Functional Description Plug&Drive-Interface

Extension of the technical manual of following products:

C5, C5-E, N5, CL3-E, CL4-E, NP5, PDx-C, PDx-E

Valid with firmware version FIR-v1825 Functional Description Version: 1.0.0

Contents

1 Introduction

With the *Plug & Drive interface* (*PDI*), you can realize simple movements with Nanotec drives.
The interface is supported by all controllers and Plug & Drive motors and is part of their firmware, independent of the present communication interface.

This document describes the operating modes and functions of the *Plug & Drive interface* as well as the implementation on the fieldbus. For information on commissioning and operating the respective Nanotec product, use the corresponding technical manual.

Observe the safety and warning notices as well as the notes regarding the use of the Nanotec products and the warranty. You can find these at **us.nanotec.com** in the technical manuals of the products.

1.1 Version information

Manual version	Date	Changes	Firmware version
1.0.0	06/2018	First edition	FIR-v1825

1.2 Copyright, marking and contact

Nanotec Electronic GmbH & Co. KG

Kapellenstraße 6

85622 Feldkirchen

Germany

Phone: +49 89 900 686-0

Fax: +49 (89) 900 686-50

us.nanotec.com

1.3 Notes and used icons

All notices are in the same format. The degree of the hazard is divided into the following classes.

VORSICHT

The CAUTION notice indicates a possibly dangerous situation.

Failure to observe the notice **may** result in moderately severe injuries.

► Describes how you can avoid the dangerous situation.

Hinweis

- Indicates an error source or likelihood of confusion.
- Failure to observe the notice **may** result in damage to this or other devices.
- Describes how device damage can be avoided.

Tipp
Shows a tip for the application or task.

1.4 Emphasis in the text

The following conventions are used in the document:

Text set in *italics* marks named objects:

- Read the *installation manual*.
- Use the *Plug & Drive Studio* software to perform the auto setup.
- For software: You can find the corresponding information in the *Operation* tab.

A text in "quotation marks" marks user input:

- Start the NanoJ program by writing object 2300_h, bit 0 = "1".
- Write the value "$203B_h$" in 2291_h:02_h (*PDI-SetValue2*).

1.5 Numerical values

Numerical values are generally specified in decimal notation. The use of hexadecimal notation is indicated by a subscript *h* at the end of the number.

The objects in the object dictionary are written with index and subindex as follows: `<Index>:<Subindex>`

Both the index as well as the subindex are specified in hexadecimal notation. If no subindex is listed, the subindex is 00_h.

Example: Subindex 5 of object 1003_h is addressed with 1003_h:05_h, subindex 00 of object 6040_h with 6040_h.

2 Description of the interface

The *Plug & Drive interface* represents a Nanotec-specific variant for controlling a drive and offers an alternative to the *device profile*, which is described in CiA 402.

With the *Plug & Drive interface*, you can immediately trigger drive commands. This eliminates the need to run through the *state machine*.

2.1 Structure of the used objects

The *Plug & Drive interface* uses two objects (*PDI objects*) that consist of six entries. An additional object is used to activate the *Plug & Drive interface*.

2.1.1 Object for activating the *Plug & Drive interface*

Object 2290_h:00_h (*PDI-Ctrl*) can be used to activate the *Plug & Drive interface*. To activate the *Plug & Drive interface*, set bit 0 to "1".

Object name	Object	Data type	Description
PDI-Ctrl	2290_h:00_h	UNSIGNED08	Bit 0: • Value = "0": *Plug & Drive interface* deactivated • Value = "1": *Plug & Drive interface* activated

In the factory settings for the controllers, the *Plug & Drive interface* is already activated.

2.1.2 PDI objects

With the *Plug & Drive interface*, the complete interface functionality is mapped via two record objects. Use object 2291_h to enter the parameters. Object 2292_h is used by the controller to output values.

With object 2291_h, you can select and start the operating mode and set the corresponding target values (target position, speed, etc.). To read/write objects, you can also use this object to access the entire object dictionary (all parameters) of the controller. The object includes the control command (*PDI-Cmd*) and three entires (*PDI-SetValue*), which are used in different ways in the individual operating modes.

Object 2292_h includes the status (*PDI-Status*) and an output value (*PDI-ReturnValue*), which is dependent on the used operating mode (e.g., the value of an object or a current actual value).

All entries of both objects use the SIGNED data type.

The following table lists all entries.

Object name	Object address	Data type	Description
PDI-SetValue1	2291_h:01_h	SIGNED32	Is used differently in the individual operating modes, primarily for target values
PDI-SetValue2	2291_h:02_h	SIGNED16	Is used differently in the individual operating modes
PDI-SetValue3	2291_h:03_h	SIGNED08	Is used for reading and writing objects as well as for setting the homing methods
PDI-Cmd	2291_h:04_h	SIGNED08	Functions as a universal control command

Object name	Object address	Data type	Description
PDI-Status	2292_h:01_h	SIGNED16	Returns the status of the controller as bit mask
PDI-ReturnValue	2292_h:02_h	SIGNED32	Returns an output value which is dependent on the used operating mode (e.g., the value of an object or a current actual value)

2.2 Control command *PDI-Cmd* (2291_h:04_h)

With *PDI-Cmd*, you can execute the commands listed in the following table. To execute a specific command, enter the corresponding value in object 2291_h:04_h. The command is executed immediately.

Command	Value	Description
NOP (No Operation)	0	Executes nothing
Switch Off	1	Switches off the control; the power stage is disabled and the motor is de-energized.
Clear Error	2	Clears a possible error, provided the cause of the error has since been rectified
Quickstop	3	Initiates a quickstop of the motor; the motor brakes with the quickstop ramp to speed 0.
OD-Read	14	Reads out an object
OD-Write	15	Writes a value in an object
Auto setup	16	Starts the auto setup
Homing on current position	17	Sets the current position to an arbitrary value
Homing	18	Starts a homing operation
Profile Position absolute	20	Starts an absolute positioning operation
Profile Position relative	21	Starts a relative positioning operation
Profile Velocity	23	Activates velocity mode
Profile Torque	25	Activates torque mode
Halt Bit	Bit 6 (64)	Uses a ramp to brake the drive to speed 0 and is supported by the following commands: 20, 21, 23 and 25
ToggleCmd Bit	Bit 7 (128)	By setting/resetting (toggling) the bit, you can execute a command several times in sequence.

2.2.1 NOP (No Operation) command – PDI-Cmd 0

This command does not execute anything, but instead deletes the existing command. The *PdiStatusToggleCmd* bit is still activated (toggled).

2.2.2 Switch Off command – PDI-Cmd 1

This command blocks the power stage of the drive. The control switches off and the motor is de-energized. An existing travel command is interrupted and the control deviation is deleted.

Tip

To switch the drive to control operation without moving it, you can send a command 23 (*Profile Velocity*) with target speed "0" or start a positioning operation (command 20) at the current position.

2.2.3 Clear Error command – PDI-Cmd 2

With this command, you clear a possible error, provided the cause of the error has since been rectified.

With *PDI-Status*, you can read on bit *PdiStatusFault* whether there is currently an error.

2.2.4 Quickstop command – PDI-Cmd 3

With this command, you initiate a quickstop of the motor. For further information, see the technical manual of the controller, chapter *CiA 402 Power State Machine*.

The motor brakes to speed 0. For this transition, you can set one of the following options in object $605A_h$:

Value in object $605A_h$	Description
0	Immediate stop with subsequent state change to *Switch on disabled*
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.
6	Braking with *quick stop ramp* and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

2.2.5 OD-Read command – PDI-Cmd 14

With this command, you can read out an arbitrary object from the object dictionary of the controller.

You write the index of the object in $2291_h:02_h$ (*PDI-SetValue2*) and the subindex in $2291_h:03_h$ (*PDI-SetValue3*).

Parameter	Object name	Object
Index	PDI-SetValue2	$2291_h:02_h$
Subindex	PDI-SetValue3	$2291_h:03_h$

Then send command *OD-Read*.

PDI-ReturnValue ($2292_h:02_h$) returns the value of the object. In case of an error, the controller resets bit 15 (*PDIStatusError*) of the *PDI-Status* ($2292_h:01_h$) and stores the error code in *PDI-ReturnValue*.

Example

To read out object $3320_h:01_h$ (analog input 1), proceed as follows:

1. Write the value "13088 (3320_h)" in $2291_h:02_h$ (*PDI-SetValue2*).

2. Write the value "1" in 2291_h:03_h (*PDI-SetValue3*).
3. Send command *OD-Read* by writing the value "14" in 2291_h:04_h (*PDI-Cmd*).

You can read the current value of the analog input in *PDI-ReturnValue* (2292_h:02_h).

To read out a second value, you must toggle the *ToggleCmd bit*.

2.2.6 OD-Write command – PDI-Cmd 15

With this command, you can write a value in an object of the object dictionary of the controller.

You write the index of the object in 2291_h:02 (*PDI-SetValue2*) and the subindex in 2291_h:03_h (*PDI-SetValue3*). Enter the value in 2291_h:01_h (*PDI-SetValue1*).

Parameter	Object name	Object
Index	PDI-SetValue2	2291_h:02_h
Subindex	PDI-SetValue3	2291_h:03_h
Wert	PDI-SetValue1	2291_h:01_h

Then send command *OD-Write*.

In case of an error, the controller resets bit 15 (*PDIStatusError*) of the *PDI-Status* (2292_h:01_h) and stores the error code in *PDI-ReturnValue*.

Example

To write the value "1000" in object $203B_h$:01_h (set rated current to 1000 mA), proceed as follows:

1. Write the value "8251 ($203B_h$)" in 2291_h:02_h (*PDI-SetValue2*).
2. Write the value "1" in 2291_h:03_h (*PDI-SetValue3*).
3. Write the value "1000" in 2291_h:01_h (*PDI-SetValue1*).
4. Send command *OD-Write* by writing the value "15" in 2291_h:04_h (*PDI-Cmd*).

To write a second value, you must toggle the *ToggleCmd bit*.

2.2.7 Auto setup command – PDI-Cmd 16

With this command, you start the auto setup. For further information, see the technical manual of the controller, chapter *Auto setup*.

Bit *PdiStatusAutosetupDone* in *PDI-Status* (2292_h:01_h) signals the end of the auto setup. If you send a new command, this bit remains set.

CAUTION

Uncontrolled motor movements!

After the auto setup, the internal coordinate system is no longer valid. Unforeseen reactions can result.

► Restart the device after an auto setup. Homing alone does not suffice.

2.2.8 Homing on current position command – PDI-Cmd 17

With this command, you can set the current actual position to an arbitrary value. You can thereby freely program homing operations if, for example, it is not possible to connect a home switch. After switching on the controller voltage, you can also write a previously stored position value in the drive.

You enter the desired actual value in $2291_h:01_h$ (*PDI-SetValue1*) and execute the command by writing the value "17" in $2291_h:04_h$ (*PDI-Cmd*).

Bit *PdiStatusHomingDone* in *PDI-Status* ($2292_h:01_h$) signals the end of the homing operation. If you send a new command, this bit remains set.

2.2.9 Homing command – PDI-Cmd 18

With this command, you can execute a homing operation. For further information on the homing mode and the homing methods, see the technical manual of the controller, chapter *Homing*.

You enter the desired homing method in $2291_h:03_h$ (*PDI-SetValue3*) and execute the command by writing the value "18" in $2291_h:04_h$ (*PDI-Cmd*).

Bit *PdiStatusHomingDone* in *PDI-Status* ($2292_h:01_h$) signals the end of the homing operation. If you send a new command, this bit remains set.

2.2.10 Profile Position absolute command – PDI-Cmd 20

With this command, you start an absolute positioning operation. For further information on this operating mode, see the technical manual of the controller, chapter *Profile Position*.

You can start a new travel command at any time by sending command *NOP* and again sending command 20 or by toggling the *ToggleCmd bit*. The drive implements the movement immediately.

Enter the target position in $2291_h:01_h$ (*PDI-SetValue1*) and the maximum speed in $2291_h:02_h$ (*PDI-SetValue2*).

Parameter	Object name	Object
Target position in user-defined units (factory setting: tenths of degree)	PDI-SetValue1	$2291_h:01_h$
Maximum speed in user-defined units (factory setting: revolutions per minute)	PDI-SetValue2	$2291_h:02_h$

Then send command *Profile Position absolute* (20).

Once the target position is reached, the controller sets the *PdiStatusTargetReached* bit in *PDI-Status* ($2292_h:01_h$).

The controller stores the current position in *PDI-ReturnValue*.

Example

To start a movement to position 2000 with a speed of 200, proceed as follows:

1. Write the value "2000" in $2291_h:01_h$ (*PDI-SetValue1*).
2. Write the value "200" in $2291_h:02_h$ (*PDI-SetValue2*).
3. Send command *Profile Position absolute* by writing the value "20" in $2291_h:04_h$ (*PDI-Cmd*).

Note

Entry *PDI-SetValue2* is limited to 16 bits. Take this into account when selecting the speed unit. For details, see chapter *User-defined units* in the technical manual of the controller.

2.2.11 Profile Position relative command – PDI-Cmd 21

With this command, you start a relative positioning operation. For further information on this operating mode, see the technical manual of the controller, chapter *Profile Position*.

You can start a new travel command at any time by sending command *NOP* and again sending command 21 or by toggling the *ToggleCmd bit*. The drive implements the movement immediately.

Enter the target position in 2291_h:01_h (*PDI-SetValue1*) and the maximum speed in 2291_h:02_h (*PDI-SetValue2*).

Parameter	Object name	Object
Target position in user-defined units (factory setting: tenths of degree)	PDI-SetValue1	2291_h:01_h
Maximum speed in user-defined units (factory setting: revolutions per minute)	PDI-SetValue2	2291_h:02_h

Then send command *Profile Position relative* (21).

Once the target position is reached, the controller sets the *PdiStatusTargetReached* bit in *PDI-Status* (2292_h:01_h).

The controller stores the current position in *PDI-ReturnValue*.

Example

To start a movement to position 2000 with a speed of 200, proceed as follows:

1. Write the value "2000" in 2291_h:01_h (*PDI-SetValue1*).
2. Write the value "200" in 2291_h:02_h (*PDI-SetValue2*).
3. Send command *Profile Position relative* by writing the value "21" in 2291_h:04_h (*PDI-Cmd*).

Note

Entry *PDI-SetValue2* is limited to 16 bits. Take this into account when selecting the speed unit. For details, see chapter *User-defined units* in the technical manual of the controller.

2.2.12 Profile Velocity command – PDI-Cmd 23

With this command, you activate the velocity mode (*Profile Velocity*). For further information on this operating mode, see the technical manual of the controller, chapter *Profile Velocity*.

Enter the target speed in 2291_h:01_h (*PDI-SetValue1*).

Parameter	Object name	Object
Target speed in user-defined units (factory setting: revolutions per minute)	PDI-SetValue1	2291_h:01_h

Then send command *Profile Velocity* (23).

The controller stores the current speed in *PDI-ReturnValue*.

Example

To start a movement with a target speed of 300, proceed as follows:

1. Write the value "300" in 2291_h:01_h (*PDI-SetValue1*).
2. Send command *Profile Velocity* by writing the value "23" in 2291_h:04_h (*PDI-Cmd*).

2.2.13 Profile Torque command – PDI-Cmd 25

With this command, you activate the torque mode (*Profile Torque*). For further information on this operating mode, see the technical manual of the controller, chapter *Profile Torque*.

Enter the target torque in 2291_h:01_h (*PDI-SetValue1*) and the maximum speed in 2291_h:02_h (*PDI-SetValue2*).

Parameter	Object name	Object
Target torque in thousandths of the rated torque	PDI-SetValue1	2291_h:01_h
Maximum speed in user-defined units (factory setting: revolutions per minute)	PDI-SetValue2	2291_h:02_h

Then send command *Profile Torque* (25).

Note

You must first configure the ramp for the torque (object 6087_h, Torque Slope), since the default value is "0".

The controller stores the current torque in *PDI-ReturnValue*.

Example

To start a movement with 50% of the rated torque and a maximum speed of 100, proceed as follows:

1. Write the value "100" in 2291_h:02_h (*PDI-SetValue2*).
2. Write the value "500" in 2291_h:01_h (*PDI-SetValue1*). You thereby set the target torque to 50% of the rated current.
3. Send command *Profile Torque* by writing the value "25" in 2291_h:04_h (*PDI-Cmd*).

2.2.14 Halt bit – bit 6 (PDI-Cmd 64)

If you set the *halt bit*, the motor brakes with a ramp to speed 0. The control remains active and the motor energized. The *halt bit* is supported for all travel commands (*Profile Position (absolute/relative)*, *Profile Velocity*, *Profile Torque*).

You must add the value 64 (bit 6) to the value of the travel command and write the result in 2291_h:04_h (*PDI-Cmd*).

You have two options for continuing the movement:

- Continue the movement with the previously set target values: to do this, you must reset the *halt bit*.
- Set a new target value that is to be taken over upon resetting of the *halt bit*: to do this, you must reset the *halt bit* and simultaneously toggle the *ToggleCmd bit*.

Example

To stop an absolute positioning operation (command *Profile Position absolute*, 20) and again continue with the same target position and target speed, proceed as follows:

1. Write the value "84" in 2291_h:04_h (*PDI-Cmd*).
   The motor brakes.
2. Write the value "20" in 2291_h:04_h (*PDI-Cmd*).
   The motor continues to the previously set target position.

To stop the absolute positioning operation and continue with a new target position, proceed as follows:

1. Write the value "84" in 2291_h:04_h (*PDI-Cmd*).
   The motor brakes.
2. Enter the new target position in 2291_h:01_h (*PDI-SetValue1*).
3. Write the value "148" in 2291_h:04_h (*PDI-Cmd*).
   The *halt bit* is thereby reset and the *ToggleCmd bit* toggled. The motor continues to the new target position.

2.2.15 ToggleCmd bit – bit 7 (PDI-Cmd 128)

By toggling (setting/resetting) the bits of the *ToggleCmd bit*, you can execute a command several times in sequence. Each toggling of the *ToggleCmd bit* automatically causes the *PdiStatusToggleCmd* bit in *PDI-Status* (2292_h:01_h) to switch.

Example

To execute a relative positioning operation (command *Profile Position relative*) three times in sequence, proceed as follows:

1. Write the value "21" in 2291_h:04_h (*PDI-Cmd*).
   The first movement is started.
2. If desired, change the target position or target speed.
3. To start the second movement, write the value "149" in 2291_h:04_h (*PDI-Cmd*).
4. To start the third movement, write the value "21" in 2291_h:04_h (*PDI-Cmd*).

2.3 *PDI-Status* (2292_h:01_h)

The *PDI-Status* returns the status of the drive and of the *PDI interface* as a bitmask.

The *PDI-Status* uses the following bits:

Bit 0: *PdiStatusOperationEnabled*

This bit is set if the drive is in the *Operation enabled* state. The drive is energized in this state.

Bit 1: *PdiStatusWarning*

This bit is set if warnings occur in the drive; you can read out the warning in object $603F_h:00_h$.

Bit 2: *PdiStatusFault*

This bit is set if errors occur in the drive; you can read out the error code in object $603F_h:00_h$.

Bit 3: *PdiStatusTargetReached*

This bit is set if the target value (position, speed) is reached.

Bit 4: *PdiStatusFollowingError*

This bit is set if a following error occurred in *Profile Position* mode or a slippage error occurred in *Profile Velocity* mode.

Bit 5: *PdiStatusLimitReached*

This bit is set if a limit (position, speed) is exceeded.

Bit 7: *PdiStatusQsHaltBit*

This bit is set if the drive is in the *Quick stop active* state or if the *halt bit* was set.

Bit 11: *PdiStatusHomingDone*

This bit is set if a *homing operation* was successfully performed; the bit remains set until you switch off the controller.

Bit 12: *PdiStatusAutosetupDone*

This bit is set if an *auto setup* was successfully performed; the bit remains set until you switch off the controller.

Bit 14: *PdiStatusToggleCmd*

This bit is toggled if a new command was detected. In combination with the *PdiStatusError* bit, you can determine whether the new command was accepted.

Bit 15: *PdiStatusError*

This bit is set if errors occurred when executing a command. You can read out the error codes in *PDI-ReturnValue* ($2292_h:02_h$).

3 Implementation on the fieldbus

The *Plug & Drive interface* functions on all fieldbuses supported by the Nanotec controllers. For further information on the special features of the respective fieldbus, refer to the corresponding chapter of the the technical manual for the controller.

You need at least two PDOs (process data objects).

Map the following RX-PDO on the receiving side:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
$2291_h:01_h$ (*PDI-SetValue1*)				$2291_h:02_h$ (*PDI-SetValue2*)		$2291_h:03_h$ (*PDI-SetValue3*)	$2291_h:04_h$ (*PDI-Cmd*)

Map the following TX-PDO on the transmitting side:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
$2292_h:01_h$ (*PDI-Status*)		$603F_h:00_h$ (*Error Code*)		$2292_h:02_h$ (*PDI-ReturnValue*)			

If you need additional process values of the drive, you can configure further PDOs. Nanotec recommends not mapping objects 6040_h (controlword) and 6060_h (Modes Of Operation), since the commands of the *Plug & Drive interface* overwrite these objects. The same applies for other objects that are overwritten by the used *PDI-Cmd* commands, such as target position ($607A_h$), speeds (6081_h, $60FF_h$), etc.

3.1 CANopen

With CANopen, Nanotec recommends configuring the PDO with acyclic processing with a realistic *inhibit time*.

If, in the network, you use a CAN or CANopen master that does not support any NMT service, proceed as follows after configuring the PDO: set object $1F80_h:00_h$ for the CANopen slave to 08_h. The slave thereby changes to the *Operational* NMT state immediately after starting, allowing the PDOs to be used.

3.2 Modbus

With Modbus, Nanotec recommends working with function codes 23 (Read/Write Multiple Registers) or 4 (Read Input Register) and 22 (Write Multiple Registers). Only with these function codes can you transfer the mapped data of the *Plug & Drive interface* in a workstep.

The PDI objects are already mapped in the factory settings. To access the PDI objects, use the Modbus register with the following addresses:

On the receiving side (master → slave):

Register address	Object
5996	$2291_h:01_h$ (*PDI-SetValue1*)
5997	
5998	$2291_h:02_h$ (*PDI-SetValue2*)
5999	• Byte 0: $2291_h:03_h$ (*PDI-SetValue3*) • Byte 1: $2291_h:04_h$ (*PDI-Cmd*)

On the transmitting side (slave → master):

Register address	Object
4996	2292_h:01_h (*PDI-Status*)
4997	$603F_h$:00_h (*Error Code*)
4998 und 4999	2292_h:02_h (*PDI-ReturnValue*)

These address ranges are directly adjacent to the ranges of the normal PDOs (from address 6000 for RX-PDO and 5000 for TX-PDO). You can use a Modbus command to simultaneously operate the *PDI interface* and access other mapped objects (e.g., with function code Read/Write Multiple Registers).