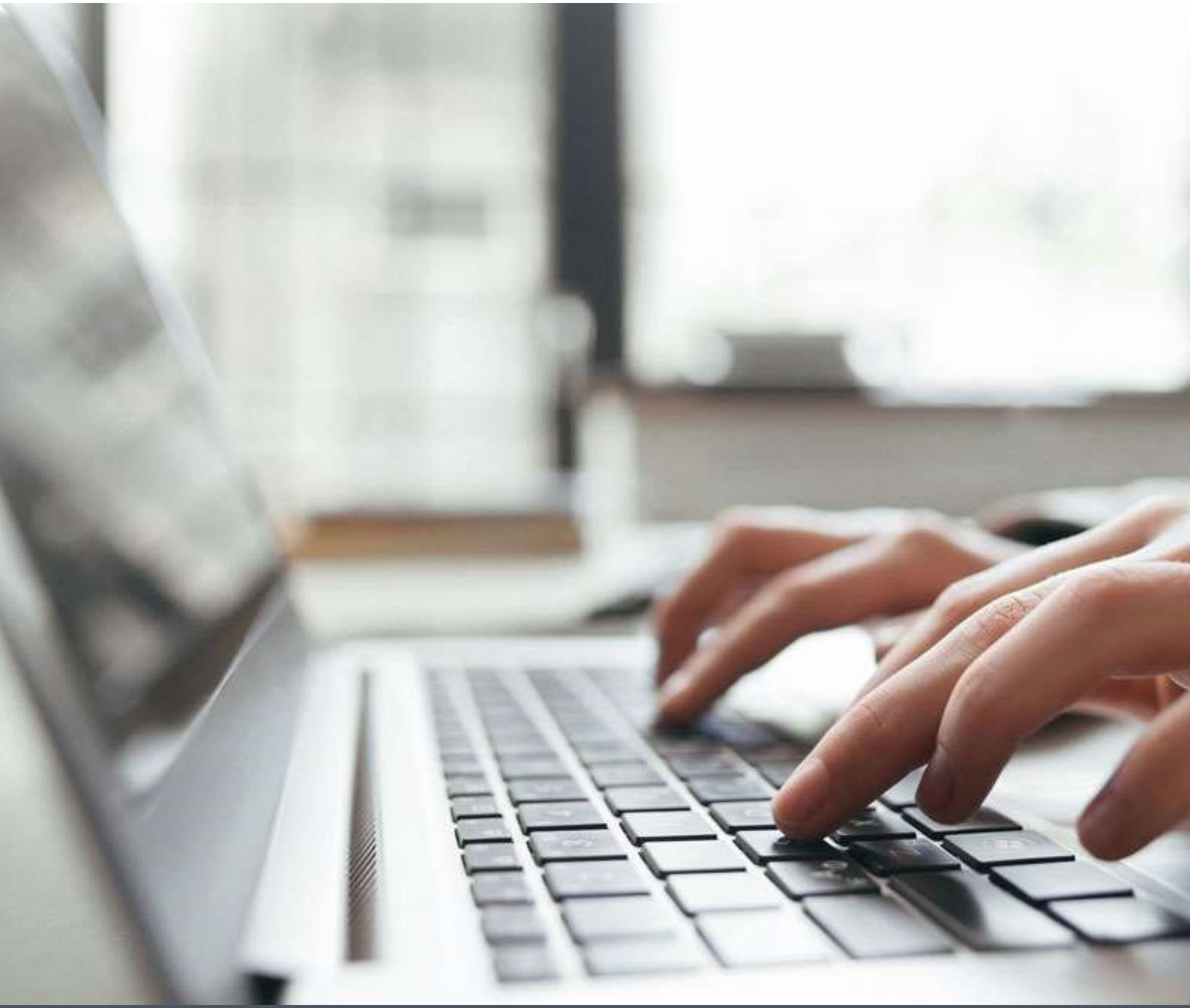
Nanotec®
PLUG & DRIVE



Application Note

How to use *Reversing Operation* in *NanoJ*

Version 1.0.1

www.nanotec.com

Contents

1 Intended use and audience

This application note shows you how to implement a reversing operation in a NanoJ program. Please find the respective NanoJ code template in the download folder.

Reversing Operation offers a NanoJ template you can use as a test program for motor tuning and to adjust the PID parameters during the reversing operation. The implementation uses the mode profile velocity. It offers all motion parameters, such as acceleration / deceleration ramps etc., as a clear set of constant variables you can easily parametrize.

Template opening / editing requires Plug & Drive Studio software which, like NanoJ itself, is for use with Nanotec products only, by trained experts only.

2 Prerequisites

NOTICE
Malfunction from incompatibility! Plug & Drive Studio comes in various software versions. Install the correct one for your Nanotec motor controller in advance.

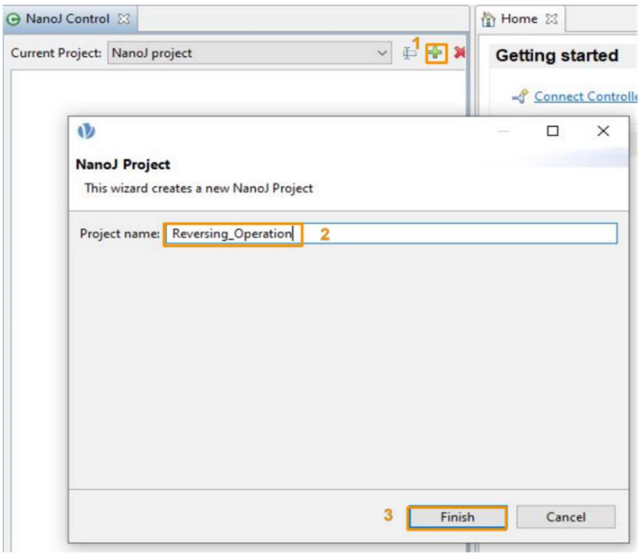
You must have the correct Plug & Drive Studio version installed on your computer:

1. Open the Nanotec software webpage.
2. Click on the *Plug & Drive Studio* buttons.
3. Browse *Compatible Products* for the version compatible with your motor controller.
4. Download and install the latest compatible Plug & Drive Studio 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 the **+ icon** (1). A *NanoJ Project* tab pops up:

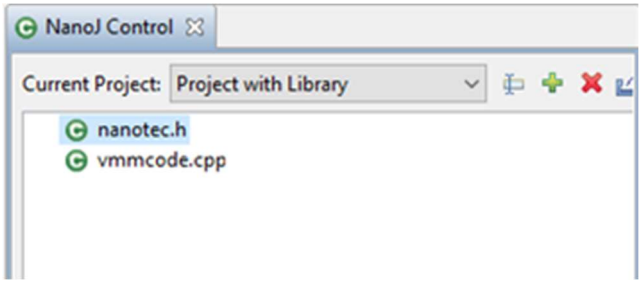
1. Assign a new project name (2).
2. Click on *Finish* (3) to close the tab.
3. Your new project is now created.



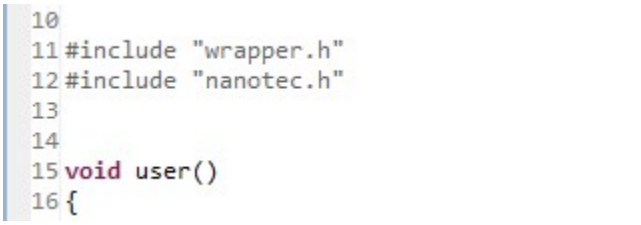
4 Including the nanotec.h library into your NanoJ project

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. Create 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.



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` to provide basic motor-control functions.

To make the `nanotec.h` library usable, we must at least add the object mappings in lines 24 to 30 to our code.

```
24 map U16 Controlword as inout 0x6040:00
25 map U16 Statusword as input 0x6041:00
26 map U32 Inputs as input 0x60FD:00
27 map U32 Outputs as inout 0x60FE:01
28 map S08 ModesOfOperation as output 0x6060:00
29 map S08 ModesOfOperationDisplay as input 0x6061:00
30 map S16 AnalogInput as input 0x3220:01
```

In lines 33 to 38, we must insert the mappings for profile velocity parameters.

```
32 // MAPPINGS FOR PROFILE VELOCITY
33 map S32 TargetVelocity as inout 0x60FF:00
34 map U32 ProfileAcceleration as inout 0x6083:00
35 map U32 ProfileDeceleration as inout 0x6084:00
36 map U32 QuickStopDeceleration as inout 0x6085:00
37 map U32 MaxAcceleration as inout 0x60C5:00
38 map U32 MaxDeceleration as inout 0x60C6:00
```

Only then, we include the libraries `wrapper.h` and `nanotec.h`.

```
40 // Include the definition of NanoJ functions and symbols
41 #include "wrapper.h"
42 // Include the NanoJ V2 Library
43 #include "nanotec.h"
```

5.2 Implementing a user interface with constant variables

We want to implement a user interface with a clear set of constant variables to define / parametrize the profile velocity for operation, but also the motion time for positive / negative direction.

Our user interface is also to adjust the acceleration / deceleration ramps in both directions. Finally, our constant variables must limit the maximum deceleration / acceleration.

```
////////// USER INTERFACE //////////

// TARGET VELOCITIES FOR LEFT AND RIGHT DIRECTION:
#define TARGET_VELOCITY_POSITIVE_DIRECTION                    300
#define TARGET_VELOCITY_NEGATIVE_DIRECTION                   -300

// TIME PARAMETERS FOR RIGHT AND LEFT DRIVE
#define TIME_POSITIVE_DIRECTION                               1000
#define TIME_NEGATIVE_DIRECTION                               1000

// ACCELERATION PARAMETERS
#define PROFILE_ACCELERATION_POSITIVE_DIRECTION               50000
#define PROFILE_ACCELERATION_NEGATIVE_DIRECTION               50000
#define PROFILE_DECELERATION_POSITIVE_DIRECTION               50000
#define PROFILE_DECELERATION_NEGATIVE_DIRECTION               50000

// ACCELERATION LIMITS
#define MAX_DECELERATION                                      100000
#define MAX_ACCELERATION                                      100000

////////// END OF USER INTERFACE //////////
```

5.3 Main program loop: void user()

5.3.1 Parametrizing motor acceleration, speed, deceleration, reversing

- Line 72 to 73: First we limit maximum acceleration / deceleration.
- Line 76 to 77: For profile velocity, we select `ModesOfOperation(3)` for object 0x6060 (cf. line 28). We switch the state machine to `EnableOperation()` to make the motor run instantly at profile speed.

```
67 // The user() function is the entry point of the NanoJ program. It is called
68 // by the firmware of the controller when the NanoJ program is started.
69 void user()
70 {
71     // Sets Acceleration Limits:
72     InOut.MaxAcceleration = MAX_ACCELERATION;
73     InOut.MaxDeceleration = MAX_DECELERATION;
74
75     // Sets mode "Profile velocity" and starts Operation
76     ModesOfOperation(3);
77     EnableOperation();
```

- Line 79: We embed the actual reverse operation in a `while(true)` loop, to avoid that our `void user()` function ends in full rerun / remapping.
- Line 82: In `while(true)`, we first set the target velocity for positive direction.

- Line 83 to 84: Only then, we set the profile acceleration / deceleration for positive direction.
- Line 85: Our `sleep` time defines how long the motor is to run in positive direction.

```
79      while(true)
80      {
81          // Sets Velocity, Ramps and Time for the positive Direction
82          InOut.TargetVelocity = TARGET_VELOCITY_POSITIVE_DIRECTION;
83          InOut.ProfileAcceleration = PROFILE_ACCELERATION_POSITIVE_DIRECTION;
84          InOut.ProfileDeceleration = PROFILE_DECELERATION_POSITIVE_DIRECTION;
85          sleep(TIME_POSITIVE_DIRECTION);
```

- Line 87: Still in `while(true)`, we then implement the negative motion parameters.
- Line 88 to 90: We set both target velocity and the profile acceleration / deceleration.
- Line 91: Our `sleep` time finally defines how long the motor is to run in negative direction.

```
87          // Sets Velocity, Ramps and Time for the negative Direction
88          InOut.TargetVelocity = TARGET_VELOCITY_NEGATIVE_DIRECTION;
89          InOut.ProfileAcceleration = PROFILE_ACCELERATION_NEGATIVE_DIRECTION;
90          InOut.ProfileDeceleration = PROFILE_DECELERATION_NEGATIVE_DIRECTION;
91          sleep(TIME_NEGATIVE_DIRECTION);
92      }
93  }
```

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