

IBD e SBD drives con control word e status word in TIA portal

Application note

Doc. TR512004
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	1
2. Creazione nuovo progetto	2
3. Importazione file GSDML	3
4. Importazione e riconoscimento dei devices	4
5. Gestione dati ciclici	8
5.1. Telegramma 200	12
6. Gestione dati aciclici	23

1. Introduzione

Questo manuale offre delle indicazioni sul primo utilizzo dell'ambiente TIA portal con i nostri azionamenti IBD e SBD PROFINET gestiti tramite telegramma 200.

I vari capitoli del manuale sono in sequenza in base a come deve essere creato e gestito il progetto:

1. Creazione di un nuovo progetto;

2. Impostazione del file GSDML;
3. Configurazione della parte hardware;
4. Configurazione della parte software e sviluppo del programma.



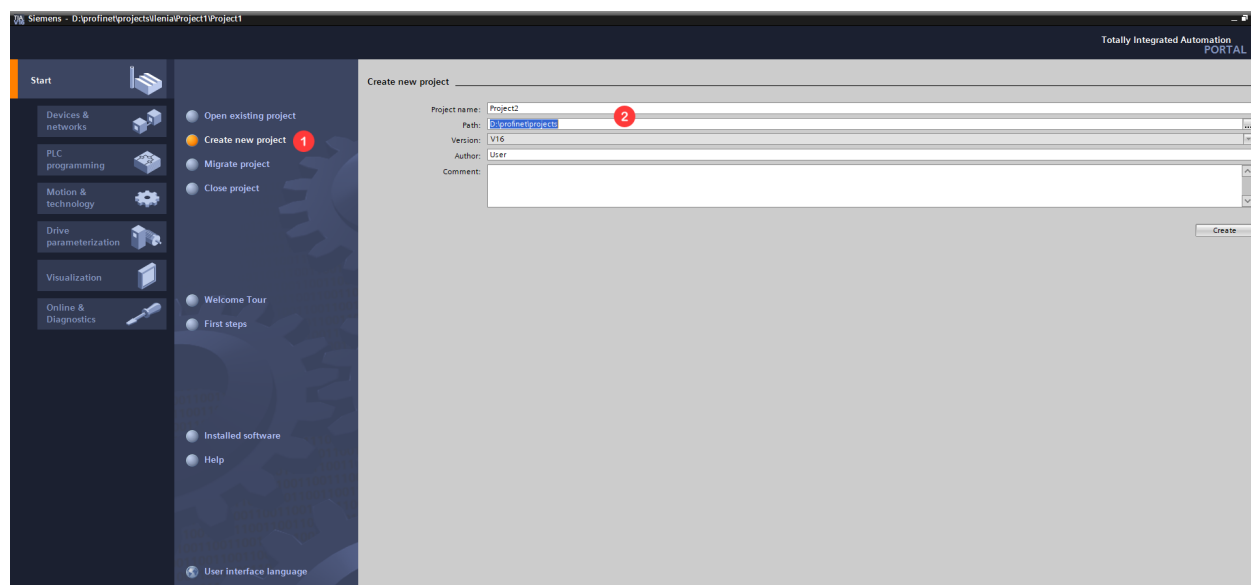
Nota

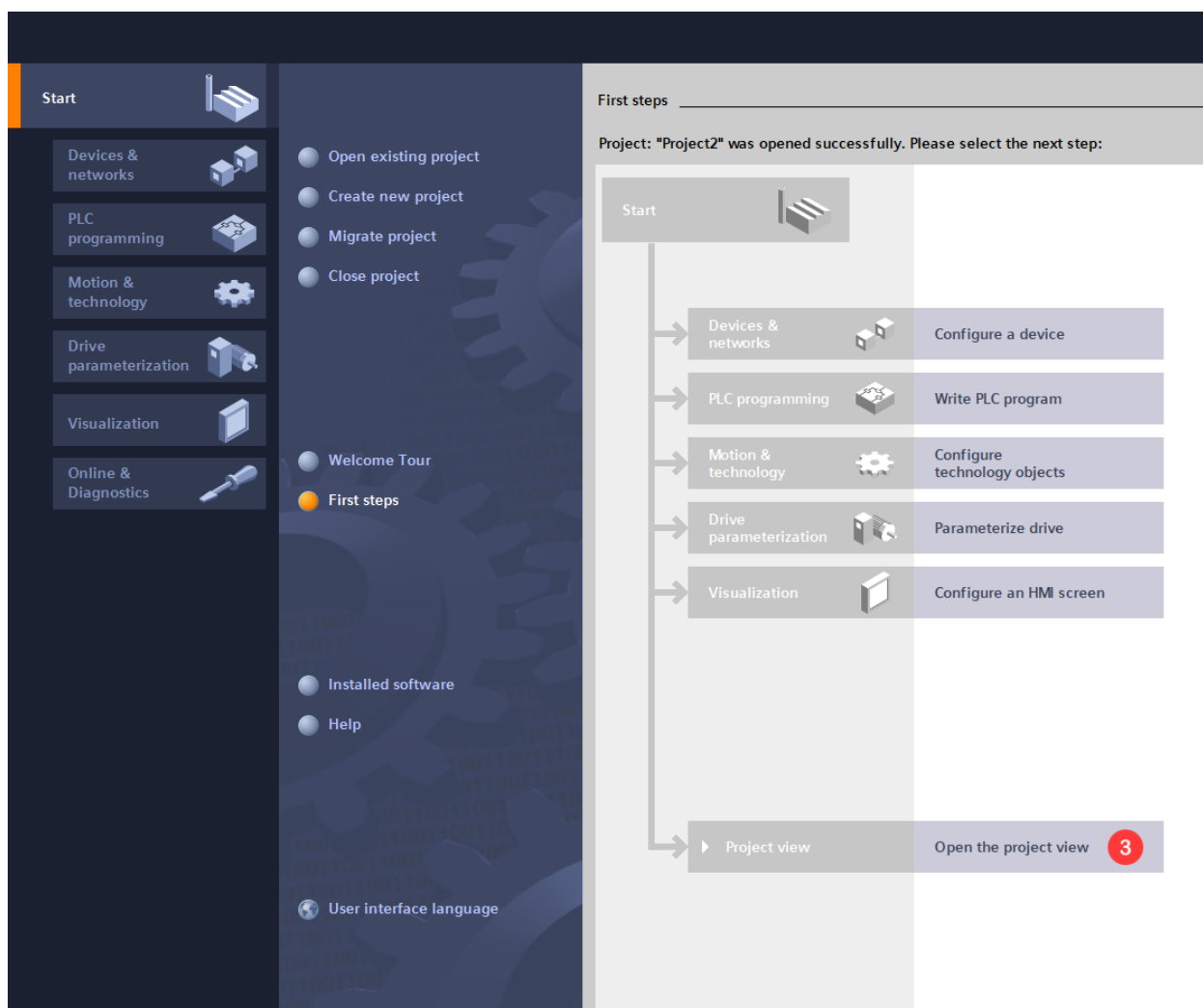
La configurazione del progetto e lo sviluppo della parte software presenti in questo manuale sono state implementate utilizzando:

- il controllore Siemens SIMATIC S7-1500 CPU 1511-1 PN modello 6ES7 511-1AK02-0AB0;
- l'ambiente di sviluppo TIA Portal v16.

2. Creazione nuovo progetto

Per creare un nuovo progetto con TIA portal seguire i passaggi qui sotto descritti:



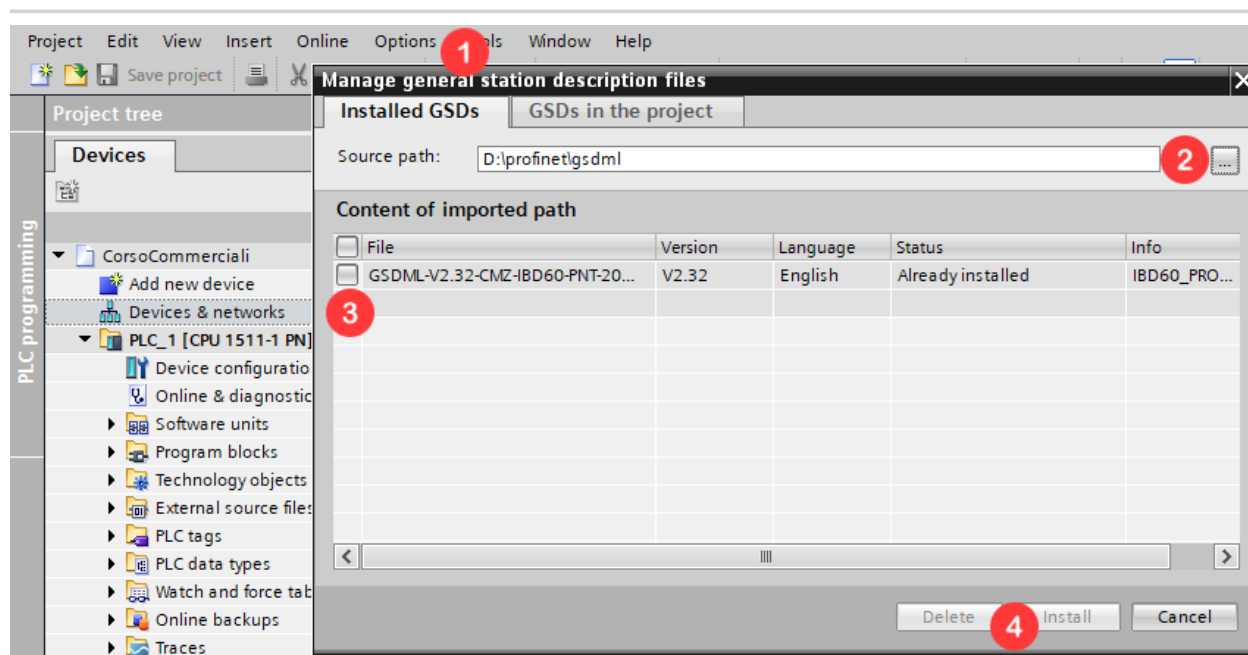


- ① Premere su *Create new project*.
- ② Inserire il nome del progetto e il percorso dove salvarlo.
- ③ Quando il progetto sarà creato, premere *Open the project view* per iniziare a configurare l'hardware e scrivere il software.

3. Importazione file GSDML

Il file GSDML è un file descrittore del device che si vuole utilizzare nell'applicazione e viene distribuito da CMZ.

Per importare il file GSDML è necessario, dopo aver creato il progetto:



1. Dalla barra dei menu premere *Option* → *Manage general station description files(GSD)* .
2. Selezionare il percorso dove si trova il file GSDML da importare.
3. Selezionare il file GSDML da importare.
4. Cliccare su *Install* per installare il file GSDML selezionato.

4. Importazione e riconoscimento dei devices

La configurazione hardware del progetto riguarda l'inserimento e la configurazione nel progetto dei device utilizzati nell'applicazione.

Per configurare la parte hardware seguire i passaggi qui sotto descritti:

1. Dall'albero del progetto fare doppio click su *Device e networks* e si presenteranno 3 tab: *Topology view*, *Network view* e *Device view*.
2. In *Topology view* trascinare, dal catalogo hardware che si trova nella finestra a destra, i dispositivi che servono nell'applicazione:
 - Controllore : Controller → selezionare il modello corretto del controllore.

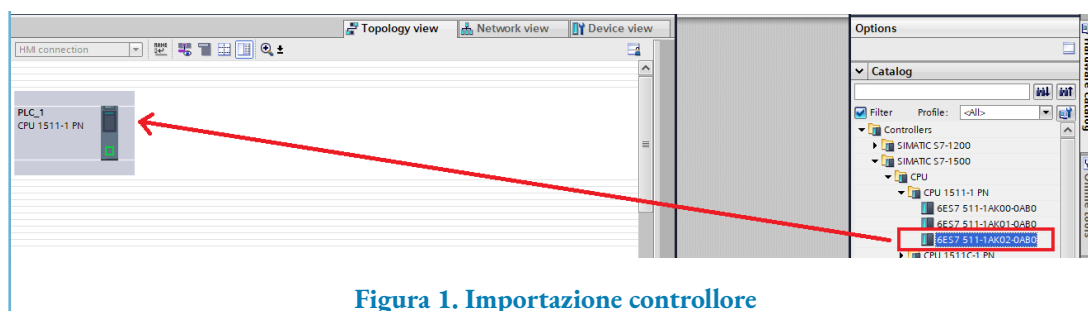


Figura 1. Importazione controllore

- Azionamento : Other filed devices → PROFINET IO → Drives → CMZ Sistemi Elettronici → BD drives → IBD60-PROFINET (per azionamento IBD) oppure SBD-SSD (per azionamento SBD).

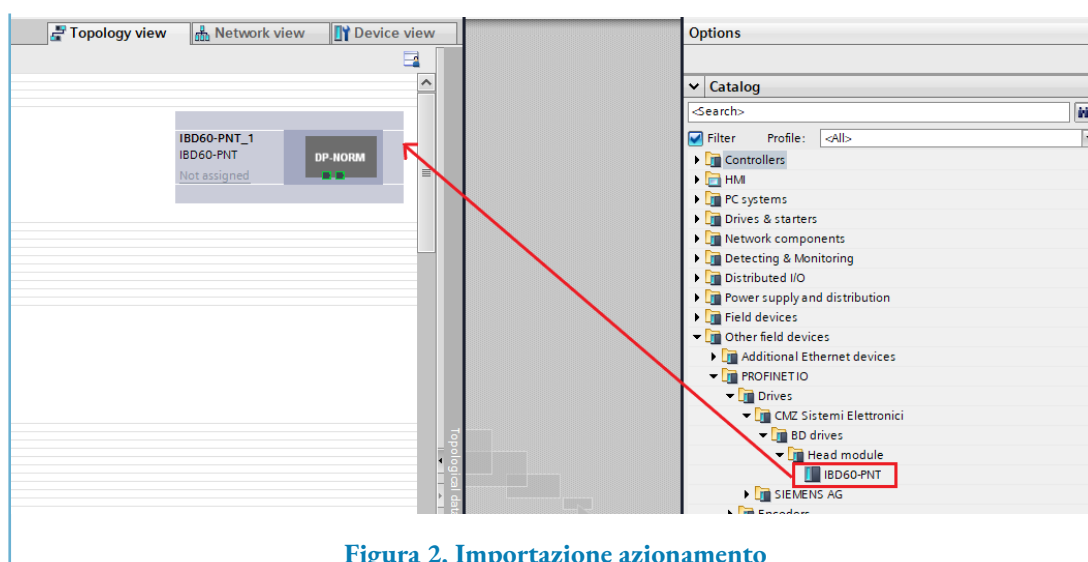


Figura 2. Importazione azionamento

3. In *Topology view* collegare il controllore e l'azionamento tirando il "cavo" dalla porta corretta del controllore alla porta corretta dell'azionamento, in base a come sono collegati fisicamente.

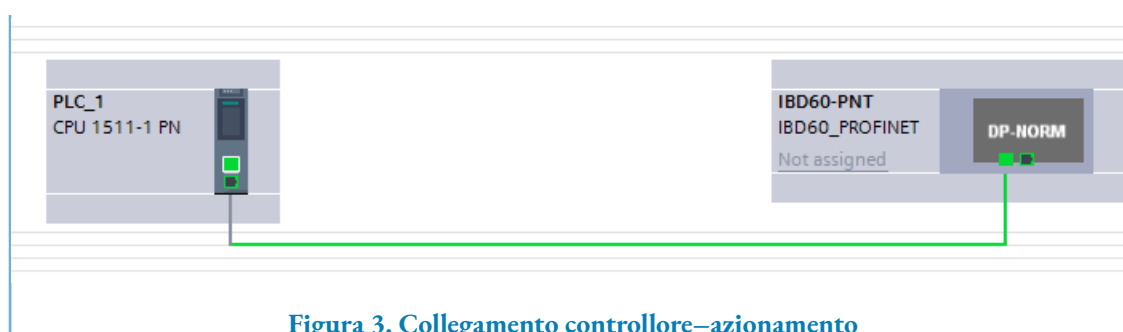


Figura 3. Collegamento controllore–azionamento

4. In *Network view* selezionare l'interfaccia PROFINET (nome del controllore dal quale è comandato l'azionamento) cliccando sulla scritta blu *Not assigned* che compare nel device dell'azionamento e selezionando l'interfaccia corretta.

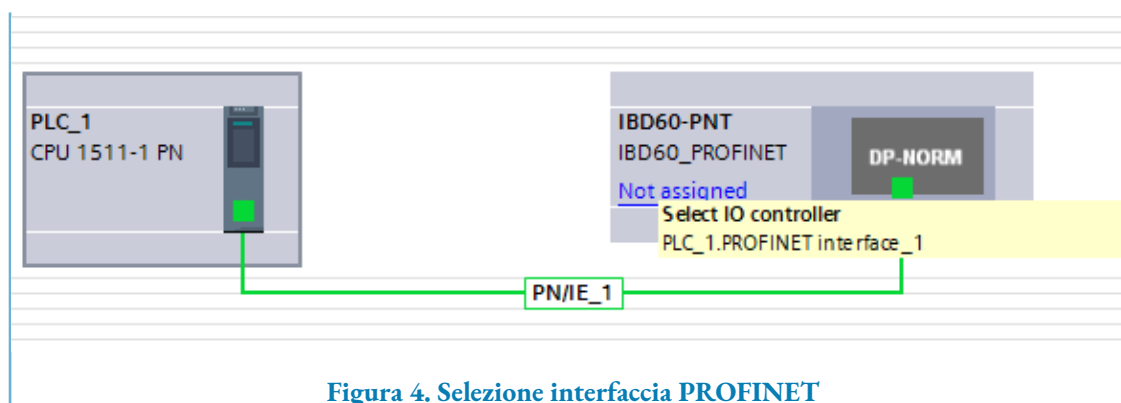


Figura 4. Selezione interfaccia PROFINET

5. In *Network view* fare doppio click sul device del controllore e nella tab *General*:
 - Dalla finestra *Ethernet addresses* impostare nella casella *Ip protocol* l'indirizzo da assegnare al controllore.
 - Dalla finestra *Cycle* impostare il massimo e il minimo tempo di ciclo. Il tempo di ciclo degli oggetti ciclici (OBs) sarà il minimo tempo di ciclo impostato.
6. In *Network view* fare doppio click sul device dell'azionamento e andare ad importare il telegramma che si vuole utilizzare, trascinandolo. Il telegramma si trova nel catalogo hardware sotto la categoria *Module*.

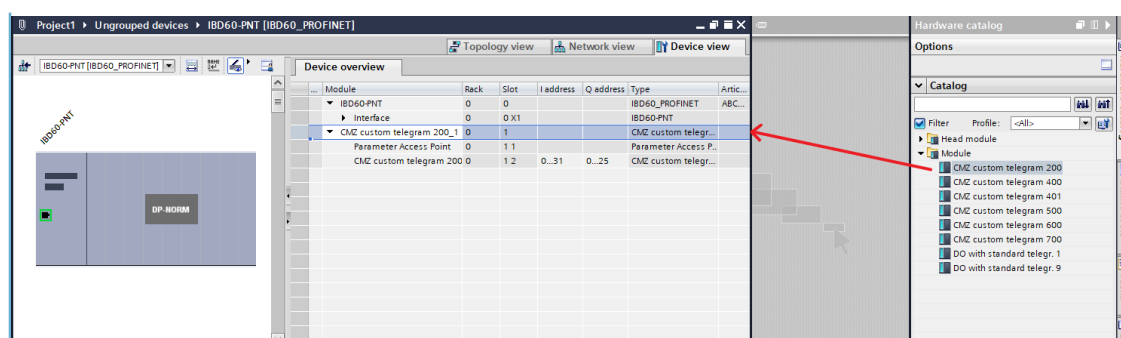
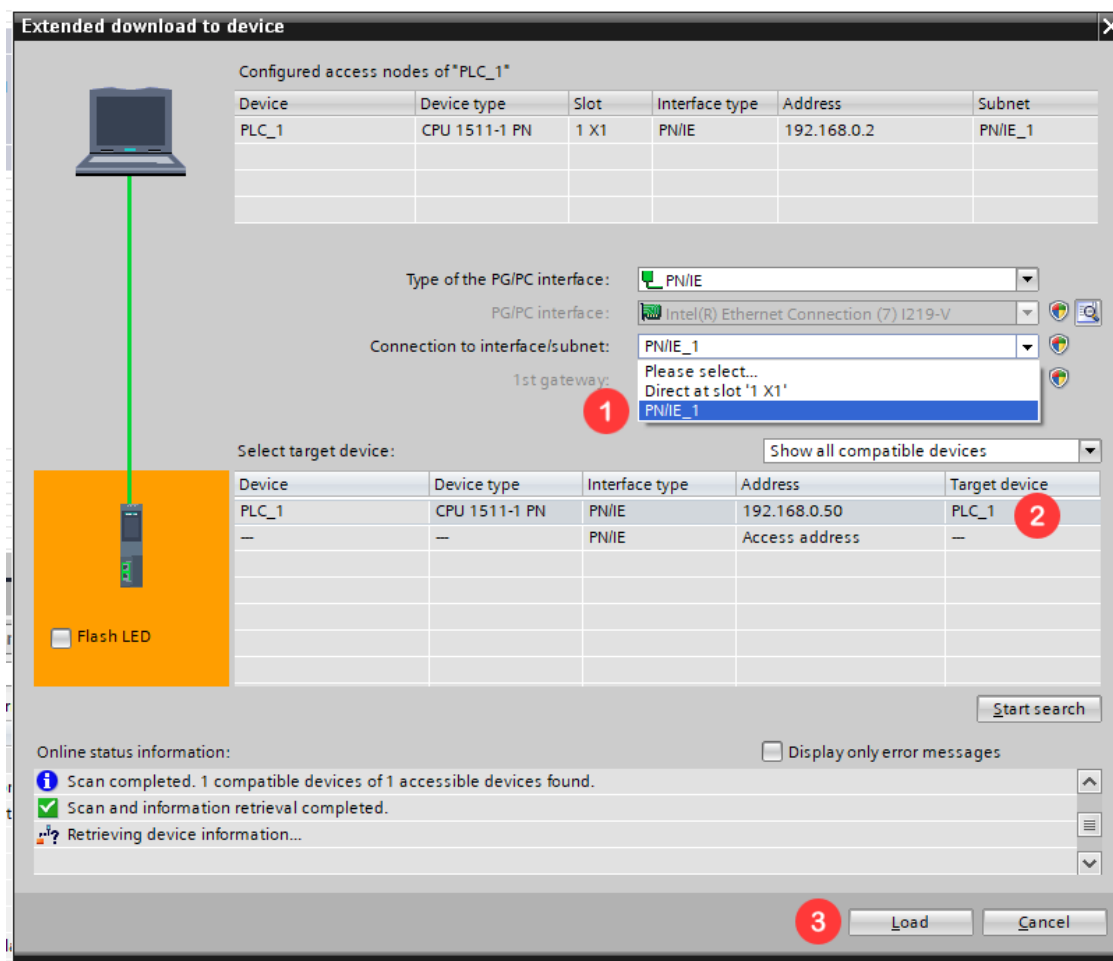


Figura 5. Importazione telegramma

7. Dall'albero del progetto premere tasto destro su *PLC_1[CPU 1511-1 PN]* → *Compile* → *Hardware* per compilare la configurazione hardware. Ripetere la stessa procedura per la parte software.
8. Dall'albero del progetto premere tasto destro su *PLC_1[CPU 1511-1 PN]* → *Download to device* → *Hardware configuration* per scaricare nel controllore la configurazione hardware. Ripetere la stessa procedura per la parte software.

La prima volta che si fa il download bisogna selezionare su quale controllore effettuare l'operazione:



- 1 Selezionare l'interfaccia.
- 2 Premere *Start search* e selezionare il device target sul quale effettuare il download.
- 3 Premere *Load*.

Se la configurazione è corretta e il download è stato effettuato correttamente sia per la parte hardware che software, quando si va in modalità online (tramite il pulsante *Go online* nella barra dei menu), nell'albero del progetto si dovrebbero vedere tutti i pallini verdi come da figura *Figura 6, «Download corretto»*.

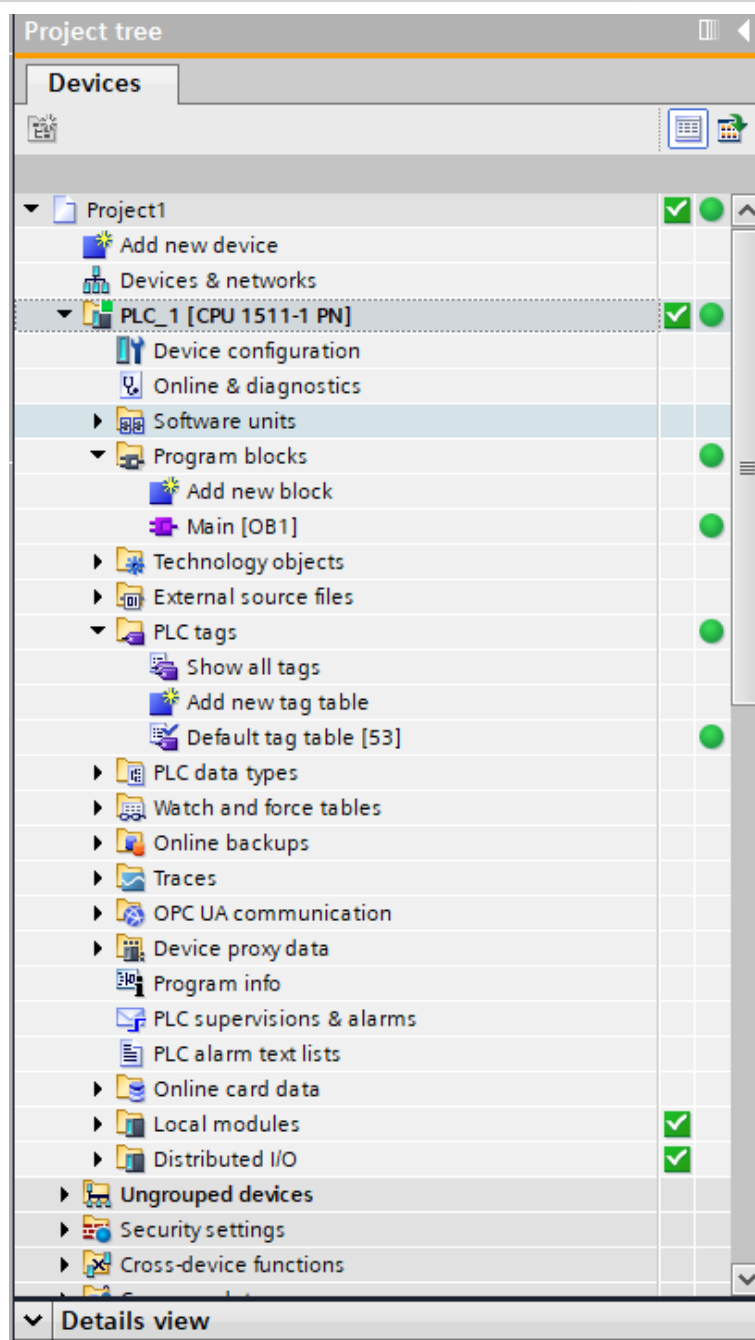


Figura 6. Download corretto

5. Gestione dati ciclici

I dati ciclici sono dei messaggi periodici che vengono scambiati grazie all'uso di telegrammi.

I telegrammi vengono scambiati con cadenza periodica tra master (Controller) e slave (Device), permettendo di gestire l'azionamento in modo real time.

Nei drives IBD e SBD, per la gestione dei dati ciclici è stato implementato il telegramma 200 vedi [Sezione 5.1, «Telegramma 200»](#).

Grazie al telegramma 200 è possibile gestire e comandare l'azionamento gestendo i bit all'interno del telegramma.

Per gestire i dati ciclici nel programma:

1. Creare un blocco dati (tasto destro sopra *Program blocks* → *Add new block* → creare un *Data block*) che conterrà le variabili e strutture comuni a tutti i programmi e dichiarare all'interno:
 - la struttura del frame di output (Controller → Device) che è *SentTelegramData* in questo progetto di esempio:

	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"/>	

Figura 7. Struttura telegramma di output

- la struttura del frame di input (Device → Controller) che è *ReceivedTelegramData* in questo progetto di esempio:

	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"/>	

Figura 8. Struttura telegramma di input

2. E' possibile accedere al telegramma in due modi:

- a. Tramite ID hardware del telegramma, seguendo i passaggi qui sotto descritti:
 - aggiungere nel blocco dati (creato al punto 1) l'ID hardware del telegramma per i dati ciclici: è l'ID del telegramma che viene assegnato automaticamente all'oggetto quando si importa nel device. Questo ID verrà utilizzato come ingresso delle istruzioni che permettono di leggere e scrivere i dati ciclici.

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	pHwTelegram	HW_IO	"IBD60-PNT-CMZ_custom_telegram_200_1-CMZ_custom_telegram_200"		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 9. ID hardware telegramma custom 200

Lo *start value* è l'etichetta dell'oggetto che 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.

Device overview

Module	Rack	Slot	I address	Q address	Type	Article ...
IBD60-PNT	0	0			IBD60_PROFINET	ABCC4...
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	1 2	0..31	0..25	CMZ custom telegram 200	

System constants

Name	Type	Hardware identi.	Used by	Comment
IBD60-PNT-CMZ_custom_telegram_200_1-CMZ_custom_telegram_200	Hw_SubModule	267	FLC_1	

Figura 10. Etichetta oggetto del telegramma

- Per leggere i dati ciclici (frame di input) utilizzare l'istruzione *DPRD_DAT* che richiede come input l'ID hardware del telegramma (pHwTelegram) e come output la struttura da riempire con i dati letti (ReceivedTelegramData).

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

Figura 11. Istruzione DPRD_DAT

- Per inviare i dati ciclici (frame di output) utilizzare l'istruzione *DPWR_DAT* che vuole come input l'ID hardware del telegramma (pHwTelegram) e come output la struttura con i dati da inviare (SentTelegramData).

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

Figura 12. Istruzione DPWR_DAT

- b. Tramite indirizzi seguendo i passaggi qui sotto descritti:
- Tra le proprietà del blocco dati creato al punto 1 (tasto destro sull'oggetto del blocco dati → *Properties*) togliere l'ottimizzazione dei dati all'interno del blocco togliendo l'attributo *Optimized block data*. Togliendo questo attributo è possibile vedere l'indirizzamento dei dati presenti del blocco dati.
 - Utilizzare l'istruzione Siemens *MOVE BLOCK* per:
 - muovere il contenuto del frame di input del telegramma nella struttura del frame di input creato nel blocco dati;
 - muovere il contenuto del frame di output creato nel blocco dati nel frame di output del telegramma.

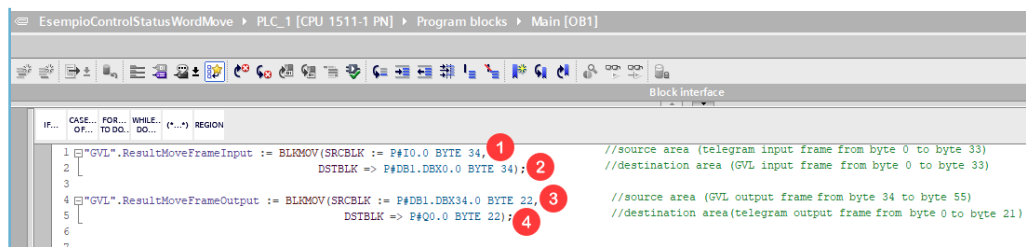


Figura 13. Istruzione MOVE BLOCK su frame di input (1,2) e output(3,4).

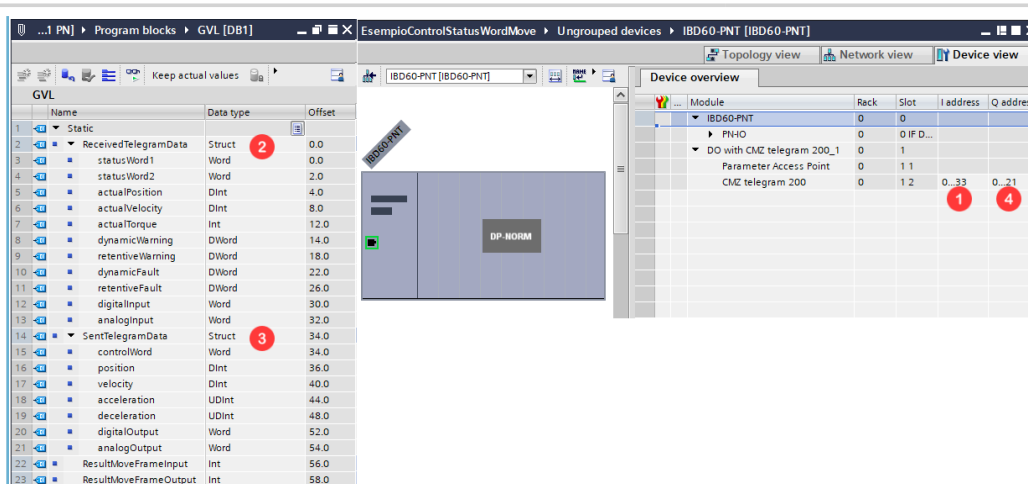


Figura 14. Dove trovare gli indirizzi dei frame del blocco dati (2,3) e gli indirizzi dei frame del telegramma(1,4).

5.1. Telegramma 200

Il telegramma 200 è un messaggio ciclico che permette di gestire l'azionamento in modo real time ed è composto da :

- **Frame di output** mediante il quale il PLC Controller può gestire le uscite digitali e analogiche e far eseguire all'azionamento i seguenti comandi:
 - Messa in coppia.
 - Reset.
 - Procedura di homing parametrizzata dentro l'azionamento.
 - Posizionamenti relativi e assoluti.
 - Movimenti in velocità.
 - Stop.
 - Stop di emergenza.
- **Frame di input** mediante il quale il PLC Controller può leggere dall'azionamento:
 - Stato
 - Posizione attuale

- Velocità attuale
- Coppia attuale
- Warning dinamici e ritentivi
- Fault dinamici e ritentivi
- Ingressi digitali
- Ingresso analogico

5.1.1. Frame di output (Controller → Device)

Il frame di output è composto da:

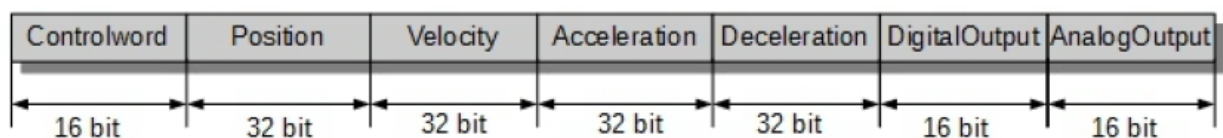


Figura 15. Frame di output (Controller → Device).

- **ControlWord:** è la word che permette di dare i comandi all'azionamento, parametrizzandoli tramite i campi del frame.

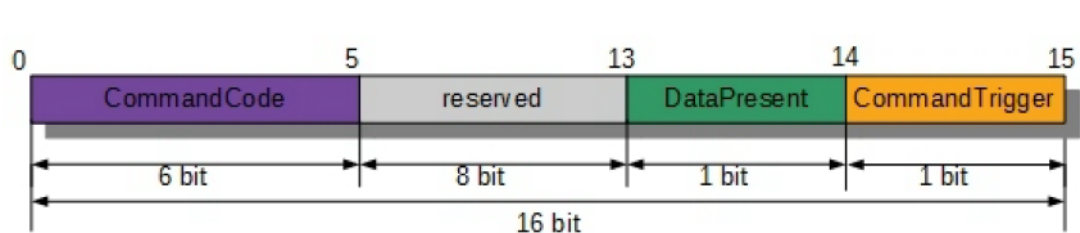


Figura 16. Struttura ControlWord.

Il significato dei bit è il seguente:

- Bit 0-5 : **CommandCode**: codice del comando da eseguire. I comandi implementati sono:

Valore	Nome	Descrizione
1	EmergencyStop	Esegue uno stop del movimento che non può essere interrotto.
2	Stop	Esegue uno stop del movimento che può essere interrotto.
4	PowerOn	Abilita l'asse.
5	PowerOff	Disabilita l'asse.
6	Home	Esegue un homing dell'asse.
7	MoveAbsolute	Esegue un movimento assoluto.
8	MoveRelative	Esegue un movimento relativo.
9	MoveVelocity	Esegue un movimento in velocità.
17	Reset	Esegue un reset dei fault e dei warning.

Tabella 1. Comandi

- Bit 6-13: riservati per uso futuro (da lasciare a 0).
- Bit 14: **DataPresent**: indica che il Controller sta inviando dei dati validi al Device. Dev'essere sempre scritto a 1. Se viene scritto a 0, l'azionamento segnala un fault.
- Bit 15: **CommandTrigger**: sul fronte di salita (0 → 1) di questo bit viene avviato il comando indicato in *CommandCode*.
- **Position**: posizione target per i comandi *MoveRelative* e *MoveAbsolute* e offset per il comando *Home*.
Espresso in incrementi.
- **Velocity**: velocità target per i comandi *MoveRelative*, *MoveAbsolute* e *MoveVelocity*.
Espresso in incrementi/s.
- **Acceleration**: accelerazione per tutti i comandi di movimento tranne per il comando *Home*.
Espresso in incrementi/s².
- **Deceleration**: decelerazione per tutti i comandi di movimento tranne per il comando *Home*.
Espresso in incrementi/s².
- **DigitalOutput**: immagine delle uscite digitali.
- **AnalogOutput**: immagine dell'uscita analogica .

5.1.1.1. Esempi di utilizzo del frame di output (Controller → Device)

Alcuni esempi su come utilizzare il frame di output del telegramma 200 per mandare in esecuzione i comandi tramite control word:

- Abilitazione asse:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //resettare il bit CommandTrigger (15) della
    //control word mettendolo a 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //scrivere 4 nel campo CommandCode e settare a 1 il bit
    //CommandTrigger della control word per abilitare l'asse
    "GVL".SentTelegram.controlWord := 16#C004;
END_CASE;
```

- Disabilitazione asse:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //resettare il bit CommandTrigger (15) della
    //control word mettendolo a 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //scrivere 5 nel campo CommandCode e settare a 1 il bit
    //CommandTrigger della control word per disabilitare
    //l'asse
    "GVL".SentTelegram.controlWord := 16#C005;
END_CASE;
```

- Reset:

```
CASE "GVL".Step OF
```

```
0:
;
1:
//resettare il bit CommandTrigger (15) della
//control word mettendolo a 0
"GVL".SentTelegram.controlWord := 16#4000;
"GVL".Step := "GVL".Step + 1;
2:
//scrivere 17 nel campo CommandCode e settare a 1 il
//bit CommandTrigger della control word per
//resettare l'asse
"GVL".SentTelegram.controlWord := 16#C011;
END_CASE;
```

- Posizionamento assoluto:

```
CASE "GVL".Step OF
0:
;
1:
//resettare il bit CommandTrigger (15) della
//control word mettendolo a 0
"GVL".SentTelegram.controlWord := 16#4000;
"GVL".Step := "GVL".Step + 1;
2:
//paramettrizzare il posizionamento
"GVL".SentTelegram.position := 0;
"GVL".SentTelegram.velocity := 8000;
"GVL".SentTelegram.acceleration := 8000;
"GVL".SentTelegram.deceleration := 8000;
//scrivere 7 nel campo CommandCode e settare a 1 il
//bit CommandTrigger della control word per
//eseguire un posizionamento assoluto
"GVL".SentTelegram.controlWord := 16#C007;
END_CASE;
```

- Posizionamento relativo:

```
CASE "GVL".Step OF
0:
;
```



```
1:
  //resettare il bit CommandTrigger (15) della
  //control word mettendolo a 0
  "GVL".SentTelegram.controlWord := 16#4000;
  "GVL".Step := "GVL".Step + 1;
2:
  //parametrizzare il posizionamento
  "GVL".SentTelegram.position := -8000;
  "GVL".SentTelegram.velocity := 8000;
  "GVL".SentTelegram.acceleration := 8000;
  "GVL".SentTelegram.deceleration := 8000;
  //scrivere 8 nel campo CommandCode e settare a 1 il
  //bit CommandTrigger della control word per
  //eseguire un posizionamento assoluto
  "GVL".SentTelegram.controlWord := 16#C008;
END_CASE;
```

- Movimento in velocità:

```
CASE "GVL".Step OF
  0:
    ;
  1:
    //resettare il bit CommandTrigger (15) della
    //control word mettendolo a 0
    "GVL".SentTelegram.controlWord := 16#4000;
    "GVL".Step := "GVL".Step + 1;
  2:
    //parametrizzare il movimento in velocità
    "GVL".SentTelegram.velocity := 8000;
    "GVL".SentTelegram.acceleration := 8000;
    "GVL".SentTelegram.deceleration := 8000;
    //scrivere 9 nel campo CommandCode e settare a 1 il
    //bit CommandTrigger della control word per
    //eseguire un posizionamento assoluto
    "GVL".SentTelegram.controlWord := 16#C009;
END_CASE;
```

- Stop:

```
CASE "GVL".Step OF
```

```

0:
;
1:
//resettare il bit CommandTrigger (15) della
//control word mettendolo a 0
"GVL".SentTelegram.controlWord := 16#4000;
"GVL".Step := "GVL".Step + 1;
2:
//parametrizzare il comando di stop
"GVL".SentTelegram.deceleration := 8000;
//scrivere 2 nel campo CommandCode e settare a 1 il
//bit CommandTrigger della control word per
//eseguire uno stop all'asse
"GVL".SentTelegram.controlWord := 16#C002;
END_CASE;

```

- Stop di emergenza:

```

CASE "GVL".Step OF
0:
;
1:
//resettare il bit CommandTrigger (15) della
//control word mettendolo a 0
"GVL".SentTelegram.controlWord := 16#4000;
"GVL".Step := "GVL".Step + 1;
2:
//parametrizzare il comando di stop
"GVL".SentTelegram.deceleration := 8000;
//scrivere 1 nel campo CommandCode e settare a 1 il
//bit CommandTrigger della control word per
//eseguire uno stop di emergenza all'asse
"GVL".SentTelegram.controlWord := 16#C001;
END_CASE;

```

- Homing:

```

CASE "GVL".Step OF
0:
;
1:

```

```

//resettare il bit CommandTrigger (15) della
//control word mettendolo a 0
"GVL".SentTelegram.controlWord := 16#4000;
"GVL".Step := "GVL".Step + 1;
2:
//impostare l'offset per la procedura di homing
"GVL".SentTelegram.position := 8000;
//scrivere 6 nel campo CommandCode e settare a 1 il
//bit CommandTrigger della control word per
//eseguire la procedura di homing
"GVL".SentTelegram.controlWord := 16#C006;
END_CASE;

```

5.1.2. Frame di input (Device → Controller)

Il frame di input è composto da:

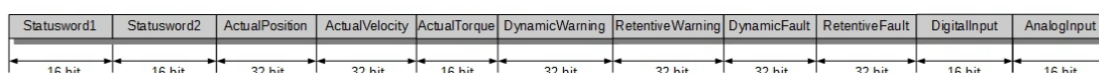


Figura 17. Frame di input (Device → Controller).

- **StatusWord1:** prima word che sintetizza lo stato dell'azionamento.

Il significato dei bit è il seguente:

Bit	Nome
0	ReadyToSwitchOn
1	SwitchedOn
2	Run
3	Fault
4	ErrorStop
5	Stopping
6	StandStill
7	DiscreteMotion
8	ContinuousMotion
9	SynchronizedMotion
10	Homing
11	Initialization
12	ConstantVelocity
13-14	Usa futuro

Bit	Nome
15	CommandTriggerEcho: è la copia (in lettura) del bit 15 della Controlword. È utile come feedback per verificare se il <i>CommandTrigger</i> è stato recepito dall'azionamento.

Tabella 2. StatusWord1

- **StatusWord2:** seconda word che sintetizza lo stato dell'azionamento.

Il significato dei bit è il seguente:

Bit	Nome
0	Accelerating (non gestito)
1	Decelerating (non gestito)
2-15	Uso futuro

Tabella 3. StatusWord2

- **ActualPosition:** posizione attuale dell'asse.
Espressa in incrementi.
- **ActualVelocity:** velocità attuale dell'asse.
Espresso in incrementi/s.
- **ActualTorque:** coppia attuale dell'asse.
Espresso in millesimi della corrente nominale.
- **DynamicWarning:** maschera a bit dei warning dinamici.
- **RetentiveWarning:** maschera a bit dei warning ritentivi.
- **DynamicFault:** maschera a bit dei fault dinamici.
- **RetentiveFault:** maschera a bit dei fault ritentivi.
- **DigitalInput:** immagine degli ingressi digitali.
- **AnalogInput:** immagine dell'ingresso analogico.

5.1.2.1. Esempi di utilizzo del frame di input (Device → Controller)

Alcuni esempi su come utilizzare il frame di input del telegramma 200:

- Lettura bit status word1:

```
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;
```

- Lettura posizione, velocità, coppia attuali dell'azionamento:

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

6. Gestione dati aciclici

I messaggi aciclici sono dei messaggi non periodici che permettono di leggere e scrivere alcuni parametri del drive.

Per gestire i dati aciclici in un progetto:

1. Nel blocco dati creare l'ID hardware del telegramma per i dati aciclici che verrà utilizzato come ingresso delle istruzioni che permettono di leggere e scrivere i dati aciclici.

GVL									
	Name	Data type	Start value	Retain	Accessible f...	Writa...	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"/>		

Figura 18. ID hardware del telegramma per i dati aciclici

Lo *start value* è l'etichetta dell'oggetto che si trova in: *Devices e networks* → *Network view* → doppio click sull'azionamento → nella finestra *Device overview* selezionare *Parameter Access Point* → dalla finestra *System constants* copiare l'etichetta.

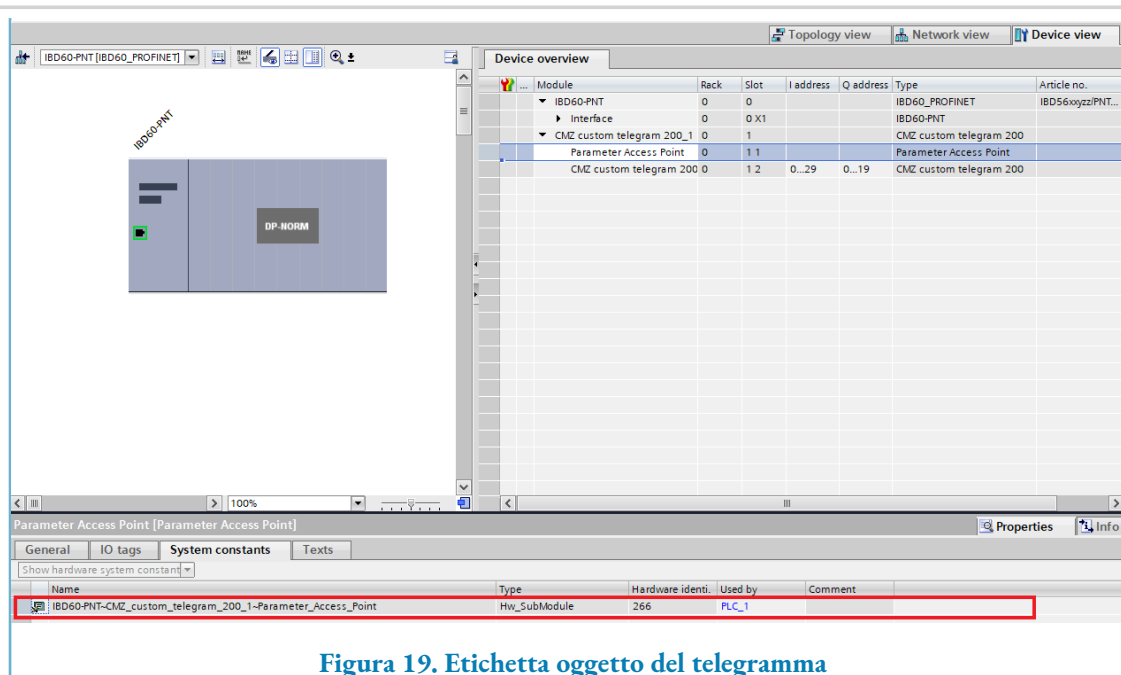
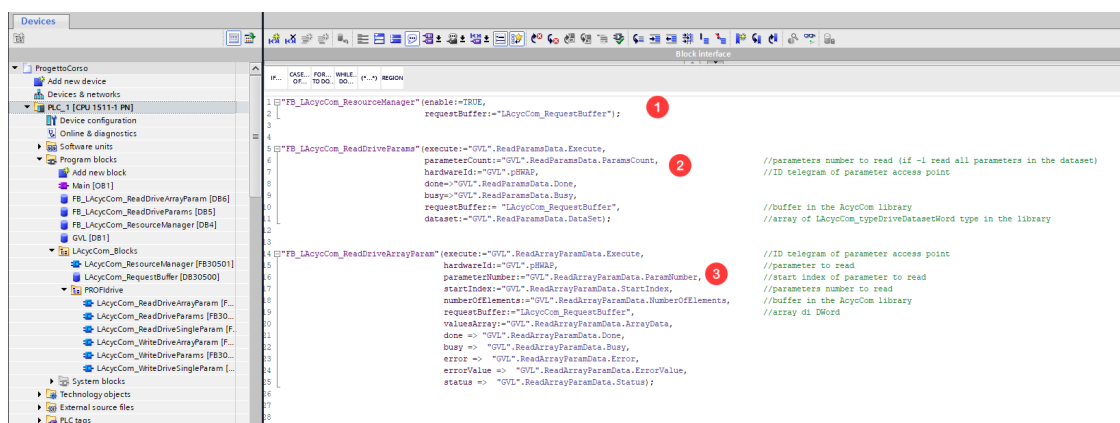


Figura 19. Etichetta oggetto del telegramma

2. Dopo aver scaricato la libreria *LAcycCom* (tramite il sito support.industry.siemens.com) ed averla importata nel progetto, è possibile utilizzare le seguenti istruzioni Siemens per leggere o scrivere i parametri nell'azionamento:

- *LAcycCom_ReadDriveSingleParam*: lettura di un singolo parametro.
- *LAcycCom_ReadDriveParams*: lettura di un numero limitato di parametri.
- *LAcycCom_ReadDriveArrayParam*: lettura di un array di parametri.
- *LAcycCom_WriteDriveSingleParam*: scrittura di un singolo parametro.
- *LAcycCom_WriteDriveParams*: scrittura di un numero limitato di parametri.
- *LAcycCom_WriteDriveArrayParam*: scrittura di un un array di parametri.

Esempio di utilizzo delle istruzioni per leggere parametri



1 Istanziare, trascinando il corrispondente oggetto nel programma, l'istruzione *LAcycCom_ResourceManager* (gestore dei dati aciclici), che richiede come ingresso anche il buffer *LAcycCom_RequestBuffer* presente tra i blocchi della libreria.

2 Leggere un numero limitato di parametri (max 39) tramite l'istruzione *LAcycCom_ReadDriveParams*.

Fare riferimento al punto 2 nell'immagine *Figura 20, «Dichiarazione ingressi e uscite istruzioni»* per la dichiarazione della struttura contenente ingressi e uscite dell'istruzione.

3 Leggere un array di parametri, inserendo la cella da leggere, l'indice di partenza e il numero di parametri da leggere (max 234 bytes), tramite l'istruzione *LAcycCom_ReadDriveArrayParam*

Fare riferimento al punto 3 nell'immagine *Figura 20, «Dichiarazione ingressi e uscite istruzioni»* per la dichiarazione della struttura contenente ingressi e uscite dell'istruzione.

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

Figura 20. Dichiarazione ingressi e uscite istruzioni

