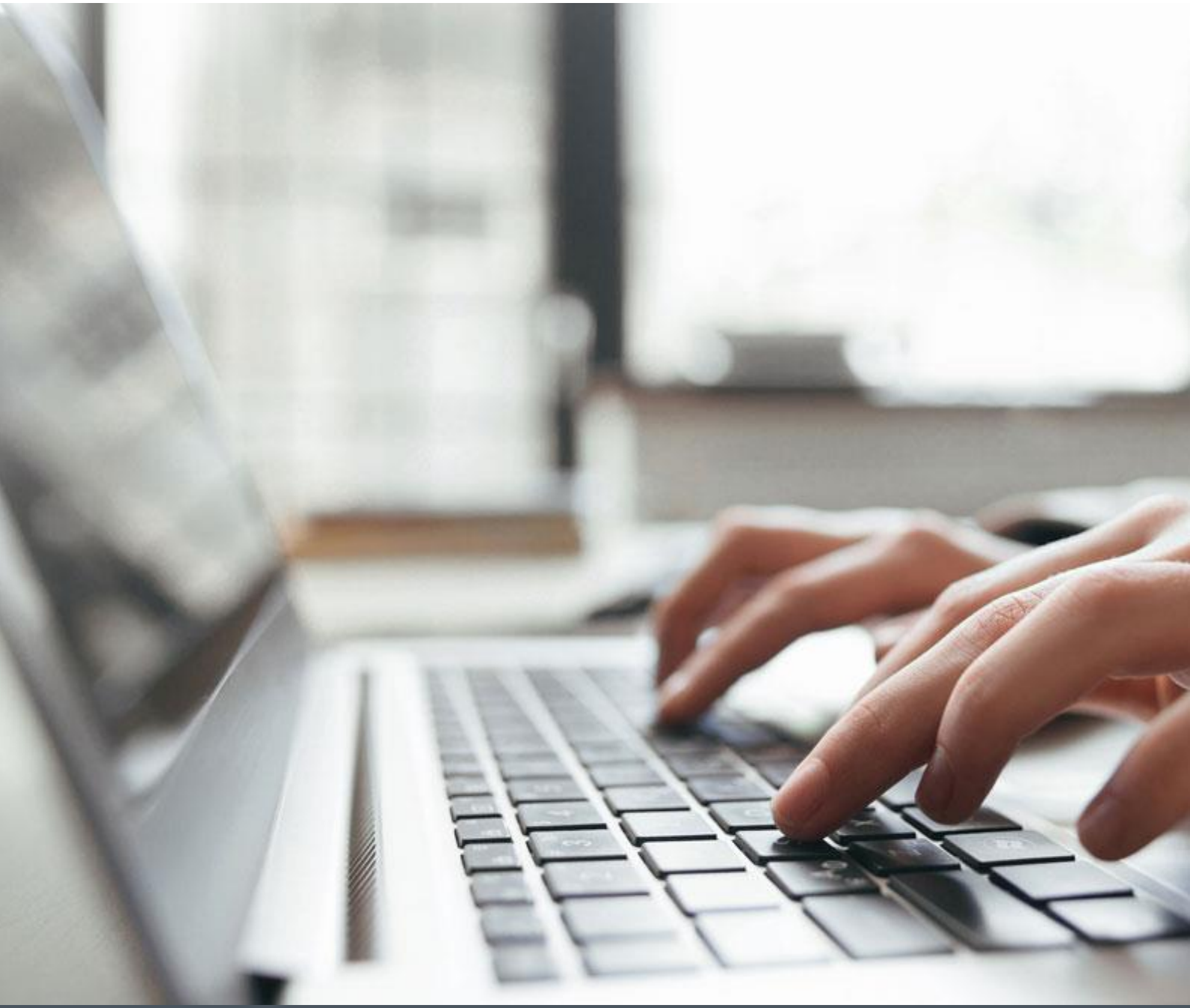

Application Note

Set up Nanotec drives in a CAN network without PLC

Version 1.0.0

www.nanotec.com

Contents

1 Intended use and audience

Based on an example combination, this application note shows you how to establish communication between Nanotec devices via CANopen using NanoJ-programming. This way you can realize simple Master-Slave-functionalities without the use of a PLC.

2 Requirements

NOTICE

Malfunction from incompatibility! The products required here come in various versions. Duly select / install / configure the correct products in advance.

- ► Fulfil the requirements.
- ► Use compatible equipment only.

1. You must ensure proper connection of the motor and encoder to the drive.
2. Keep controller / drive operation unhindered by any stand-alone program (NanoJ) running on the slave etc.

Hardware	• 2x PD4-C5918L424-E-08, Firmware Version FIR-v2139-B1022383 Converter USB-to-CAN V2 compact (IXXAT)
Software	• Plug&Drive-Studio v2.1.3

3 Pre-settings and connection

Before starting to program, make sure the connection and some pre-settings are done.

1. Set the Baudrate for all controllers to the same speed (default 1Mbaut).
2. Set different Node-IDs for all controllers. The controller that should act as a master must have the Node-ID 1, for the other devices need Node-IDs in the range between 2 and 63.
3. Connect all used devices in a CANopen daisy chain (build a connection chain by connecting “Can Out” of the first device to “Can In” of the second device, “Can Out” of the second device to “Can In” of the third device etc.)

4 NanoJ Code for Master and Slave Devices

The NanoJ Code examples provided enable reading and writing objects on the Slave Devices via commands in the NanoJ Code of the Master-Device.

This allows stand-alone applications with up to 64 Devices controlled from one source (the NanoJ Code for the Master Device).

To make this possible the Code uses Dummy Objects (NanoJ Inputs 0x2400:01/02 and NanoJ Outputs 0x2500:01/02) to generate a request-response communication protocol processed via CANopen PDOs.

The necessary PDO Mappings are managed in the function “Config_PDOs()”.

Messages on the Bus consist of two S32 Objects.

The first Object includes the Header, the second includes the Value.

The Header is built as follows:

31	30	29…24	23…8	7…0
Toggle-Bit	Write-Bit	Address	Index	Sub-Index

Toggle-Bit: Every Slave has an own Toggle-Bit to identify new messages

Write-Bit: Identifies if the message is a Read or Write Request

Address: The Node-ID of the addressed device

Index: Index of the Object to write/read

Sub-Index: Sub-Index of the Object to write/read

To send a message to a Slave Device the following commands are used in the NanoJ Code of the Master Device:

Send_Write_Request(U16 Address, U16 Index, U08 SubIndex, S32 Value);

Send_Read_Request(U16 Address, U16 Index, U08 SubIndex);

5 Usage of the NanoJ Code examples

To edit and/or compile and transfer the NanoJ program to your Nanotec drive, open the provided CPP-File in the Plug&Drive-Studio software. Please refer to the Plug&Drive-Studio documentation for details.

The NanoJ Code “PDO-Slave” has to be uploaded to all Slave-Devices. No changes of the Code are necessary. The code automatically gets the Device Address by reading the Node-ID of the device.

The NanoJ Code “PDO-Master” includes all necessary preparations to establish communication to the Slave Devices. You can add your own code starting at line 117.

When done with programming, the code has to the uploaded to the Master-Device.

6 Example motion program

The example code at lines 119 to 129 of “PDO-Master” would set up the Slave Device with Node-ID 2 to follow every movement of the Master Device by using the Cyclic Synchronous Position Mode.

The Master Device sends its Actual Position (0x6064) as a Target Position (0x607A) to the Slave Device in an endless loop:

```
117     //from this point on you can write your own code
118
119     Send_Write_Request(2,0x6060, 0x00, 8);
120     Send_Write_Request(2,0x60C2, 0x01, 10);
121     Send_Write_Request(2,0x6040, 0x00, 6);
122     Send_Write_Request(2,0x6040, 0x00, 7);
123     Send_Write_Request(2,0x6040, 0x00, 15);
124
125     while(true)
126     {
127         Send_Write_Request(2,0x607A, 0x00, od_read(0x6064, 0x00));
128         yield();
129     }
130 }
```

7 Liability

This Application Note is based on our experience with typical user requirements in a wide range of industrial applications. The information in this Application Note is provided without guarantee regarding correctness and completeness and is subject to change by Nanotec without notice.

It serves as general guidance and should not be construed as a commitment of Nanotec to guarantee its applicability to all customer applications without additional tests under the specific conditions and – if and when necessary – modifications by the customer.

The provided information does not replace datasheets and other product documents. For the latest version of our datasheets and documentations please visit our website at www.nanotec.com.

The responsibility for the applicability and use of the Application Note in a particular customer application lies solely within the authority of the customer. It is the customer's responsibility to evaluate, investigate and decide, whether the Application Note is valid and suitable for the respective customer application, or not.

Defects resulting from the improper handling of devices and modules are excluded from the warranty. Under no circumstances will Nanotec be liable for any direct, indirect, incidental or consequential damages arising in connection with the information provided.

In addition, the regulations regarding the liability from our Terms and Conditions of Sale and Delivery shall apply.

8 Imprint

Nanotec Electronic GmbH & Co. KG │ Kapellenstraße 6 │ 85622 Feldkirchen │ Germany
Tel. +49 (0)89 900 686-0 │ Fax +49 (0)89 900 686-50 │ info@nanotec.de │ www.nanotec.com