

# Application Note

How to use *Homing* in *NanoJ*

Version 1.0.0

## Contents

<b>1</b>	<b>Intended use and audience .....</b>	<b>1</b>
<b>2</b>	<b>Prerequisites .....</b>	<b>1</b>
<b>3</b>	<b>Creating a new project in Plug &amp; Drive Studio .....</b>	<b>2</b>
<b>4</b>	<b>Including the nanotec.h library into your NanoJ project.....</b>	<b>2</b>
<b>5</b>	<b>Using the code template for analog input in NanoJ.....</b>	<b>2</b>
5.1	Including libraries, mappings.....	2
5.2	Main program loop: void user() .....	3
5.2.1	Selecting the homing operation mode / method.....	3
<b>6</b>	<b>Liability .....</b>	<b>4</b>
<b>7</b>	<b>Imprint.....</b>	<b>4</b>

## 1 Intended use and audience

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

*Homing* offers a template for selecting and parametrizing the homing function for Nanotec motor controllers. You can use this template to perform a homing on the encoder index, a reference/limit switch, the actual position or on block.

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.

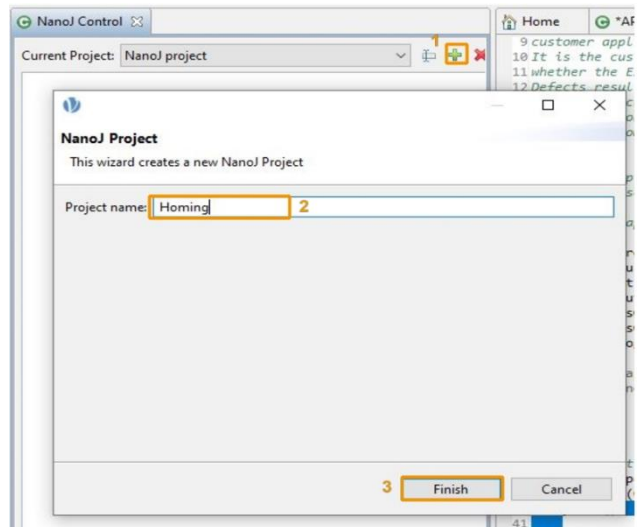
You must have the correct Plug & Drive Studio version installed on your computer:

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:

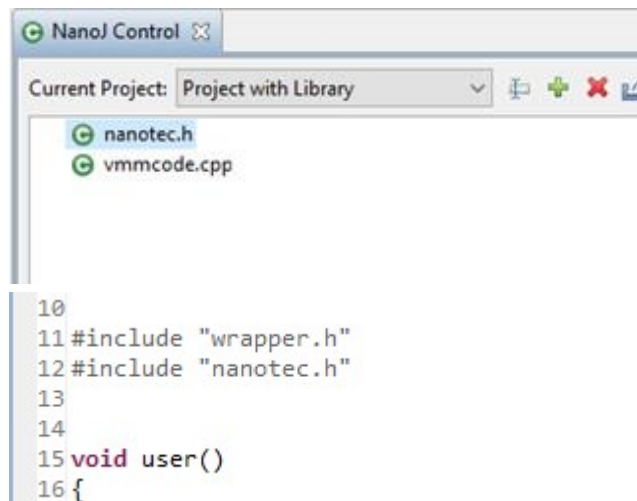
1. Assign a new project name (2).
2. Click on *Finish* (3) to close the tab.
3. Your new project is now created.



### 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 code template's 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 23 to 29 to our code.

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

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

```
31#include "wrapper.h"
32#include "nanotec.h"
```

## 5.2 Main program loop: void user()

### 5.2.1 Selecting the homing operation mode / method

- Line 38: For operation mode, we select `ModesOfOperation(6)` for object `0x6060` (cf. line 27).
- Line 39: By writing `35` into object `0x6098`, we select homing method `35` for this example. Please refer to the product manual to find the suitable homing method for your application.
- Line 42 to 45: We ensure the homing speed for objects `0x6099:01` and `0x6099:02`, and the homing acceleration for object `0x609A:00`.

```
35void user()
36{
37    //Activation of the Homing operation mode:
38    ModesOfOperation(6);
39    od_write(0x6098, 0x00, 35);    //Homing Method, please refer to the manual to select the method, fitting your applicaiotn
40    yield();
41
42    od_write(0x6099, 0x01, 50);    //Homing Speed: Speed During Search For Switch
43    od_write(0x6099, 0x02, 10);    //Homing Speed: Speed During Search For Zero
44    od_write(0x609A, 0x00, 500);    //Homing Acceleration
45    yield();
```

- Line 47 to 49: For homing on block, we configure objects `0x203A:01` (= blocking current level) and `0x203A:02` (= blocking time).
- Line 51 to 53: For homing on a switch, we assign value `7` to object `0x3240:01` (= digital inputs control, special function enabled). We enable the switches for negative limit (= bit 0), positive limit (= bit 1), and home (= bit 2). Home offset defines the actual position after homing success.

```
47    od_write(0x203A, 0x01, 0xFFFFFBA); //Homing On Block Configuration: Minimum Curren for Block Detection (only for Homing on Block)
48    od_write(0x203A, 0x02, 200);    //Homing On Block Configuration: Block Detection Time (only for Homing on Block)
49    yield();
50
51    od_write(0x3240, 0x01, 7);    //Digital Inputs Control: Special Function Enable (only for Homing on Switches)
52    od_write(0x607C, 0x00, 0);    //Home Offset
53    yield();
```

- Line 55 to 56: With the homing parameter fully configured, we first switch the power state machine to `EnableOperation()`. Only then, we use `NewSetPoint(true)` to toggle the control word bit 4 and start homing.
- Line 58 to 61: The motor is to run if bit 10 (= target reached) and bit 12 (= new setpoint acknowledged) are not set, yet.

```
55    EnableOperation();    //switch to Operation Enabled
56    NewSetPoint(true);    //start the Homing
57
58    while(!TargetReached() & !NewSetPointAcknowledge()) //wait while homing is not complete (Bit 10 and Bit 12 are not set yet)
59    {
60        yield();
61    }
```

- Line 63 to 64: After homing success, we set bit 4 (= control word) back to 0. We also switch the state machine **off**.
- Line 66 to 68: Finally, we set object `0x2300:00` to 0. This ends program execution and prevents a rerun / remapping from the very beginning.

```
63    NewSetPoint(false);
64    Shutdown();    //switch off
65
66    od_write(0x2300, 0x00, 0);    //stop the NanoJ-Program
67    yield();
68 }
```

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](http://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](mailto:info@nanotec.de) | [www.nanotec.com](http://www.nanotec.com)