homing procedure
  "GVL".SentTelegramData.controlWord := 16#C006;
  "GVL".Step := "GVL".Step + 1;
END_IF;

5:
  //if the command has reached the drive
  //(bit 15 status word1 = 1) and axis is in standstill
  //(homing terminated)
  IF "GVL".ReceivedTelegramData.statusWord1.%X15
  AND "GVL".ReceivedTelegramData.statusWord1.%X6 THEN
    //reset control bit (bit 15 control word = 0)
    "GVL".SentTelegramData.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  END_IF;

6:
  //if the control bit has been reset correctly
  //(bit 15 status word1 = 0)
  IF NOT "GVL".ReceivedTelegramData.statusWord1.%X15 THEN
    //set position for first relative movement
    "GVL".SentTelegramData.position := 40000;
    //set velocity for first relative movement
    "GVL".SentTelegramData.velocity := 50000;
    //set acceleration for first relative movement
    "GVL".SentTelegramData.acceleration := 50000;
    //set deceleration for first relative movement
    "GVL".SentTelegramData.deceleration := 50000;
    //execute first relative movement
    "GVL".SentTelegramData.controlWord := 16#C007;
    "GVL".Step := "GVL".Step + 1;
  END_IF;

7:
  //if the command has reached the drive and axis
  //is in standstill (relative movement terminated)
  IF "GVL".ReceivedTelegramData.statusWord1.%X15
  AND "GVL".ReceivedTelegramData.statusWord1.%X6 THEN
    //reset control bit (bit 15 control word)
    "GVL".SentTelegramData.controlWord := 16#4000;
```

```
"GVL".Step := "GVL".Step + 1;
END_IF;

8:
//if the control bit has been reset correctly
//(bit 15 status word1 = 0)
IF NOT "GVL".ReceivedTelegramData.statusWord1.%X15 THEN
    //set position for second relative movement
    "GVL".SentTelegramData.position := -40000;
    //set velocity for second relative movement
    "GVL".SentTelegramData.velocity := 50000;
    //set acceleration for second relative movement
    "GVL".SentTelegramData.acceleration := 50000;
    //set deceleration for second relative movement
    "GVL".SentTelegramData.deceleration := 50000;
    //execute second relative movement
    "GVL".SentTelegramData.controlWord := 16#C007;
    "GVL".Step := 5;
END_IF;
END_CASE;
```

- Reading of the actual position, velocity, torque of the drive:

```
#ActPosition := "GVL".ReceivedTelegramData.actualPosition;
#ActVelocity := "GVL".ReceivedTelegramData.actualVelocity;
#ActTorque := "GVL".ReceivedTelegramData.actualTorque;
```

6. Management of the acyclic data

The acyclic data are non periodic messages that allow to read and write some parameters of the drive.

To manage the acyclic data in a project:

1. In the data block create the hardware ID of the telegram for the acyclic data, that will be used as input of the instructions that allow to read and write the acyclic data.

	Name	Data type	Start value	Retain	Accessible f...	Write...	Visible in ...	Setpoint	Supervis...
1	Static								
2	pHWAP	HW_IO	*IBD60-PNT-CMZ_custom_telegram_200_1-Parameter_Access_Point	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		

Figure 18. Hardware ID of the telegram for the acyclic data

The *start value* is the label of the object that is in: *Devices e networks* → *Network view* → double click on the drive → in the window *Device overview* select *Parameter Access Point* → from the window *System constants* copy the label.

Device overview

Module	Rack	Slot	I address	Q address	Type	Article no.
IBD60-PNT	0	0			IBD60_PROFINET	IBD56xyz/PNT...
Interface	0	0 X1			IBD60-PNT	
CMZ custom telegram 200_1	0	1			CMZ custom telegram 200	
Parameter Access Point	0	1 1			Parameter Access Point	
CMZ custom telegram 200 0	0	1 2	0...29	0...19	CMZ custom telegram 200	

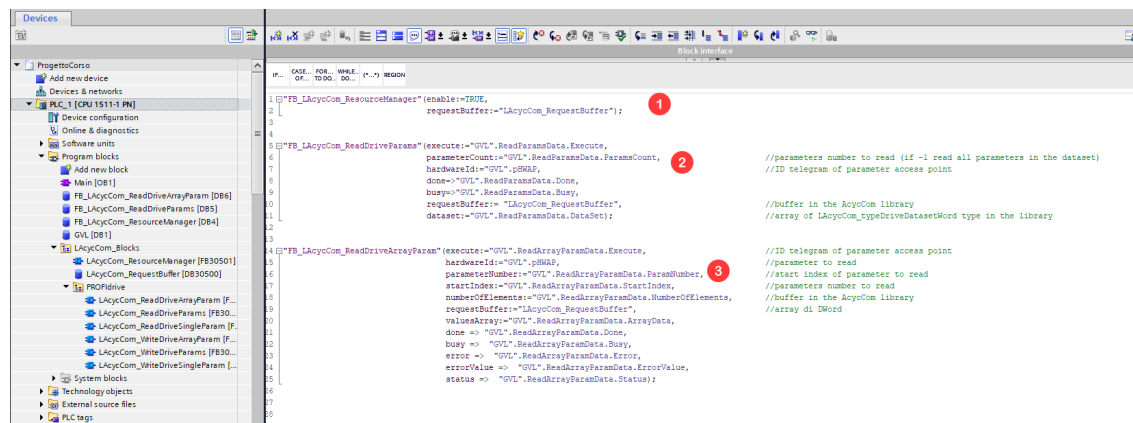
Parameter Access Point [Parameter Access Point]

Name	Type	Hardware identi.	Used by	Comment
IBD60-PNT-CMZ_custom_telegram_200_1-Parameter_Access_Point	Hw_SubModule	266	PLC_1	

Figure 19. Label, object of the telegram

- After the library *LAcycCom* has been downloaded (through the support.industry.siemens.com website) and imported in the project, it is possible to use the following instructions of Siemens to read or write the parameters in the drive:
 - LAcycCom_ReadDriveSingleParam*: reading of a single parameter.
 - LAcycCom_ReadDriveParams*: reading of a limited number of parameters.
 - LAcycCom_ReadDriveArrayParam*: reading of an array of parameters.
 - LAcycCom_WriteDriveSingleParam*: writing of a single parameter.
 - LAcycCom_WriteDriveParams*: writing of a limited number of parameters.
 - LAcycCom_WriteDriveArrayParam*: writing of an array of parameters.

Example of using the instructions to read the parameters



- 1 Instance, by dragging the corresponding object in the program, the instruction *LAcycCom_ResourceManager* (manager of the acyclic data), that requires as input even the buffer *LAcycCom_RequestBuffer* that is present among the blocks of the library.
- 2 read a limited number of parameters (max 39) through the instruction *LAcycCom_ReadDriveParams*.

Refer to the point 2 of the image [Figure 20, “Declaration of the instruction inputs and outputs”](#) for the declaration of the structure that contains the inputs and the outputs of the instruction.

- 3 Read an array of parameters, by inserting the cell to be read, the starting address and the number of parameters to be read (max 234 bytes), through the instruction *LAcycCom_ReadDriveArrayParam*

Refer to the point 3 of the image [Figure 20, “Declaration of the instruction inputs and outputs”](#) for the declaration of the structure that contains the inputs and the outputs of the instruction.

GVL							
	Name	Data type	Start value	Retain	Accessible f...	Writa...	V...
1	Static						
2	ReadParamsData	Struct					
3	Execute	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
4	ParamsCount	Int	1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
5	DataSet	Array[0..2] of *LAcycCom_typeDriveDatasetDWord*			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
6	Busy	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
7	Done	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
8	Error	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
9	Status	Word	16#0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
10	ReadArrayParamData	Struct					
11	Execute	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
12	ParamNumber	UInt	3858		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
13	StartIndex	UInt	0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
14	NumberOfElements	Int	1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
15	ArrayData	Array[0..15] of DWord			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
16	Busy	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
17	Done	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
18	Error	Bool	false		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
19	Status	Word	16#0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
20	ErrorValue	Byte	16#0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Figure 20. Declaration of the instruction inputs and outputs