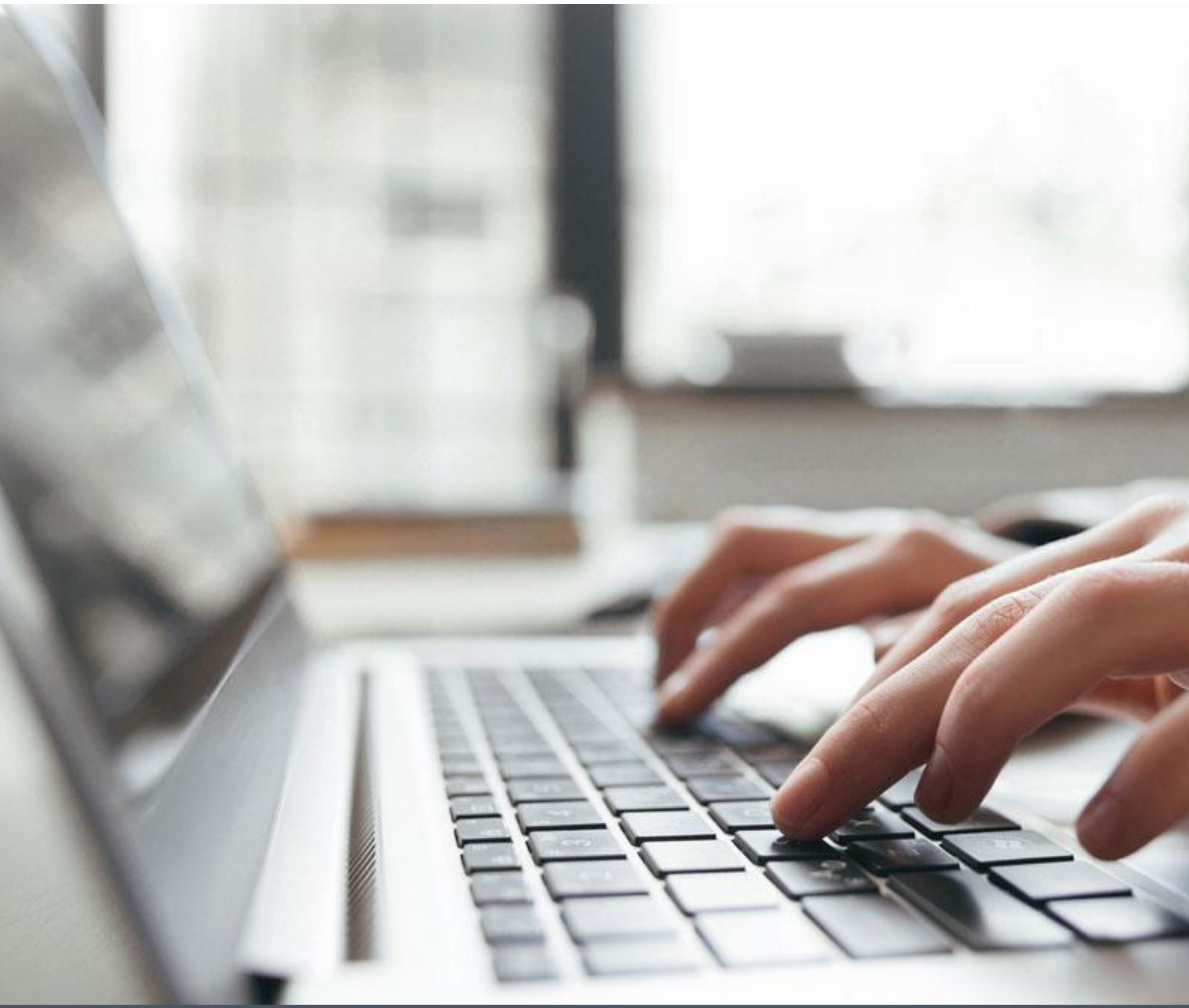


Application Note

How to use *Analog Position* in *NanoJ*

Version 1.0.0

www.nanotec.com

Contents

1 Intended use and audience

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

Analog Position offers a NanoJ code template for setting the target position 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.

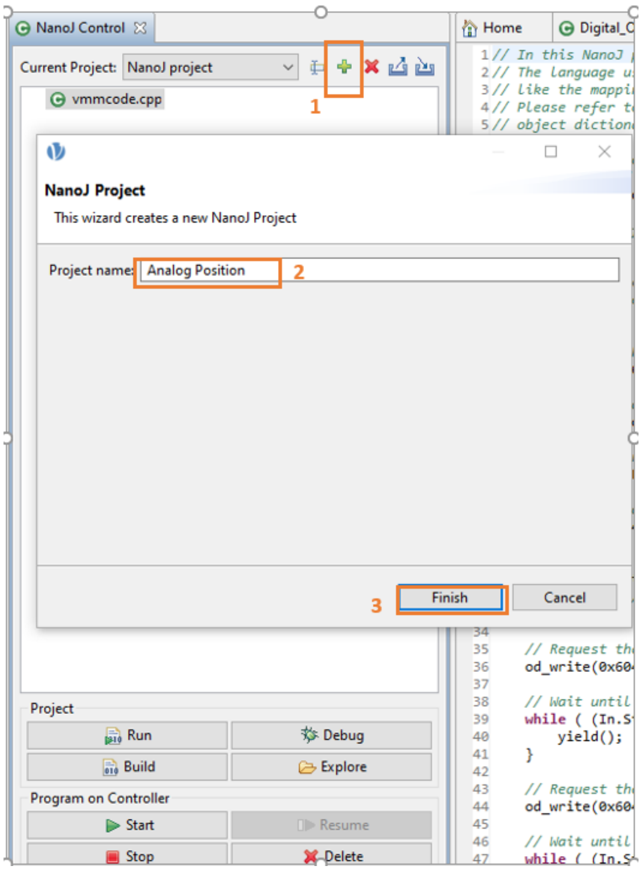
You must have the correct Plug & Drive Studio version installed on your computer:

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:

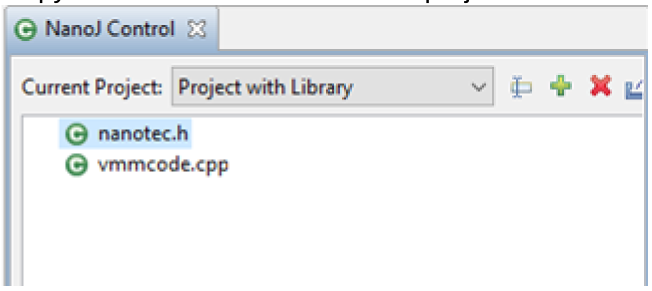
1. Assign a new project name (2).
2. Click on *Finish* (3) to close the tab.
3. Your new project is now created.



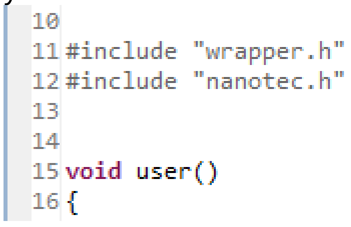
4 Including the nanotec.h library into your NanoJ project

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:



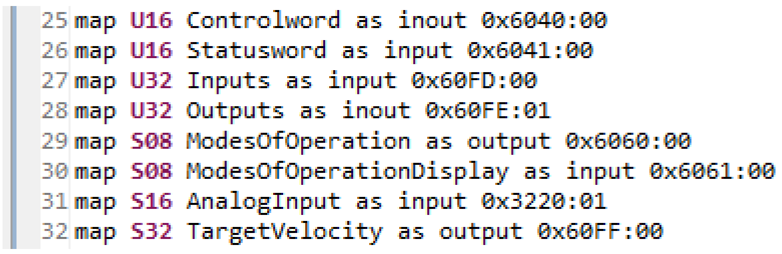
```

#include "wrapper.h"
#include "nanotec.h"


void user()
{
```

5 Using the code template for analog positions in NanoJ

In our example, we start the motor in profile position mode, setting the target position via analog input. The controller translates analog input voltage into values from 0 to 1023 digits.

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 add the object mappings in lines 25 to 32 to our code. We also include the libraries `wrapper.h` and `nanotec.h`:



```
25 map U16 Controlword as inout 0x6040:00
26 map U16 Statusword as input 0x6041:00
27 map U32 Inputs as input 0x60FD:00
28 map U32 Outputs as inout 0x60FE:01
29 map S08 ModesOfOperation as output 0x6060:00
30 map S08 ModesOfOperationDisplay as input 0x6061:00
31 map S16 AnalogInput as input 0x3220:01
32 map S32 TargetVelocity as output 0x60FF:00
```

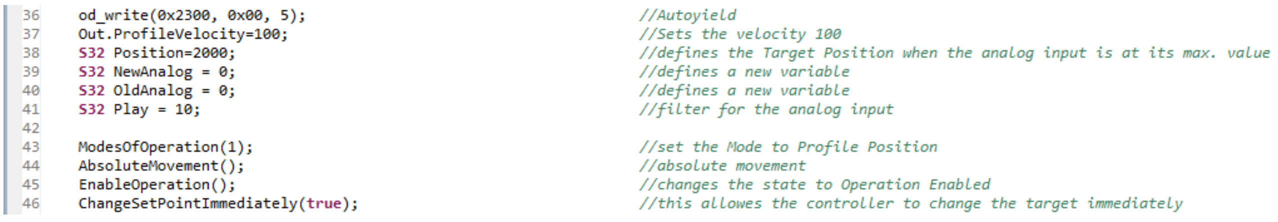
5.2 Main program loop: void user()

5.2.1 Selecting a profile position, defining local variables

A NanoJ program receives computing time cyclically via 1-ms clock. Firmware interrupts and system functions reduce the computing time available for the user program.

The user program must run through the 1-ms-cycle, either by completing it, or by yielding computing time via `yield()` function. On the next cycle, a completion fully restarts the program, whereas a yield simply goes on to the next program command.

- Line 36: If we replace `yield()` with `autoyield()`, we get the same result without a yield call after every function. With `autoyield()`, our vmm code is no longer real-time capable. Also, counters no longer count up every ms, but may be slower.
- Line 38:
- After setting a 100-rpm profile velocity, we define a target position to be reached on analog input at its maximum.
- Line 39, 40: For use in an analog input filter, we define two variables, `NewAnalog` and `OldAnalog`, but also a `Play`.
- Line 45: Via `EnableOperation()`, we power the motor on automatically, without enable signal.
- Line 43: Via `ModesOperation(1)`, we select a profile position.
- Line 44: Via `AbsoluteMovement()`, we set the positioning mode to absolute.
- Line 46: With `ChangeSetPointImmediately()` set to `true`, you activate an immediate setpoint change. Thus, `NewSetPoint()` executes each new travel command immediately, interrupting the current one before reaching its target position:



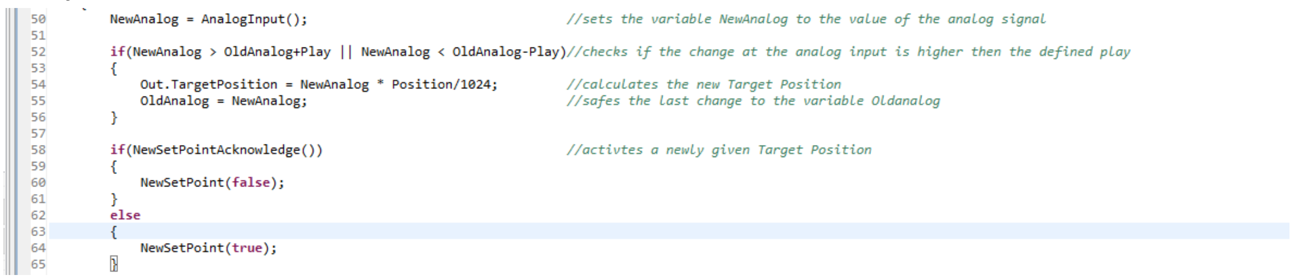
```
36    od_write(0x2300, 0x00, 5);                                //Autoyield
37    Out.ProfileVelocity=100;                                  //Sets the velocity 100
38    S32 Position=2000;                                        //defines the Target Position when the analog input is at its max. value
39    S32 NewAnalog = 0;                                        //defines a new variable
40    S32 OldAnalog = 0;                                        //defines a new variable
41    S32 Play = 10;                                            //filter for the analog input
42
43    ModesOfOperation(1);                                      //set the Mode to Profile Position
44    AbsoluteMovement();                                       //absolute movement
45    EnableOperation();                                        //changes the state to Operation Enabled
46    ChangeSetPointImmediately(true);                          //this allowes the controller to change the target immediately
```

5.2.2 Starting the motor via analog input

Once powered up, the motor is ready to be positioned via analog signals between 0 and 10 V.

- Line 50: First, we set the `NewAnalog` variable to meet the analog input value.

- Line 52: We then implement a filter function to check if the analog input change exceeds the defined `Play`.
- Line 54: If the change was big enough, our default formula calculates a new target position for our motor.
- Line 55: We then write the new analog value to the `OldAnalog` variable for the filter in the next cycle.



```
50          NewAnalog = AnalogInput();                              //sets the variable NewAnalog to the value of the analog signal
51
52          if(NewAnalog > OldAnalog+Play || NewAnalog < OldAnalog-Play)//checks if the change at the analog input is higher then the defined play
53          {
54              Out.TargetPosition = NewAnalog * Position/1024;         //calculates the new Target Position
55              OldAnalog = NewAnalog;                                  //safes the last change to the variable Oldanalog
56          }
57
58          if(NewSetPointAcknowledge())                            //activtes a newly given Target Position
59          {
60              NewSetPoint(false);
61          }
62          else
63          {
64              NewSetPoint(true);
65          }
```

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