User's Manual

Nano Stage

(ST01, PDV01)

25.5.14

Sejong Scientific Instruments Inc.

SEJONG
Scientific
Instrumunts

Title of equipment: Nano Stage

Model: ST01, PDV01

Product Description:

This product combines a stepper motor-driven motorized stage with an encoder offering 10 nm resolution, effectively compensating for backlash. The driver integrates a microcontroller with a 1/256 micro-step based motor driver, and employs advanced control algorithms to enable precise control at the 10 nm level.

included items: X, XY, XYZ stage, motor driver, labtop PC

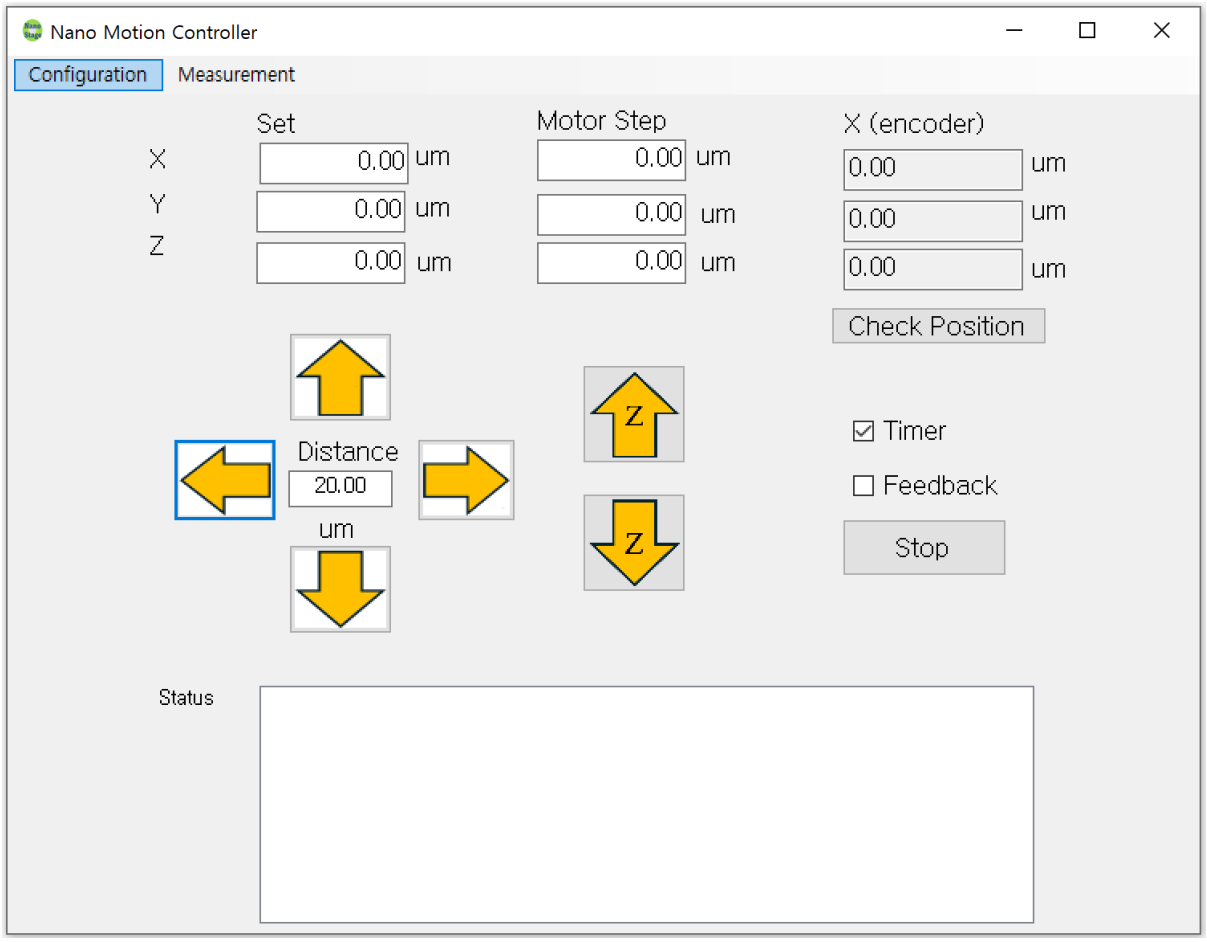
Hardware Description:

Connection:

Connect the PC and the motor driver using a USB cable and connect the motor driver to the stage using the provided D-sub cables.
Each axis requires two D-sub cables—one for motor operation and one for the encoder—both must be connected.

Software Description:

This application runs on Windows 10 and 11.



Main Display

"**Set**" is the input field where the user enters the desired position.
"**Motor Step**" displays the step value determined by the motor driver, shown for reference.
"**X (encoder)**" indicates the actual position of the stage measured by the encoder.

When the user clicks the "**Check Position**" button, the encoder's current measured position is displayed in the "**X (encoder)**" field.

The arrow buttons allow the user to manually change the position using the mouse. The stage moves by the distance specified in the "**Distance**" field. The value of "Distance" can be freely adjusted by the user.

When the "**Timer**" checkbox is checked, the encoder value is continuously updated in real-time.

When the "**Feedback**" checkbox is checked, the system performs continuous feedback control to ensure that the encoder-measured value matches the "**Set**" value. If the checkbox is not checked, feedback control is disabled.

The "**Status**" window displays the current control status of the motor driver.

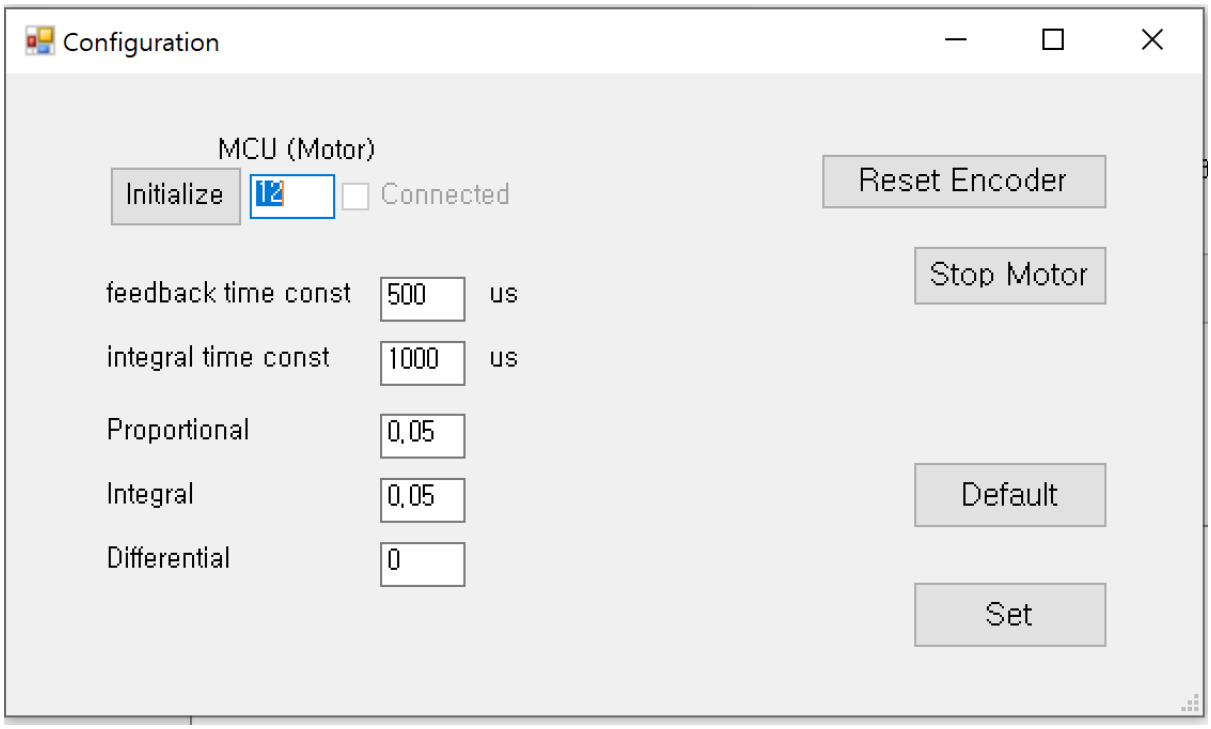


Figure 5: Configuration Menu

The Configuration Menu allows users to modify the overall settings for writing conditions and connected devices.

MCU Port Number:

• When the Arduino processor is properly connected, the "Connected" checkbox will be marked.

• If the connection fails, you need to identify the correct COM port number for the microcontroller:

- Open **Device Manager** in Windows.
- Check the port number assigned to the MCU device under **Ports (COM & LPT)**.

• Enter the identified port number into the input field and click the **Initialize** button.

• Upon successful connection, the "Connected" checkbox will be marked.

- **"Feedback Time Const"** displays the time constant used for feedback control.

- **"Integration Time Const"** shows the integration time constant.
- **"Proportional"** displays the gain value used for proportional (P) control.
- **"Integral"** displays the gain value used for integral (I) control.
- **"Differential"** displays the gain value used for derivative (D) control.
- **"Reset Encoder"** is a button that resets the encoder value.
- **"Stop Motor"** is a button that forcibly stops motor movement.
- **"Default"** resets all PID control parameters to their original factory settings.
- **"Set"** applies the currently modified parameter values.

User Application Development:

For user programming, the system follows the UART serial communication standard. It uses standard communication functions provided by common programming languages to send and receive the text-based commands listed below.

On a Windows PC, you must first check the port number in Device Manager and open the corresponding port.

For example, when using C++ with the Windows API (Win32), you can use functions like the following.

```
#include <windows.h>
        o  CreateFile()   // Open port
        o  ReadFile(), WriteFile()   // Send/Receive Functions
        o  SetCommState(), GetCommState() – // port configuration
```

In Python, you can use functions like the following:

```
import serial
ser = serial.Serial('COM4', 9600)   # port, data rate
ser.write(b'Hello')                  # send data
data = ser.readline()                 # receive data
ser.close()
```

Main functions:

serial.Serial()

read(), readline(), write(), in_waiting

Serial Communication Commands for User Programming:

- **Command "m"**: Moves the motor to the specified position, ignoring the encoder value.
- **Command "v"**: Moves the stage to the specified position based on the encoder reading.

The "m" or "v" command must be followed by:

1. A number (0, 1, or 2) representing the axis,
2. A space,
3. Another number indicating the target absolute position.

- **"p"**: Position request — asks the system to send the current position.
- **"s"**: Sets the movement speed. An integer follows the "s" command to specify speed. The maximum recommended speed is 64000. Values above this may cause the motor to malfunction.
- **"cl"**: Checks the limit switch — verifies whether the stage is within the allowed range.
- **"fb 1"**: Enables feedback control while in a stationary state.
- **"fb 0"**: Disables feedback control while in a stationary state.

Example:

1. "m 0 5000": Move the X-axis by 5000 steps unconditionally, ignoring the encoder.
2. "v 1 5000": Move the Y-axis to position 5000 steps based on the encoder value.
3. "s 64000": Set the movement speed to 64000.

Commands for Maintenance Technicians

These commands are used to adjust parameters for PID control and are not recommended for modification by general users.

"ft 200": feedback time constant as 200 us

"fg 2000": integral time constant as 2000 us

"fp 10": feedback proportionality setting as 0.1

"fi 50": feedback integration setting as 0.5

"fd 3": feedback differential setting as 0.03


SEJONG
Scientific
Instrumunts