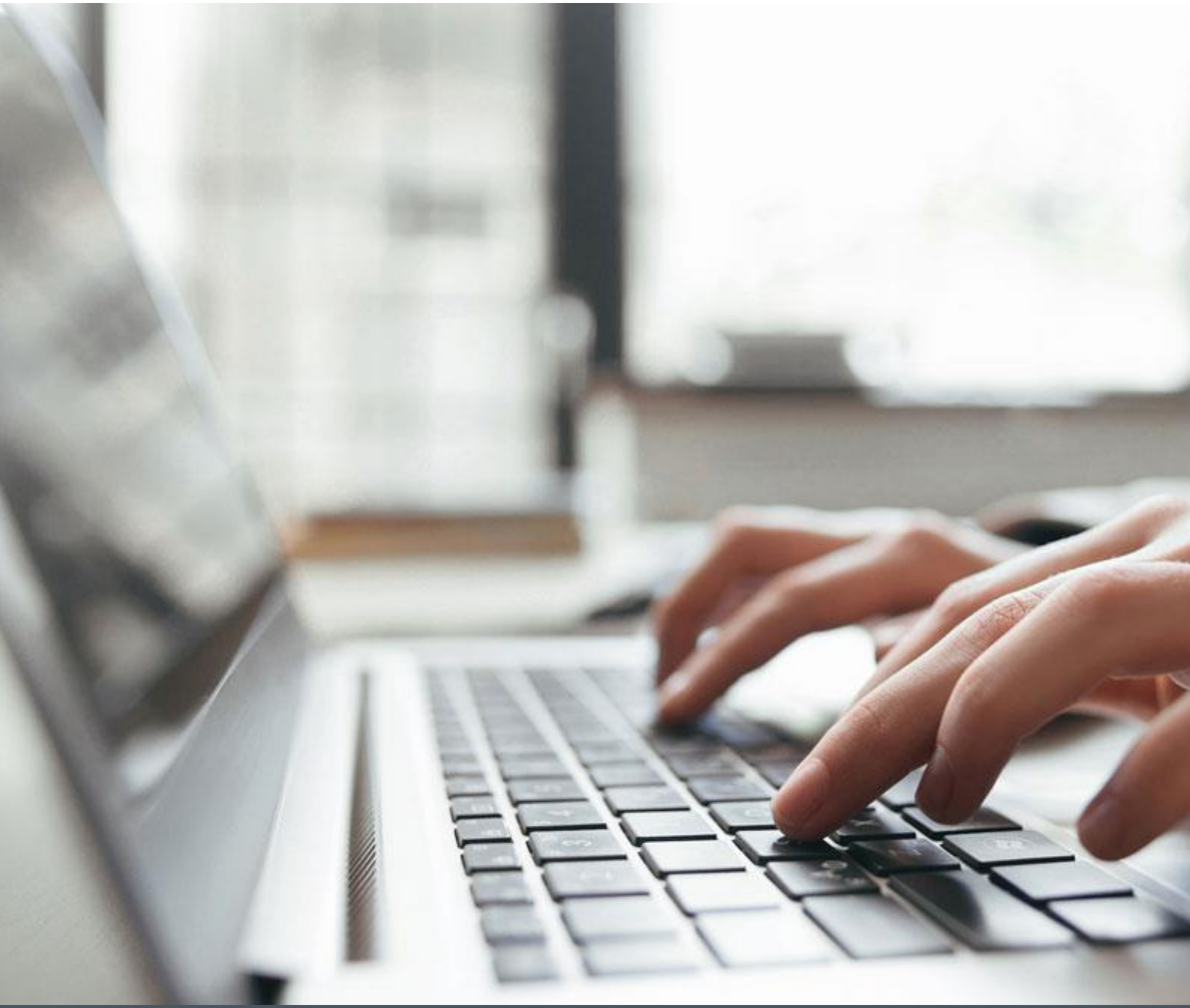


Application Note

How to use *Analog Input* in *NanoJ*

Version 1.0.0

www.nanotec.com

Contents

1 Intended use and audience 1

2 Prerequisites 1

3 Creating a new project in Plug & Drive Studio 2

4 Including the nanotec.h library into your NanoJ project 2

5 Using the code template for analog input in NanoJ 3

5.1 Including libraries, mappings 3

5.2 Main program loop: void user() 3

5.2.1 Selecting a profile velocity, defining local variables 3

5.2.2 Implementing a release function (Input 1) 4

5.2.3 Starting the motor via analog input 4

6 Liability 4

7 Imprint 5

1 Intended use and audience

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

Analog Input offers a NanoJ code template for setting the target velocity via analog input of an electronic Nanotec motor controller. 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.

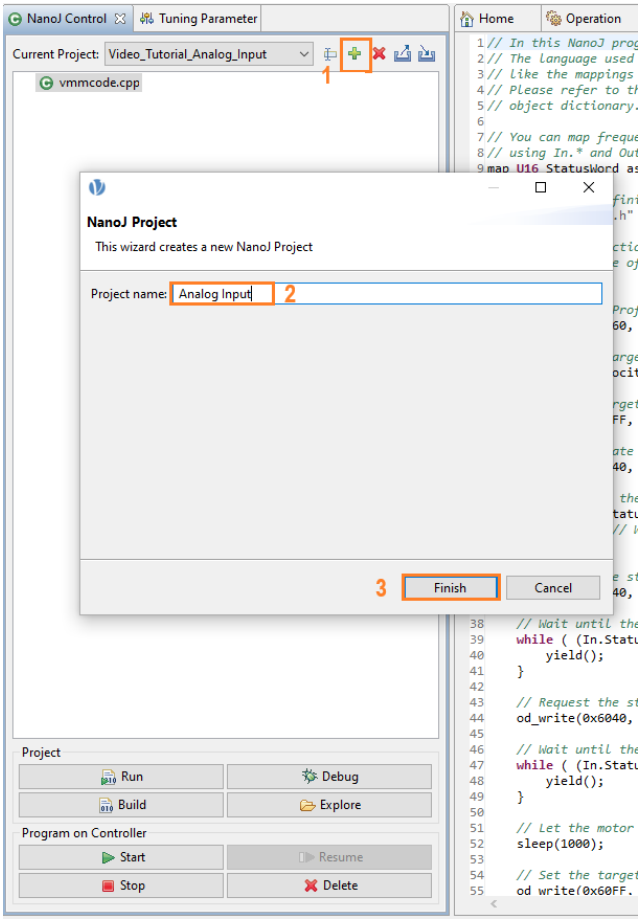
You must have the correct Plug & Drive Studio version installed on your computer:

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:

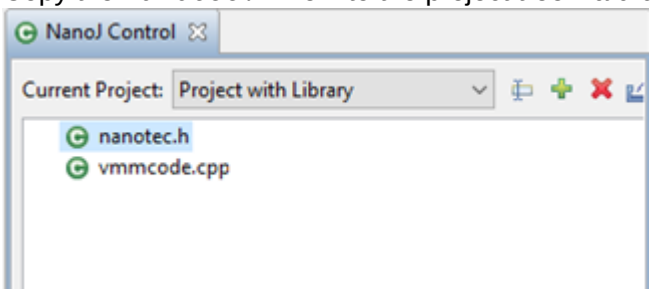
1. Assign a new project name (2).
2. Click on *Finish* (3) to close the tab.
3. Your new project is now created.



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 analog input in NanoJ

5.1 Including libraries, mappings

For our case, we use the Nanotec NanoJ V2 library `nanotec.h` in our code template to provide basic functions to control our motor. To include the `nanotec.h` library, we must at least add the object mappings in lines 29 to 36 to our code. We also include the libraries `wrapper.h` and `nanotec.h`:

```
28 // You can map frequently used objects to be able to read or write them
29 map U16 Controlword as inout 0x6040:00
30 map U16 Statusword as input 0x6041:00
31 map U32 Inputs as input 0x60FD:00
32 map U32 Outputs as inout 0x60FE:01
33 map S08 ModesOfOperation as output 0x6060:00
34 map S08 ModesOfOperationDisplay as input 0x6061:00
35 map S16 AnalogInput as input 0x3220:01
36 map S32 TargetVelocity as inout 0x60FF:00
37
38 // Include the definition of NanoJ functions and symbols
39
40 #include "wrapper.h"
41 #include "nanotec.h"
```

5.2 Main program loop: void user()

5.2.1 Selecting a profile velocity, defining local variables

We can switch some analog controller input from voltage to current measurement. Some Nanotec controls have an analog input of -10 V to +10 V (= 1024 digits). Other controls use 1024 digits for +10 V-only analog input. **Note:** Even at 0 V, some potentiometers still deliver minimal signals that the control reacts to. As a result, the motor begins to vibrate.

- Line 42: For operation mode, we select *ProfileVelocity*.
- Line 45, 46: Via objects 0x6083 and 0x6084, we define profile acceleration and deceleration.
- Line 49, 50: We define a maximum for acceleration (0x60C5) and deceleration (0x60C6).
- Line 53: For our case, we set a maximal velocity of 2000 rpm to be reached at analog input maximum (= 10 V if measured in voltage).
- Line 54: To avoid Motor vibration, we first create a filter with an offset.

```
42    // Using operation mode "Profile Velocity":
43    Out.ModesOfOperation = 3;
44    // Acceleration and Deceleration
45    od_write(0x6083, 0x00, 1000);
46    od_write(0x6084, 0x00, 1000);
47    yield();
48    // Max Accleration and Deceleration
49    od_write(0x60C5, 0x00, 5000);
50    od_write(0x60C6, 0x00, 5000);
51    yield();
52    // Max Motor Speed and Offset
53    S32 MaxVelocity = 2000;
54    S32 Offset = 50;
```

5.2.2 Implementing a release function (Input 1)

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

```
51      while(true)
52      {
53          // Release signal on Input 1
54          while(DigitalInput(1))
55          {
```

With a release signal set to low, we shut down the power state machine via `Shutdown()` function:

```
97          }
98
99          // If Release signal is low
100         Shutdown();
101         yield();
102     }
```

5.2.3 Starting the motor via analog input

After activating the enable signal via Input 1, we can control the motor via potentiometer. **Note:** On analog values lower than offset, the motor won't move.

- Line 71: Only on analog values higher than offset, the motor rotates at calculated velocity.

```
63      // Defining an offset for the analog input. When the analog input is below this offset the motor should stop (Velocity = 0)
64      if(In.AnalogInput < Offset )
65      {
66          Out.TargetVelocity = 0;
67      }
68      // If the analog input is above the offset-value, it should scale the velocity up to the MaxVelocity in a linear connection
69      else
70      {
71          Out.TargetVelocity = (MaxVelocity * (AnalogInput()-Offset)) / (1023-Offset);
72      }
73
74      EnableOperation();
75  }
```

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

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