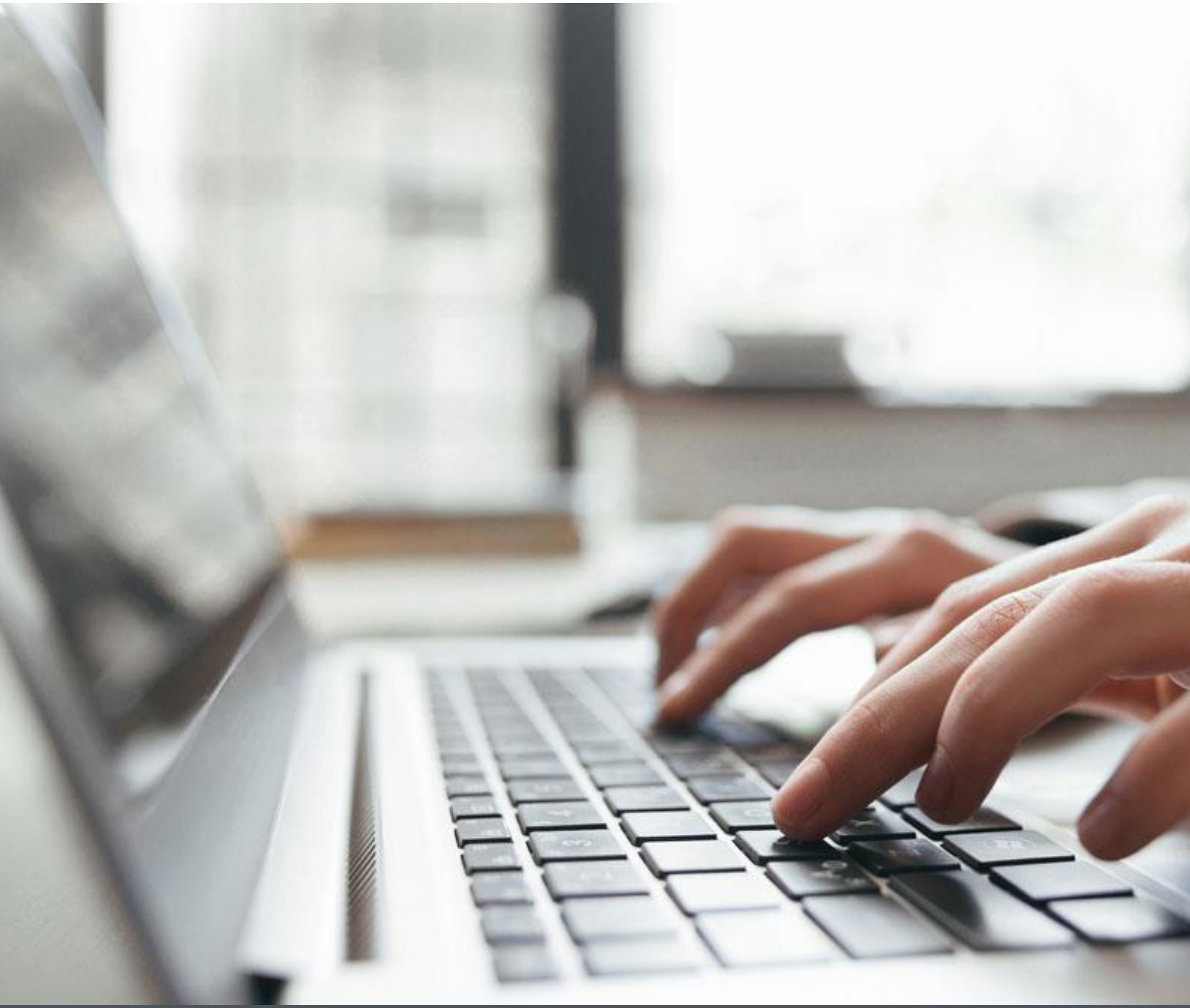


Application Note

How to use *Digital Output* in *NanoJ*

Version 1.0.0

www.nanotec.com

Contents

1 Intended use and audience

This application note shows you how to use the digital outputs of a Nanotec motor controller in a NanoJ program. You can find the corresponding NanoJ code template in the download folder.

Digital Output offers a NanoJ code template for assigning certain functions to electronic Nanotec motor controller ouput signals. To open and edit the template requires Plug & Drive Studio software. Both NanoJ and Plug & Drive Studio are for use with Nanotec products only, by trained specialists only.

2 Prerequisites

NOTICE
Malfunction from incompatibility! Plug & Drive Studio comes in various software versions. Find out and, if necessary, install the correct version for your Nanotec motor controller in advance.

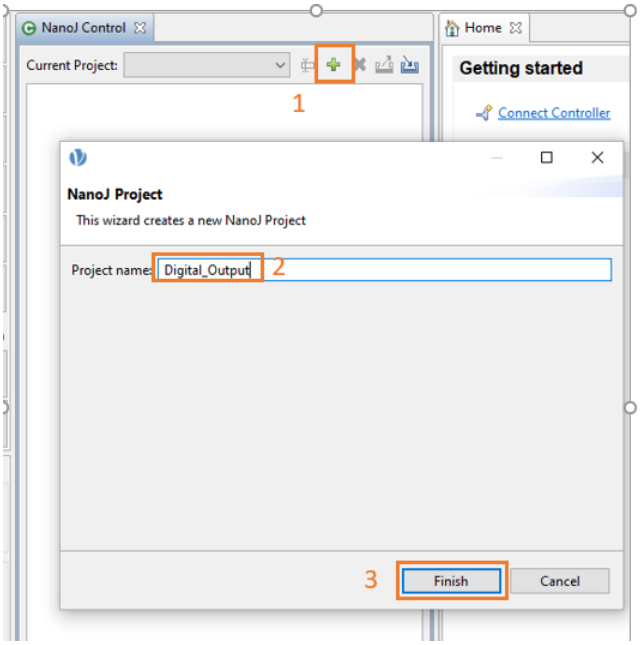
You must have the correct Plug & Drive Studio version installed on your computer:

1. Open the Nanotec software webpage.
2. Click on the *Plug & Drive Studio* buttons.
3. Browse *Compatible Products* to find out which version is compatible with your motor controller.
4. Download and install the latest compatible Plug & Drive Studio version on your computer.
5. If not done so yet: Also download the latest NanoJ V2 Library (`nanotec.h`).

3 Creating a new project in Plug & Drive Studio

Open the *NanoJ Control* tab and click on the "**+**" icon (1). A *NanoJ Project* tab pops up:

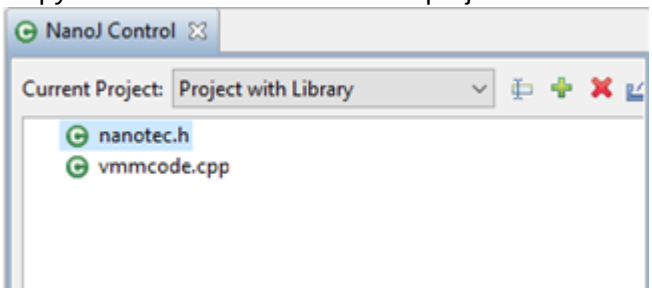
1. Assign a new project name (2).
2. Click on *Finish* (3) to close the tab.
3. Your new project is now created.



4 Including the nanotec.h library into your NanoJ project

The Plug & Drive Studio installation folder does include `wrapper.h`. But you must download the NanoJ V2 library (`nanotec.h`) from our knowledge base and copy it into NanoJ:

1. Generate a new NanoJ project or open an existing one.
2. Copy the `nanotec.h` file into the project tree via drag & drop:



3. To implement the NanoJ V2 library, add `#include wrapper.h` and `#include nanotec.h` to your code:

```
10
11 #include "wrapper.h"
12 #include "nanotec.h"
13
14
15 void user()
16 {
```

5 Using the code template for digital outputs in NanoJ

5.1 Including libraries, mappings

For our case, we use the Nanotec NanoJ V2 library `nanotec.h` to implement the code template and provide basic functions to control our motor. To include the `nanotec.h` library, we must at least add the object mappings in lines 26 to 34 to our code:

```
26 map U16 Controlword as inout 0x6040:00
27 map U16 Statusword as input 0x6041:00
28 map U32 Inputs as input 0x60FD:00
29 map U32 Outputs as inout 0x60FE:01
30 map S08 ModesOfOperation as output 0x6060:00
31 map S08 ModesOfOperationDisplay as input 0x6061:00
32 map S16 AnalogInput as input 0x3220:01
33 map S32 TargetPosition as output 0x607A:00
34 map U32 ProfileVelocity as output 0x6081:00
```

5.2 Main program loop: void user()

5.2.1 Selecting a profile velocity, defining local variables

With `ProfileVelocity` mapped to `output 0x6081` (see line 34 above), we now set it to 200 rpm (which you can change anytime):

```
41 void user()
42 {
43     Out.ProfileVelocity=200;            //set the profile velocity to 200rpm
```

5.2.2 Implementing a release function (Input 1)

For Input 1 signals, we implement a release function. A high release signal **powers** the motor, a low signal **unpowers** it. The release function thus ensures the motor to run on a high release signal only.

- Line 54: With a release signal not set to high, you can stop the motor via `Quickstop()` function.
- Line 46: Input 1 also selects *Profile Position* via `ModesOperation(1)`.
- Line 48: `AbsoluteMovement()` defines the absolute movement operation mode.
- Line 50: With `ChangeSetPointImmediately()` set to `true`, you can activate *change setpoint immediately* to execute each new travel command immediately:

```
42     while(true)
43     {
44         if(DigitalInput(1))                          //if Input 1 is high.
45         {
46             ModesOfOperation(1);                     //set the Mode to Profile Position
47             EnableOperation();                       //change the state to Operation Enabled
48             AbsoluteMovement();                      //set to absolute movement
49             //RelativeMovement();                    //set to relative movement
50             ChangeSetPointImmediately(true);         //change setpoint immediately
51         }
52         else                                         //else...
53         {
54             Quickstop();                             //stop the motor
55         }
```

5.2.3 Changing the target position (Input 2 & 3)

By default, the code template selects a target position of *1000* if Input 2 is high (and *3000* if low). The position starts with high Input 3 `if(DigitalInput(3)`. A new setpoint is set only with Input 3.

- Line 78: With *change setpoint immediately* active in line 50, `NewSetPoint()` runs each new travel command immediately, interrupting the current one before reaching its target position. Use this, for example, to set a new target position with a different profile velocity during movement:

```
62          if(DigitalInput(2))                     //if Input 2 is active
63          {
64              Out.TargetPosition=1000;            //set target position to 1000
65          }
66          else                                    //else
67          {
68              Out.TargetPosition=3000;            //set target position to 3000
69          }
70
71
72          if(DigitalInput(3))                     //if Input 3 is active
73          {
74              NewSetPoint(true);                  //set new setpoint
75          }
76          else                                    //else
77          {
78              NewSetPoint(false);                 //reset new setpoint
79          }
```

5.2.4 Setting the digital outputs

Implemented as open drain, the outputs require external voltage supply. Some controls have two, others have three outputs. Our example uses three.

- Line 82: Reaching the target position will set Output 1 to high (otherwise, it remains low).
- Line 92: An *Operation Enabled* controller state will set Output 2 to high.
- Line 98: Otherwise, Output 2 is set to low. Output 3 is set to high if an error occurs.

```
82          if(TargetReached())                     //if the target position has been reached
83          {
84              SetDigitalOutput(1);                //set output 1
85          }
86          else                                    //else
87          {
88              ClearDigitalOutput(1);              //clear output 1
89          }
90
91
92          if ((In.Statusword & 0xEF) == 0x27)     //if the "Operation Enabled"-State is reached
93          {
94              SetDigitalOutput(2);                //set output 1
95          }
96          else                                    //else
97          {
98              ClearDigitalOutput(2);              //clear output 1
99          }
100
101
102         if ((In.Statusword & 0x8) == 0x8)       //if an error occurs
103         {
104             SetDigitalOutput(3);                //set output 3
105         }
106         else                                    //else
107         {
108             ClearDigitalOutput(3);              //clear output 3
109         }
110
111         yield();
112     }
113 }
```

Your code is finally implemented.

6 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.

7 Imprint

© 2021 Nanotec Electronic GmbH & Co. KG, all rights reserved. Original version.

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