# Teaching materials

# Teacher guide notes – Mechatronics Motion Platform

## MISCE project

Mechatronics for Improving and Standardizing Competences in Engineering

Competence: Mechatronics

Workgroup: Universidade do Minho

This document corresponds to the Teacher guide notes – Mechatronics Motion Platform, for the competence ‘Mechatronics’.

Version: 1.0

Date: October 5th, 2023

Visit https://misceproject.eu/ for more information.

# Index of contents



# Index of figures

# 1 Motion - Axis Configuration 1 - Hardware

## 1.1 Motion Control

To control one or more axes from the NJ controller, it is necessary to associate the project with the hardware information present in the EtherCAT network (machine network).

To do so, when starting the programming of the Motion Control module, we must set up the system so that it knows what it is controlling.

## 1.2 EtherCAT Network

To obtain the EtherCAT network automatically:

- In the **Multiview Explorer** window, under **Configurations and Setup > EtherCAT**, double-click.

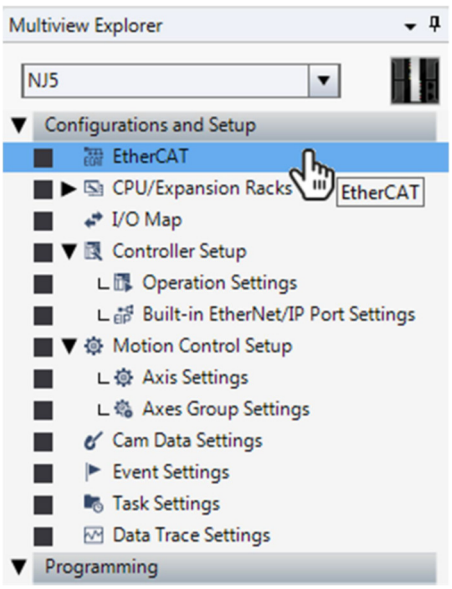



Figure 1. Multiview Explorer window

- The workspace will display the EtherCAT network diagram.
- Initially, only the NJ controller appears in the network. To scan the network, switch to **Online** mode.

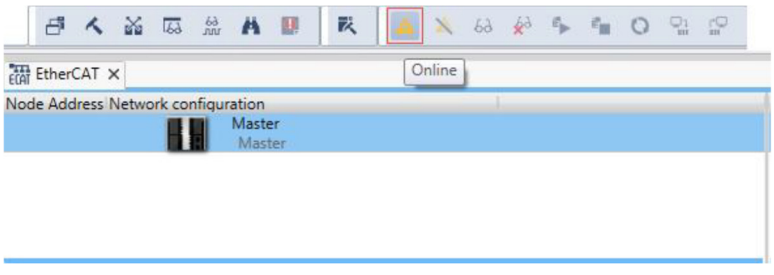



Figure 2. Scan network in online mode

- Right-click on the NJ controller and select **Compare and Merge with Actual Network Configuration**.

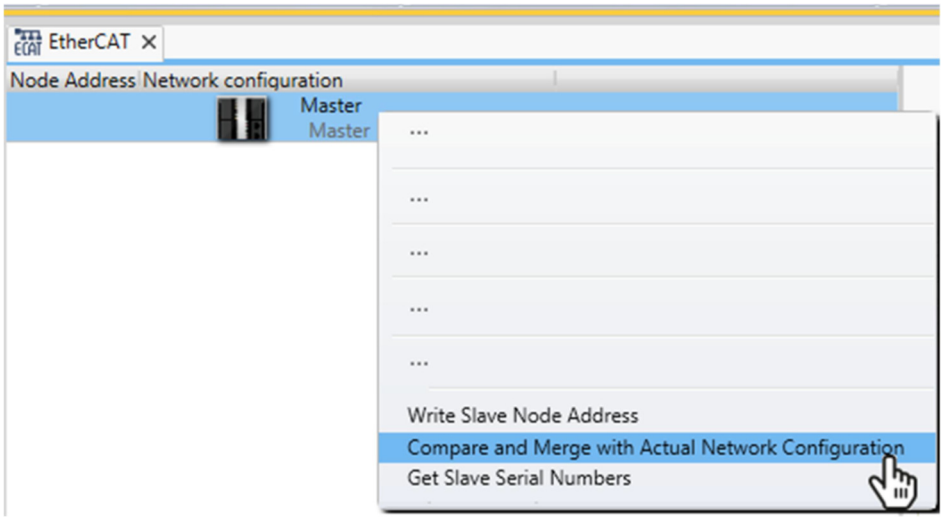



Figure 3. Compare and Merge with Actual Network Configuration

- The left side shows the project data, and the right shows the live network configuration.

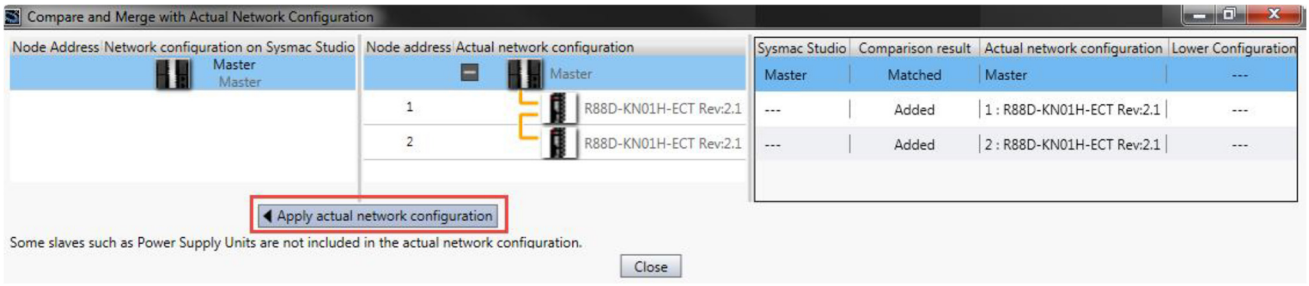
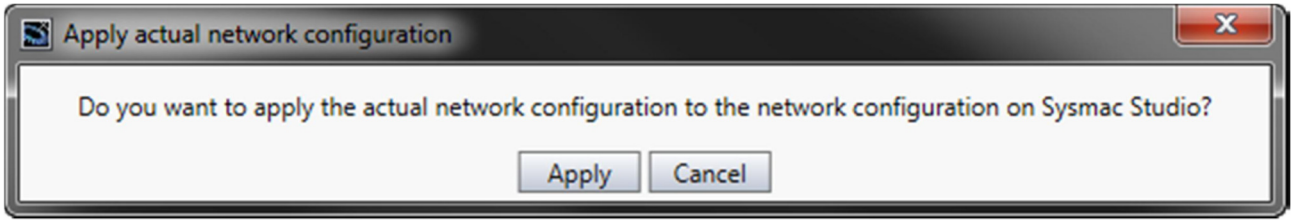



Figure 4. Left side shows the project data and the right shows the live network configuration

- Click **Apply actual network configuration** to incorporate the network into the project.




- Confirm when prompted and click **Close** to finalize.

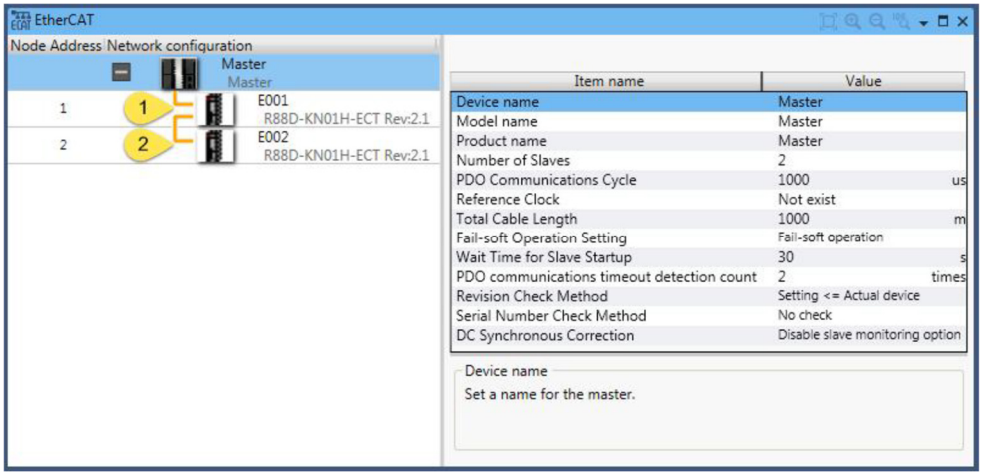



Figure 5. Project network configuration

## 1.3 CPU BUS

To configure the CPU/bus network, use the same approach.

- Go to **Configurations and Setup > CPU/Expansion Racks**.

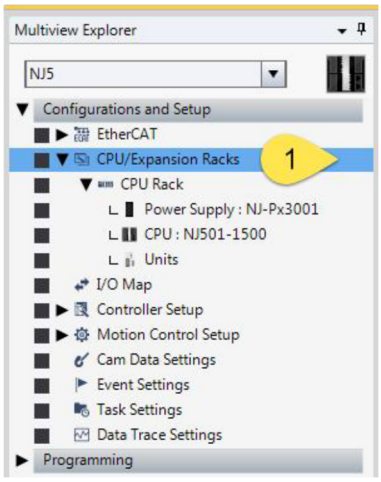



Figure 6. CPU/Expansion Racks

- If there are modules on the bus, right-click and select **Compare and Merge with Actual Unit Configuration** to apply it to the project.

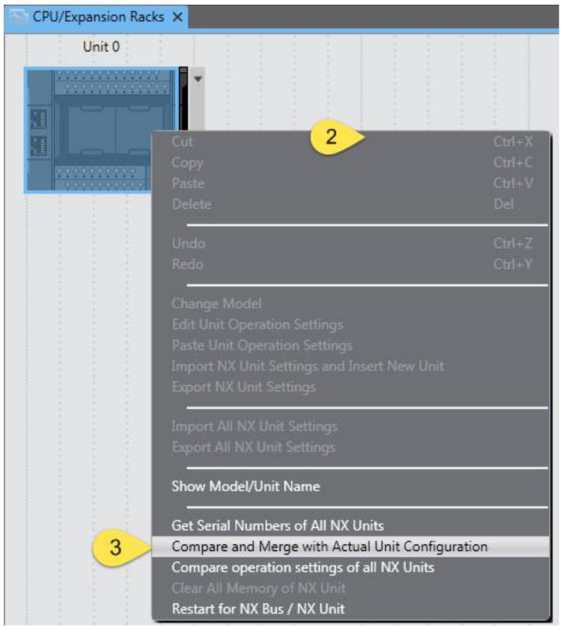



Figure 7. Compare and Merge with Actual Unit Configuration

## 1.4 System Setup - Drive Configuration

- Based on the updated project, click: **Project > Rebuild Controller > Synchronize > Transfer to Controller**.
- Right-click on the corresponding servo drive node and select **Online**.

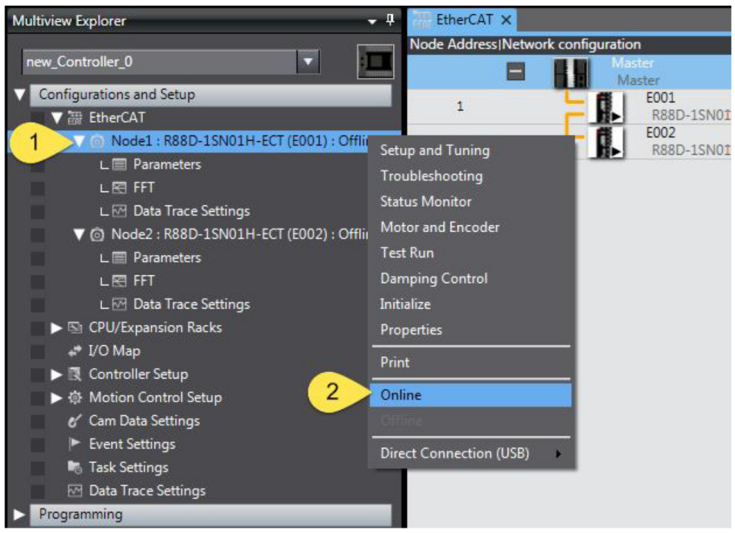



Figure 8. Servo drive node and Online mode selection

- Then, right-click **Node 1 > Parameters** and choose **Setup and Tuning**.

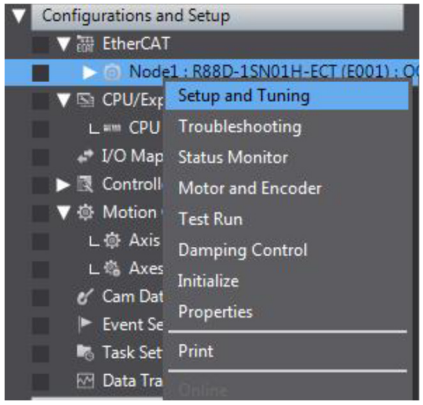



Figure 9. Setup and Tuning selection

- A configuration and tuning window opens.

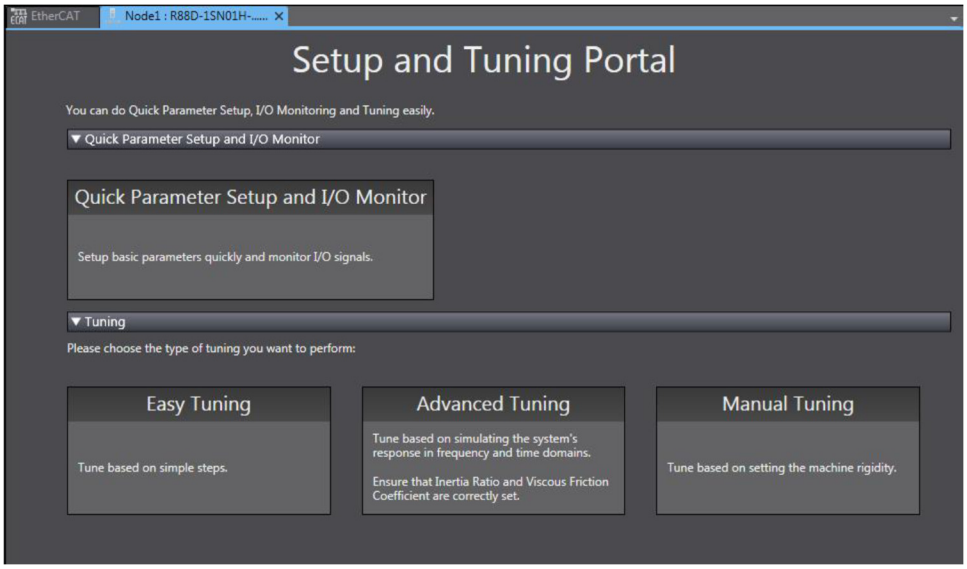



Figure 10. Setup and Tuning selection window

- Click **Quick Parameter Setup and I/O Monitor** to open the Sysmac Studio wizard.

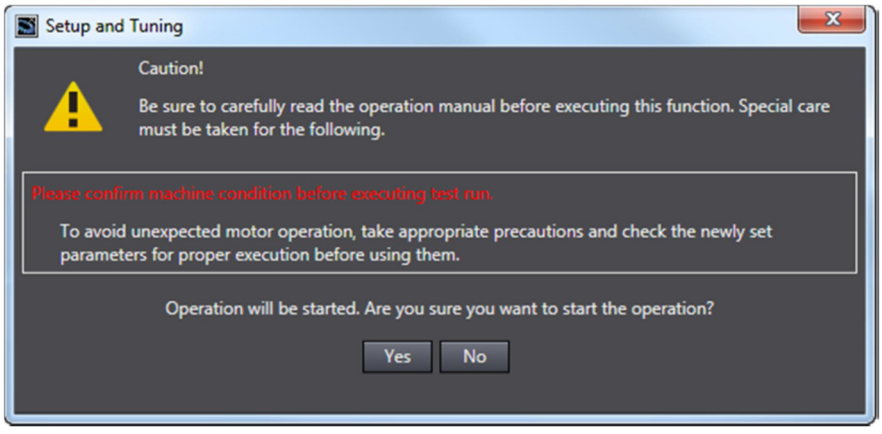



Figure 11. Quick Parameter Setup and I/O Monitor

- A warning appears indicating that the configuration might affect machine behavior.
- Confirm to proceed.

### Step 1: Motor and Encoder Configuration

- The window will display motor parameters.

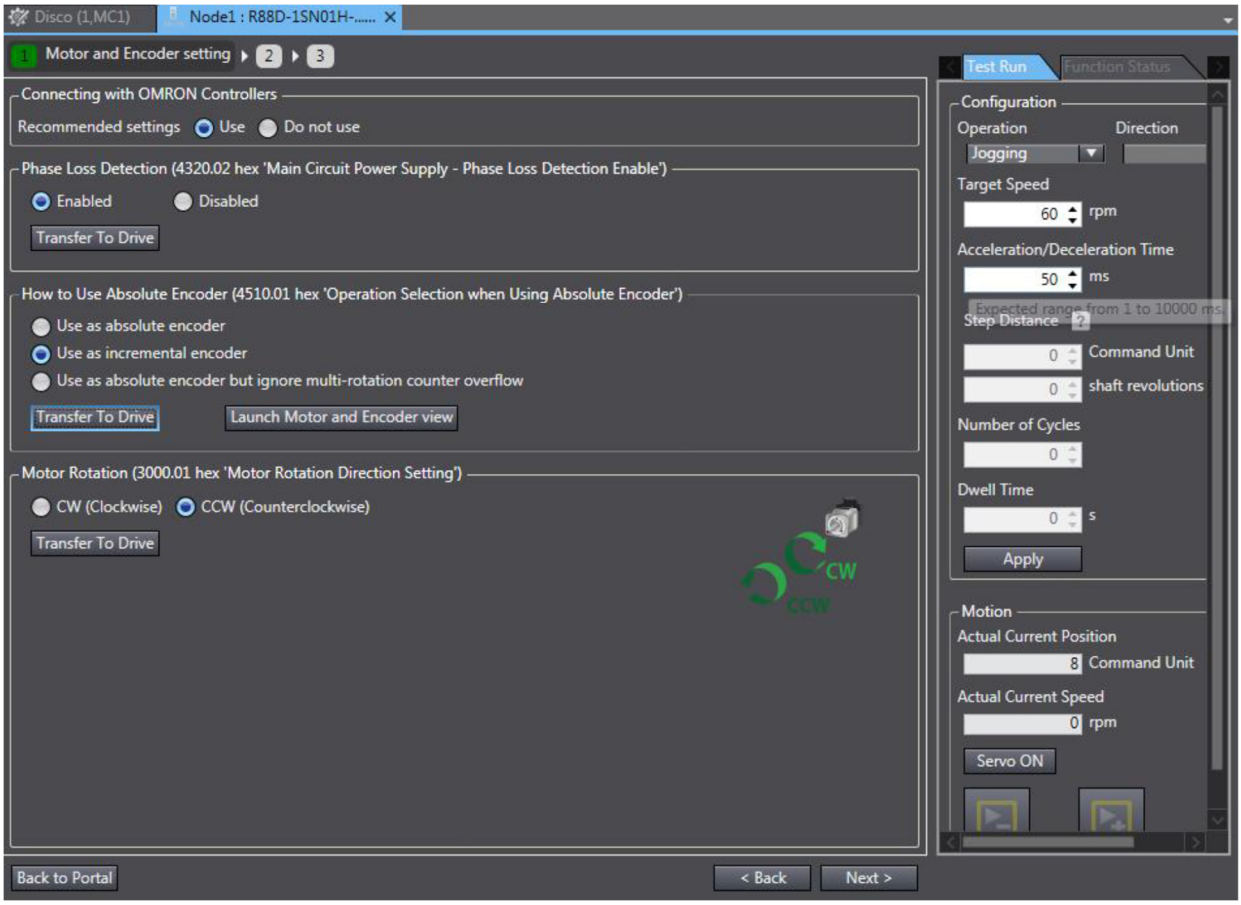



Figure 12. Quick Parameter Setup and I/O Monitor

- Confirm each setting and click **Transfer to Drive**.
- After transferring, Sysmac Studio will prompt to restart the drive. Click **Yes**.

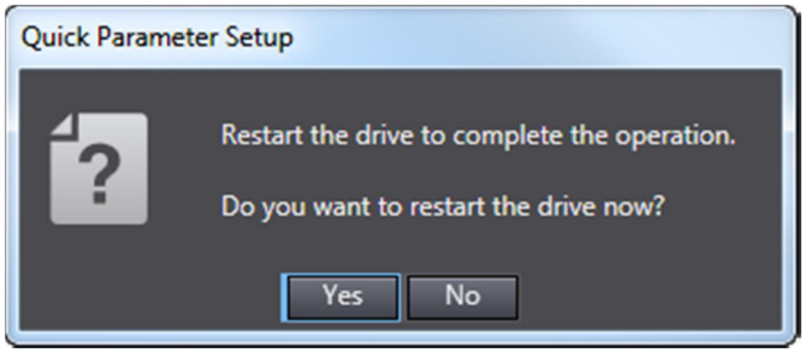



Figure 13. Asking for restart the drive

### Step 2: Input Configuration

- Select input configuration and click **Transfer to Drive**.

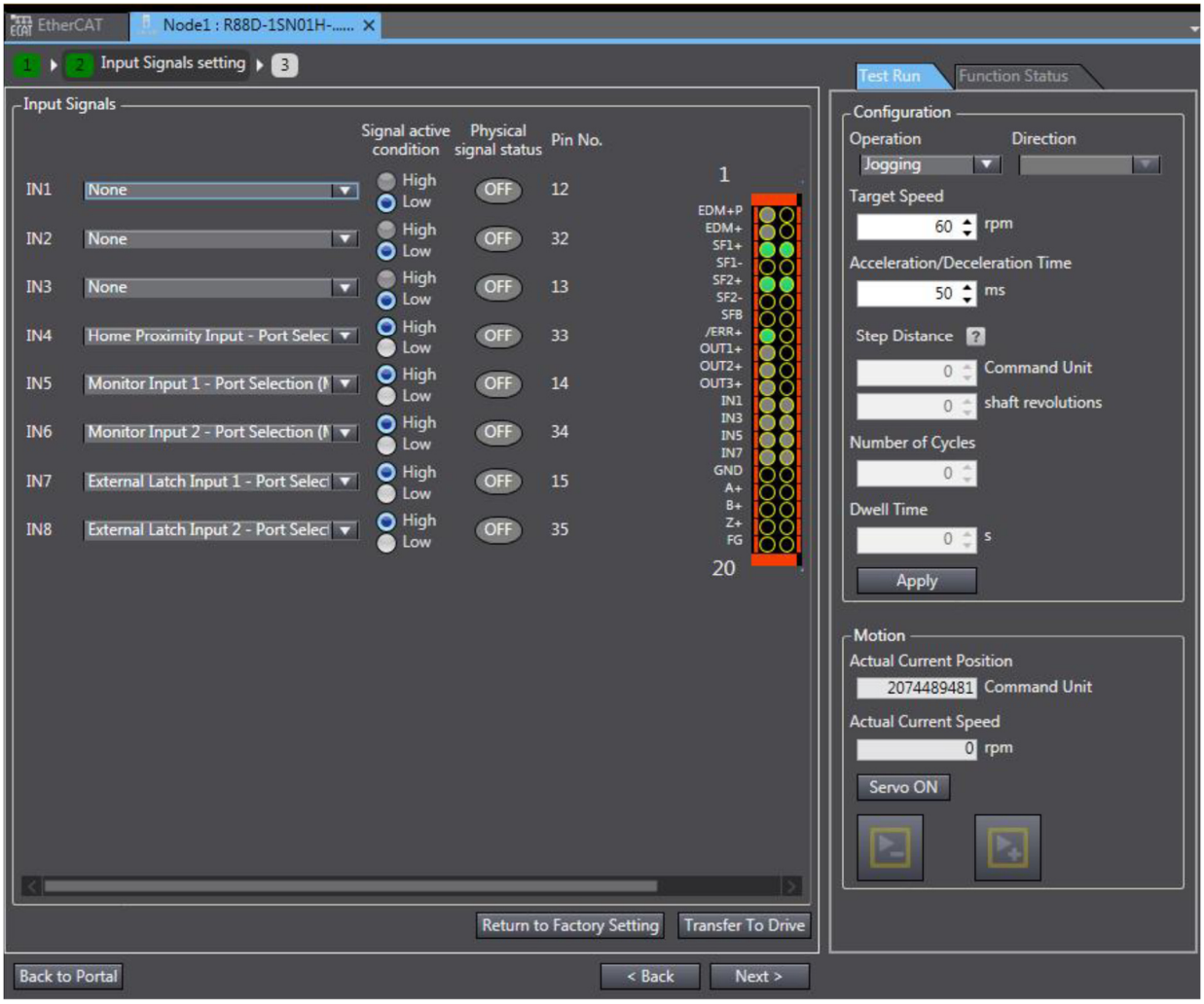



Figure 14. Selecting input configuration

- Then click **Next**.

### Step 3: Output Configuration

- Select output configuration, click **Transfer to Drive**, then click **Finish**.

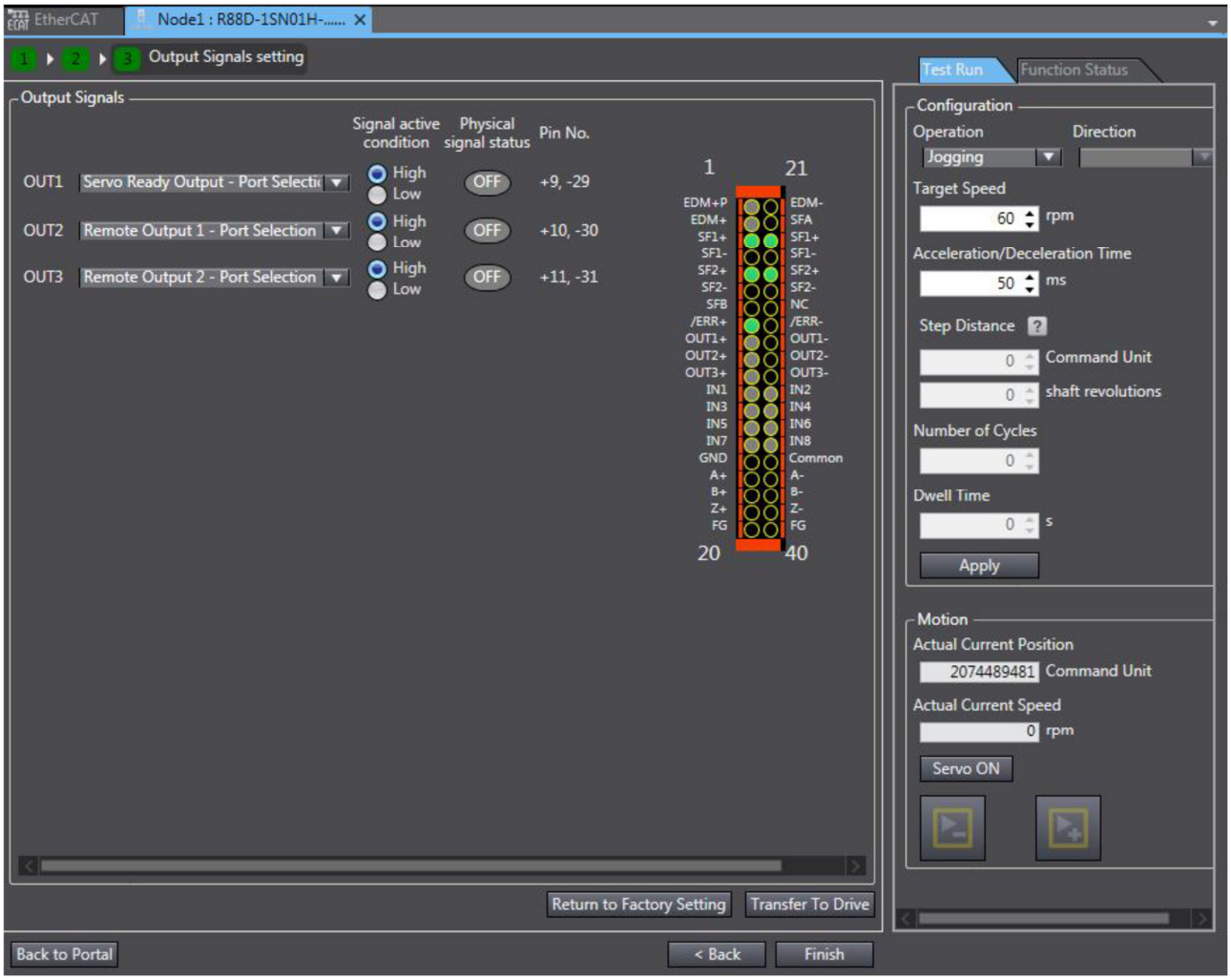



Figure 15. Output Configuration

- After configuring the first servo drive, switch Node 1 to **Offline**.

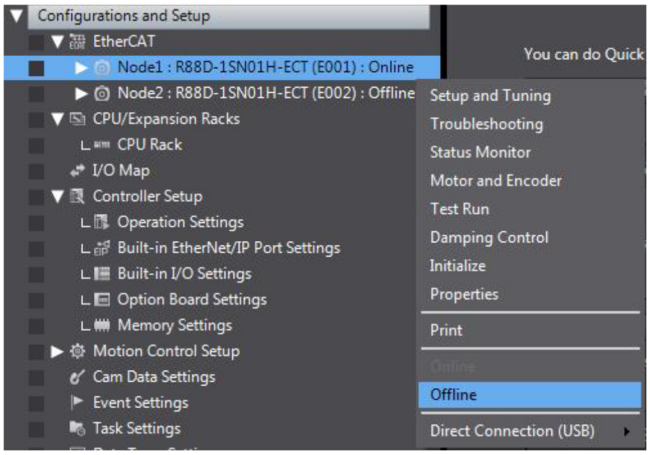



Figure 16. Configuring the first servo drive and switch Node 1 to Offline

- Repeat the process for the second node.

## 1.5 Axis Configuration

After validating the EtherCAT network and configuring the drives, the axes can be defined.

- Go to **Configurations and Setup > Motion Control Setup > Axis Settings**.
- Right-click and select **Add > Axis Settings** twice (for two axes).

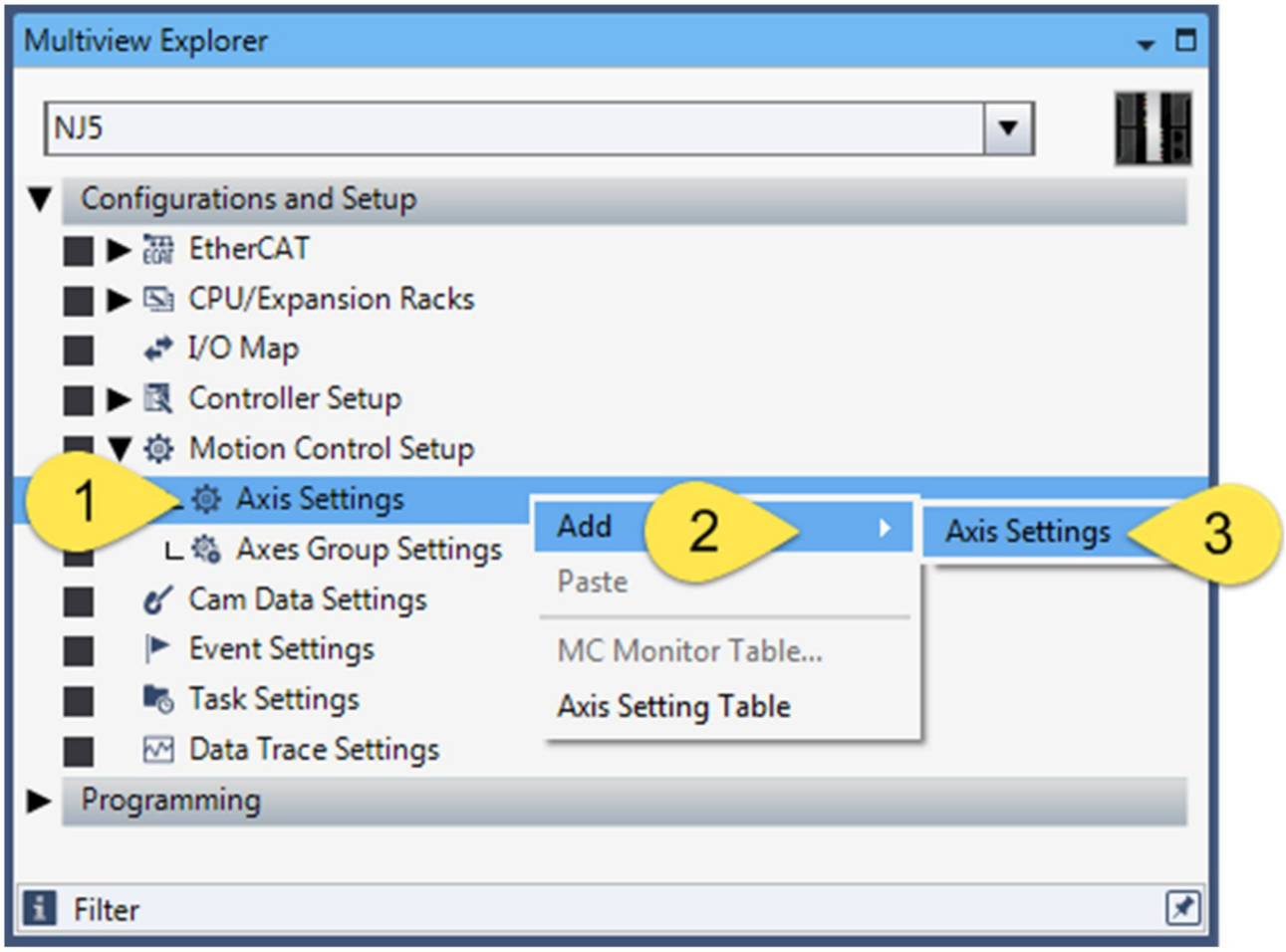



Figure 17. Axis definitions

- Example names:
  - MC_Axis000 (0) → renamed to Axis1 → Conveyor
  - MC_Axis001 (1) → renamed to Axis2 → Disk

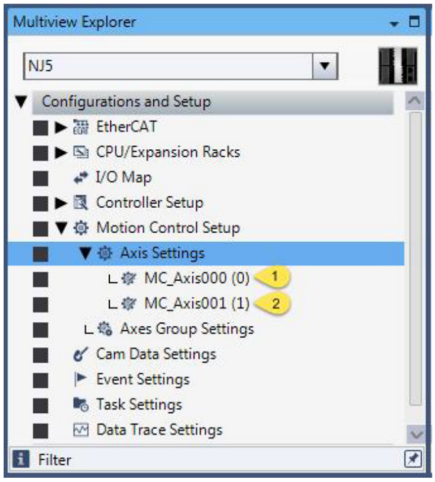
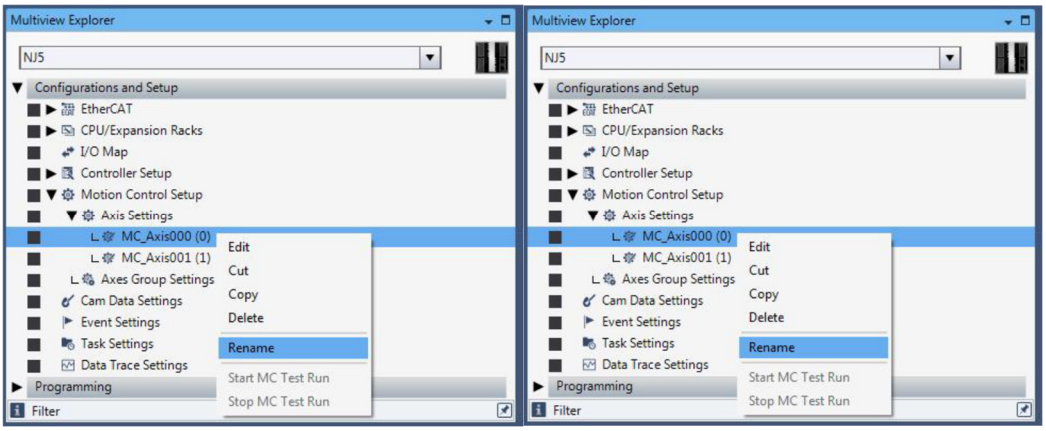
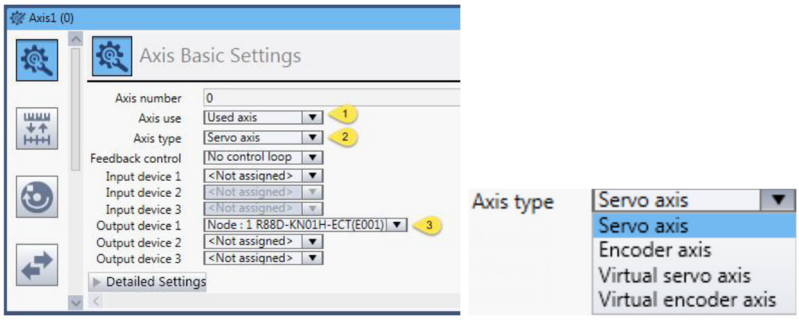



Figure 18. Axis name definitions

- Right-click on each axis to rename them appropriately.
- Double-click **Axis1** and assign **Node 1** as the output device:
    - Axis Use: Used Axis
    - Axis Type: Servo Axis
    - Output Device: Node 1 R88D-KN01H-ECT (E001)
- Repeat for **Axis2** using **Node 2**.
- Then execute:
    - **Project > Rebuild Controller > Online > Synchronize > Transfer to Controller**



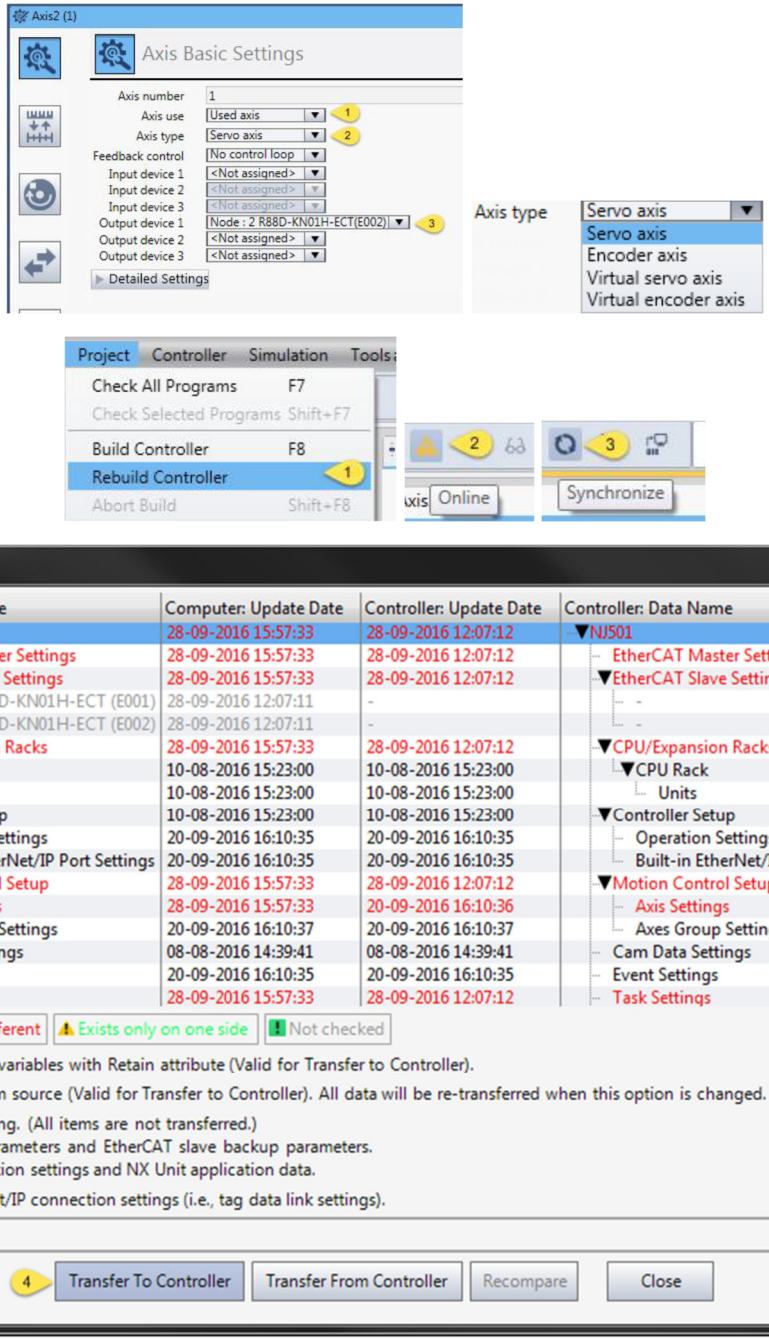



Figure 19. Final configurations

## 1.6 Start Motion Control (MC) Test Run

- Right-click on **Axis1 (0)** > **Start MC Test Run**

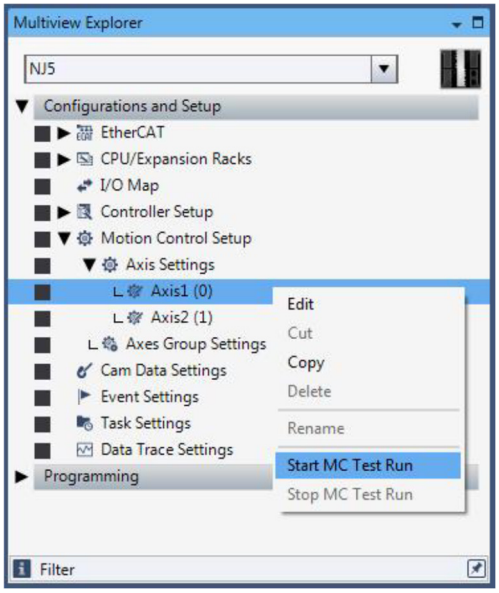



Figure 20. Start MC Test Run

- A warning will appear because the servomotor will rotate. Take precautions and confirm.

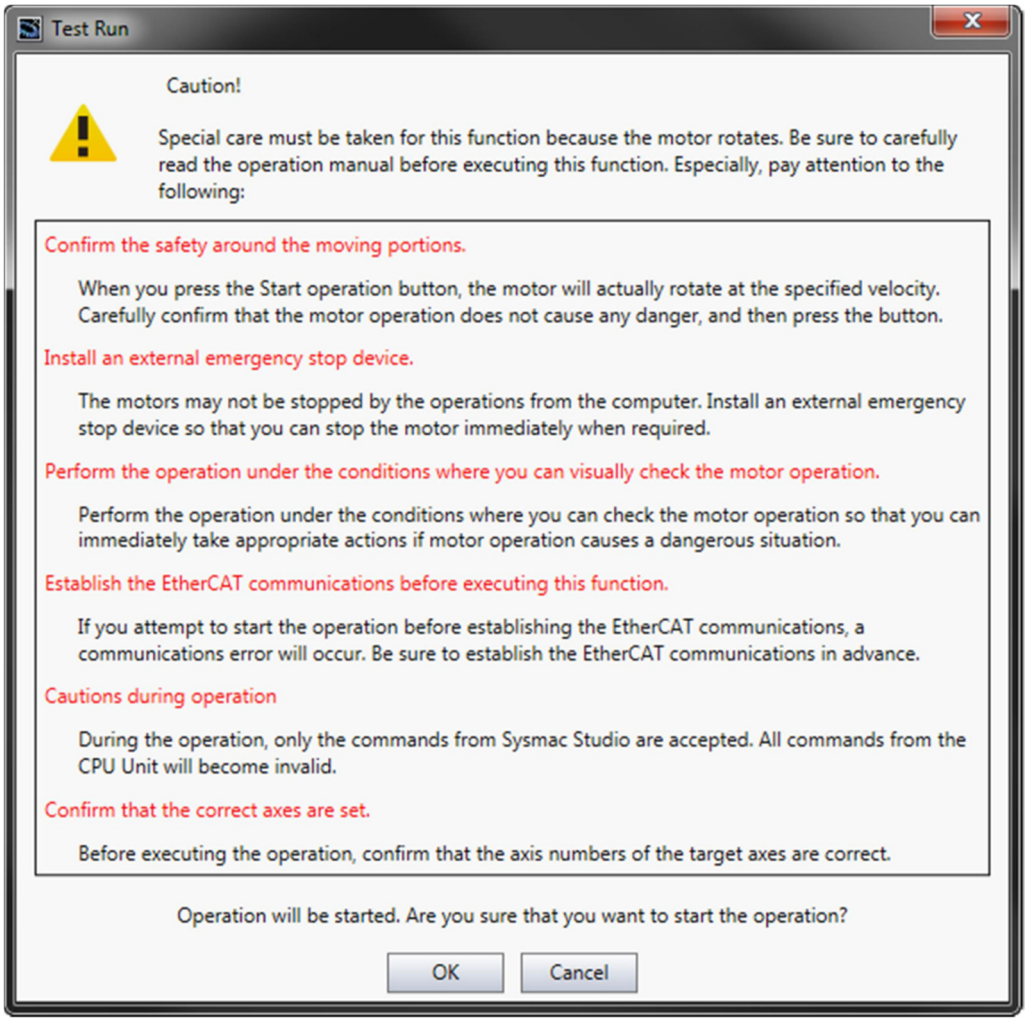



Figure 21. Start MC Test Run

**Test Run Interface includes:**

1. Axis selection
2. Axis state and error list
3. Axis ON/OFF
4. MC tabs: Jogging, Absolute Positioning, Relative Positioning, Homing
5. Speed, distance, acceleration, etc.
6. Execute MC commands

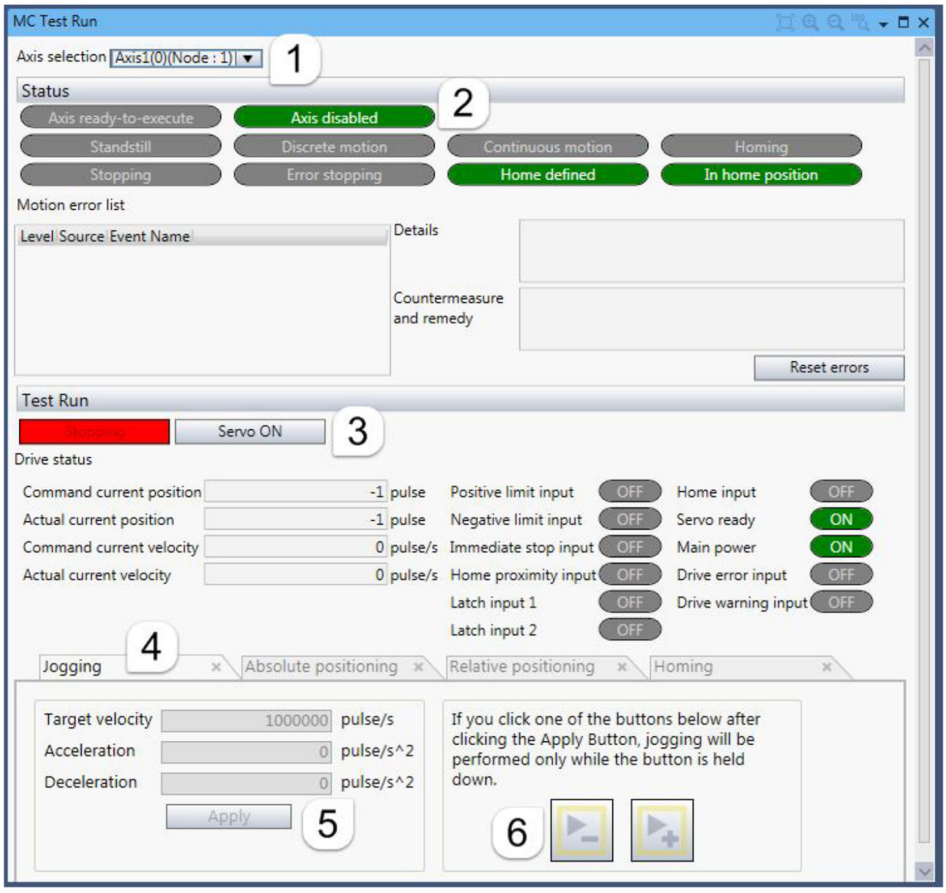
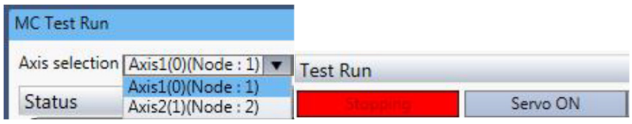



Figure 22. Test Run Interface steps

**Example:**

- Set the axis to ON. The button will switch to OFF.
- Under **Jogging**, click **Apply**.
- Click **MC Jogging +** → motor rotates counterclockwise (CCW), position increases.
- Click **MC Jogging -** → motor rotates clockwise (CW), position decreases.



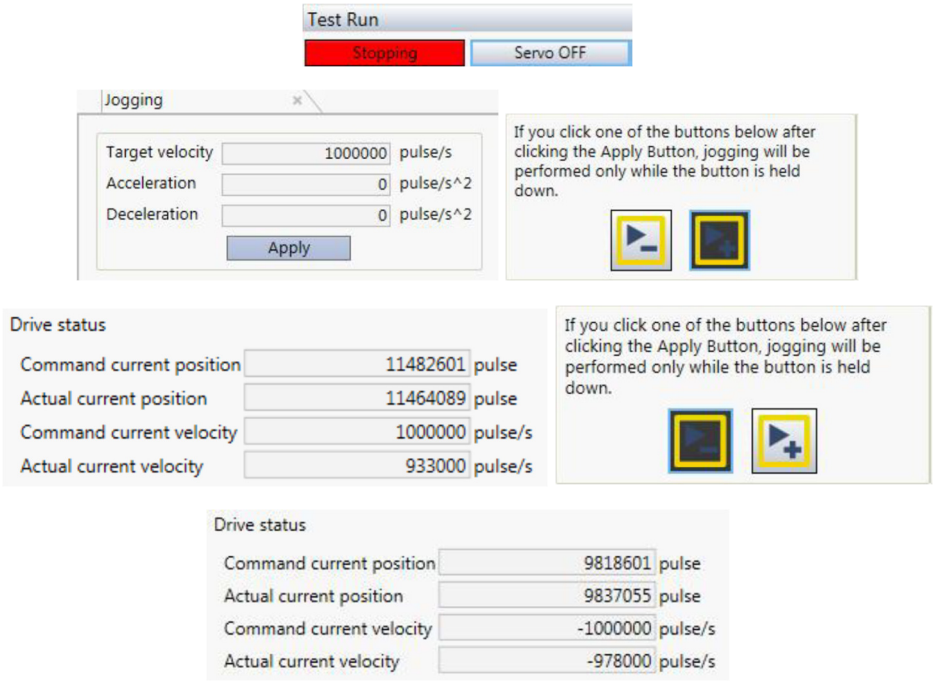



Figure 23. Example configuration

## 1.7 MC Commands

- **Jogging:** moves the motor in + or – direction at the defined speed.

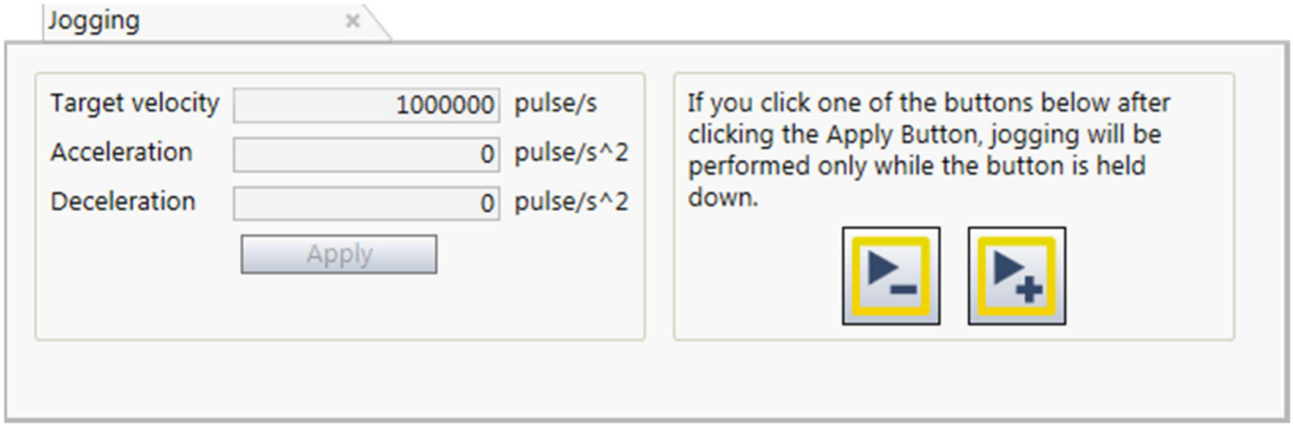



Figure 24. Jogging

- **Absolute Positioning:** moves motor to defined position from origin.

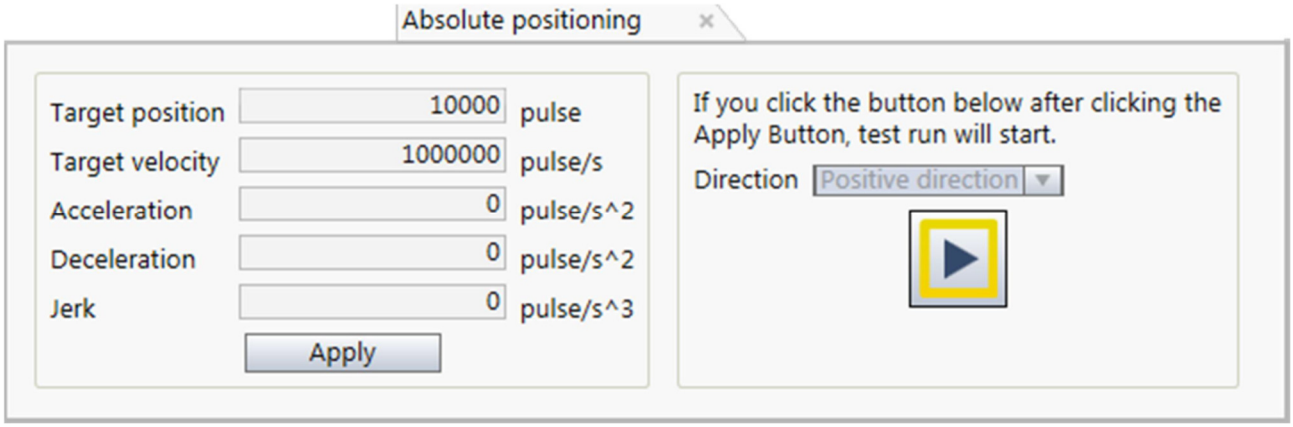



Figure 25. Absolute Positioning

- **Relative Positioning:** moves motor a defined distance from current position.

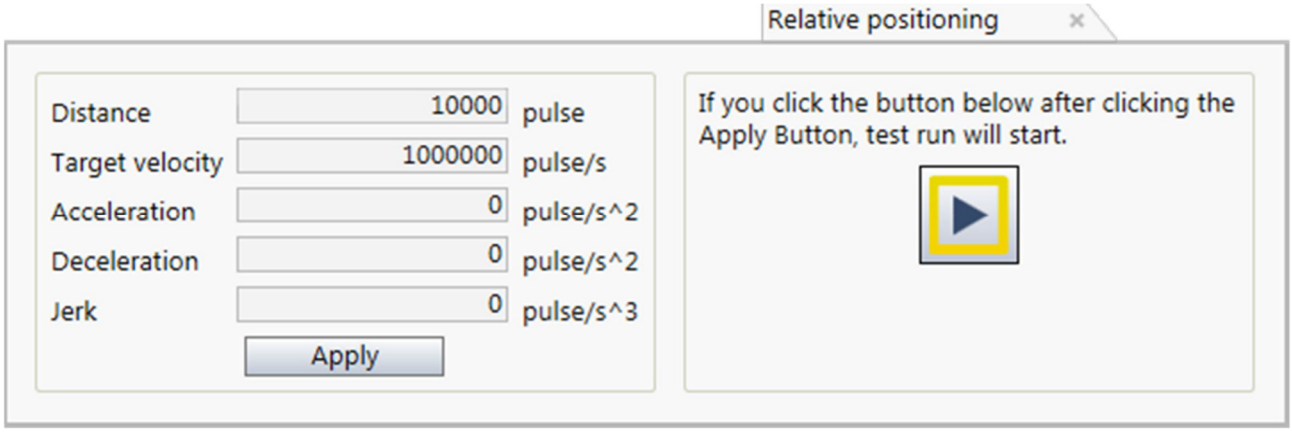



Figure 26. Relative Positioning

- **Homing:** defines origin, may use limits, proximity, or home sensors.

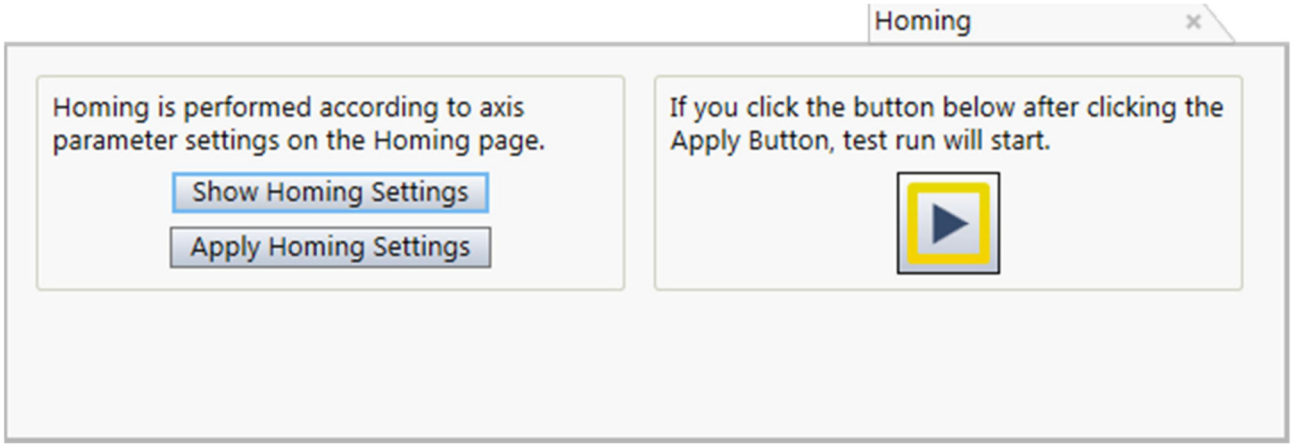



Figure 27. Homing

# 2 Motion - Axis Configuration 2 - Hardware

## 2.1 Axis Configuration – Axis1 [Conveyor]

After assigning the hardware (servo drive) to the axis via **Axis Basic Settings** and validating correct operation via **MC Test Run**, we continue configuration via **Motion Controller Setup > Axis Settings**.

| Button | Name | Description |
|---|---|---|
| | Axis Basic Settings | Set whether to use the axis, the axis type, the axis number, an input or output device, and the channel. |
| | Unit Conversion Settings | Set the gear ratio of the electronic gear.<br>Set the pulses per motor rotation and the travel distance. |
| | Operation Settings | Set the velocity, acceleration rate, deceleration rate, torque warning values, and other monitor parameters. |
| | Other Operation Settings | Set the I/O for the Servo Drive. |
| | Limit Settings | Set software limits and following error limits. |
| | Homing Settings | Set the homing operation. |
| | Position Count Settings | Set Count Mode of the Controller. |
| | Servo Drive Settings | Set the Servo Drive parameters. |

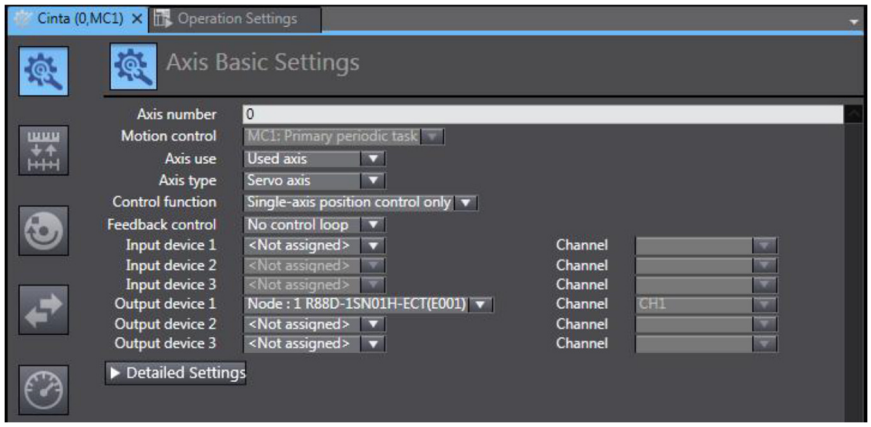



Figure 28. Axis Settings

## 2.2 Unit Conversion Settings

**Unit Conversion Settings**

- Configure units for **Axis1 (Conveyor)** in **Unit Conversion Settings**.
- Based on mechanical data:
  - Each motor rotation moves the conveyor **110 mm**
  - Encoder: 23-bit → **8,388,608 pulses/rev**

**Settings:**

1. Working unit: mm
2. Pulses per rotation: 8,388,608
3. Distance per rotation: 110 mm

→ After these values are set, the **Operation Settings** icon shows an error until configured.

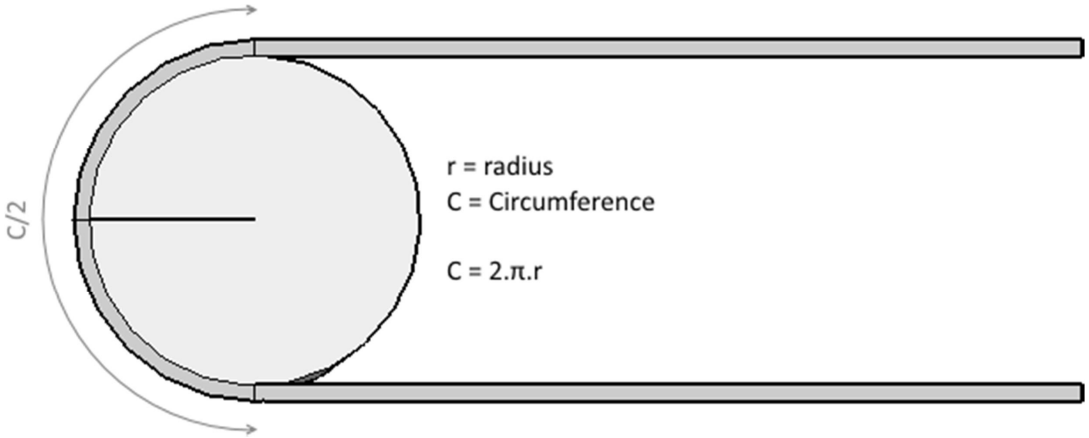



Figure 29. Unit Conversion Settings

For the existing 1S servo drive / servo motor, we have:

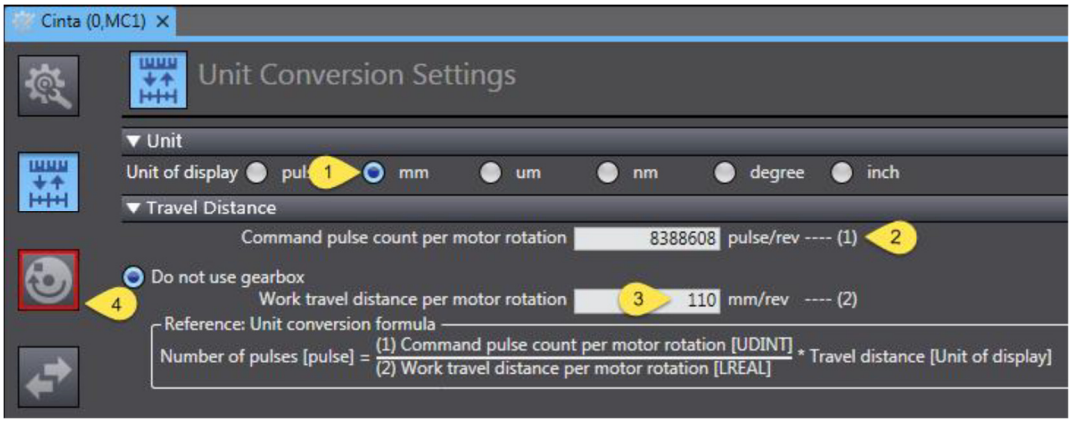
- 23-bit incremental encoder;
- 3,000 rpm.

**Standard Servo Motors**

Servo motors 1,000 – 2,000 – 3,000 r/min

100 to 3,000 Kw

| Specifications | | | Servo Drive |
|---|---|---|---|
| Voltage | Encoder | Capacity | 1S |
| 230 V | Absolute encoder (23 bit) | 50 W | R88D-1SN01H-ECT |




**(1)** Definition of the working unit, in this case **mm**

**(2)** Pulses per motor revolution

The motor generates $2^{23} = 8{,}388{,}608$ pulses per revolution, based on its 23-bit incremental encoder.

**(3)** Distance traveled per rotation: 110 mm

**(4)** After defining these values, the **Operation Settings** icon will indicate an error.

In the next step, we must configure the **Operation Settings** for axis 1.

## 2.3 Operation Settings

Velocity calculations:

- Max Velocity:

  $RPM$ / 60 = RPS

  3000 / 60 = 50 RPS

  50 RPS × 110 mm = **5,500 mm/s**

- Max Jog Velocity:

  10 RPS × 110 mm = **1,100 mm/s**

## 2.4 Home Settings

- Since Axis1 controls a conveyor, homing is configured so that when executed, the current position is reset to **zero**.

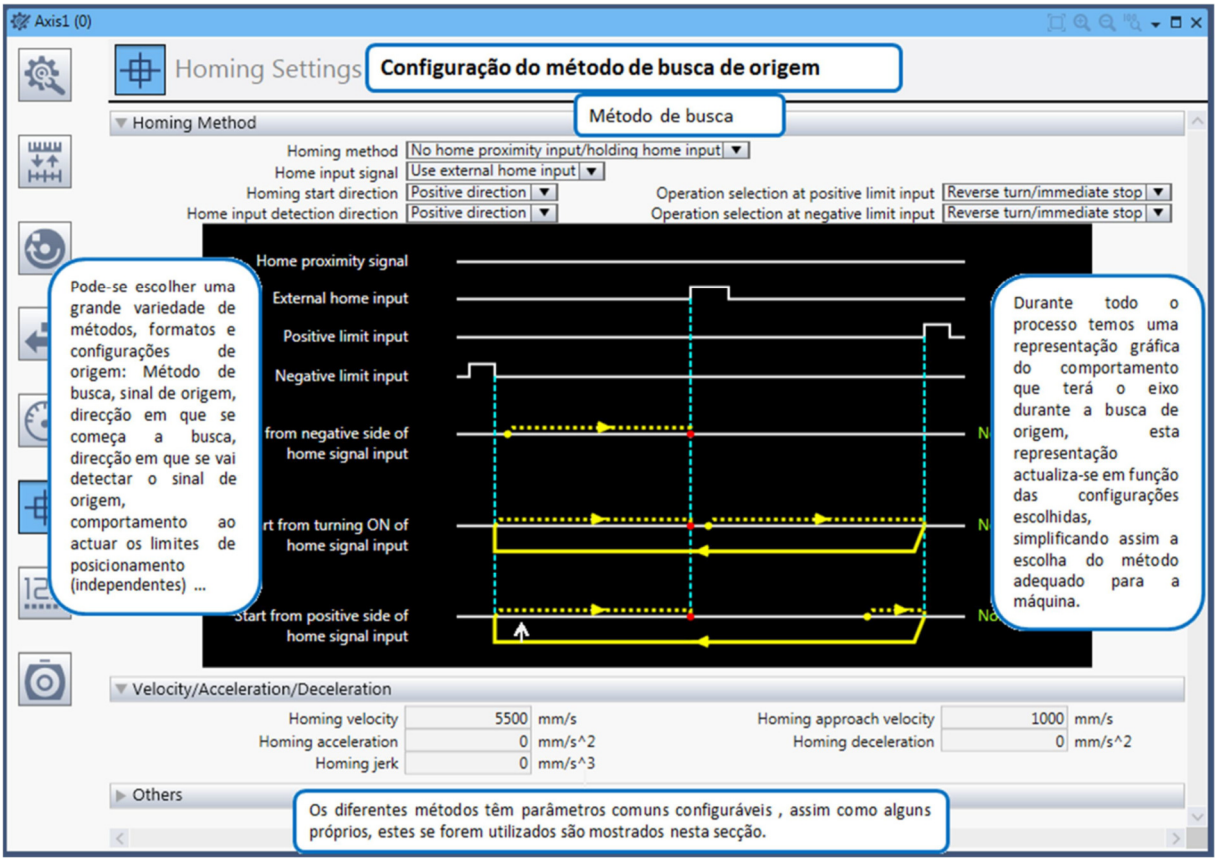



Figure 30. Homing configurations

Since this axis controls a conveyor, the homing method is configured as follows. This means that when the homing instruction is executed, the current axis position is set to zero.

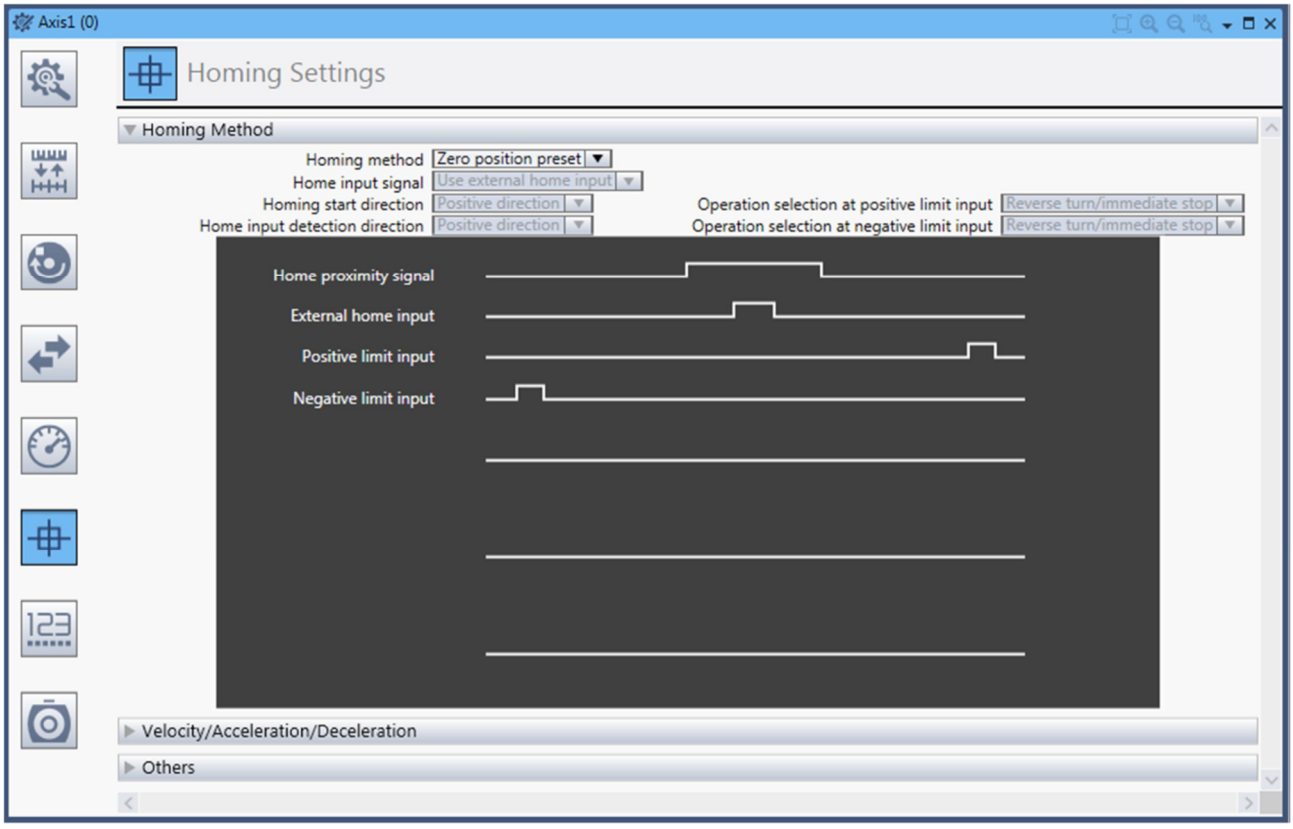
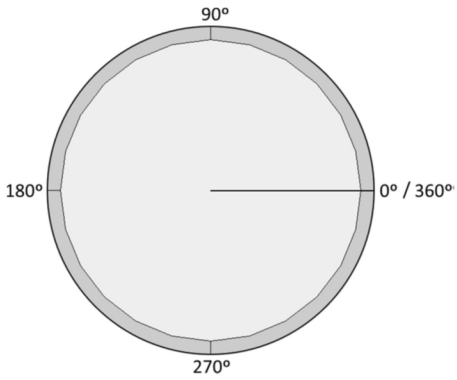



Figure 31. Homing configurations

## 2.5 Unit Conversion Settings

We must configure the units in which we want to work for **axis 2 (disk)**. These are defined in **Unit Conversion Settings**.

Based on the mechanical element driven by the servo motor of axis 2, we have the following:




**Working in degrees [°], we verify that for each motor revolution, the disk moves 360°.**

For the servo drive / servo motor used in the **Sysmac didactic bench**, we have:

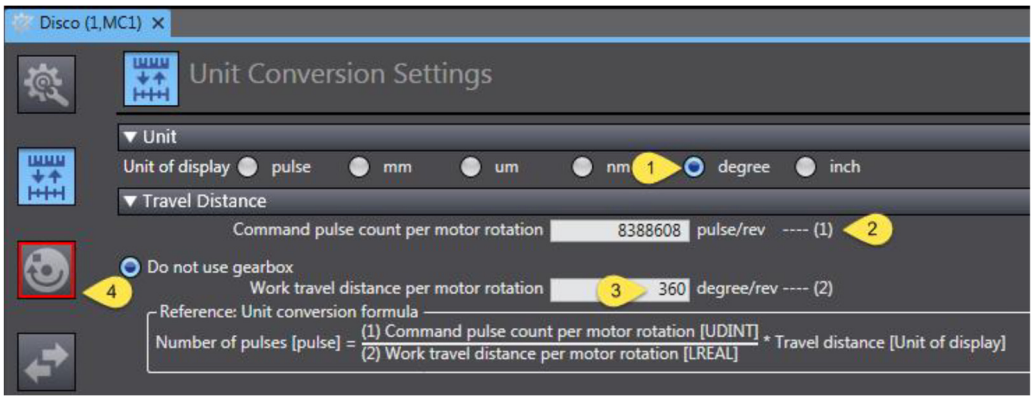
- 23-bit incremental encoder
- 3,000 rpm

**Standard Servo Motors**

Servo motors 1,000 – 2,000 – 3,000 r/min

100 to 3,000 Kw

| Specifications | | | Servo Drive |
|---|---|---|---|
| Voltage | Encoder | Capacity | 1S |
| 230 V | Absolute encoder (23 bit) | 50 W | R88D-1SN01H-ECT |




**(1)** Definition of the working unit, in this case degrees [°]

**(2)** Pulses per motor revolution:

$$2^{23\ \text{bits}} = 8{,}388{,}608\ \text{pulses/revolution}$$

**(3)** Distance traveled per revolution: **360°**

**(4)** After setting these values, the **Operation Settings** icon displays an error.
In the next step, we must configure the **Operation Settings** for axis 2.

## 2.6 Operation Settings

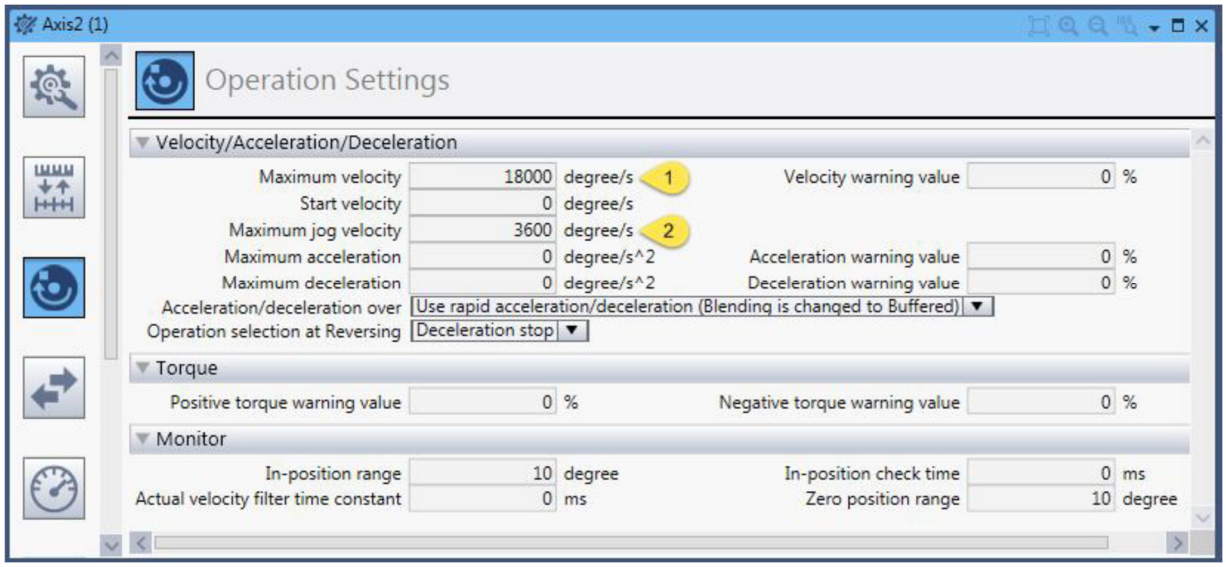



Figure 32. Operation Settings

$$velMax\ [Hz] = velMax\ [rpm]/60\ [s];$$

$$velMax\ [Hz] = 3\,000\ [rpm]/60\ [s] = 50\ [Hz];$$

**(3)** Maximum Speed

**18,000 °/s**

$$velMax\ [^\circ/s] = velMax\ [Hz] \times \Delta\theta\ [^\circ]$$

$$velMax\ [^\circ/s] = 50\ [Hz] \times 360\ [^\circ] = 18,000\ [^\circ/s]$$

**(4)** Maximum Jog Speed

**3,600 °/s**

$$velMaxJog\ [^\circ/s] = 10\ [Hz] \times \Delta\theta\ [^\circ]$$

$$velMaxJog\ [^\circ/s] = 10\ [Hz] \times 360\ [^\circ] = 3,600\ [^\circ/s]$$

## 2.7 Home Settings

In the homing method configuration, we define the behaviour of the axis during the homing operation. Since this axis controls a disk, the homing method is configured as follows.

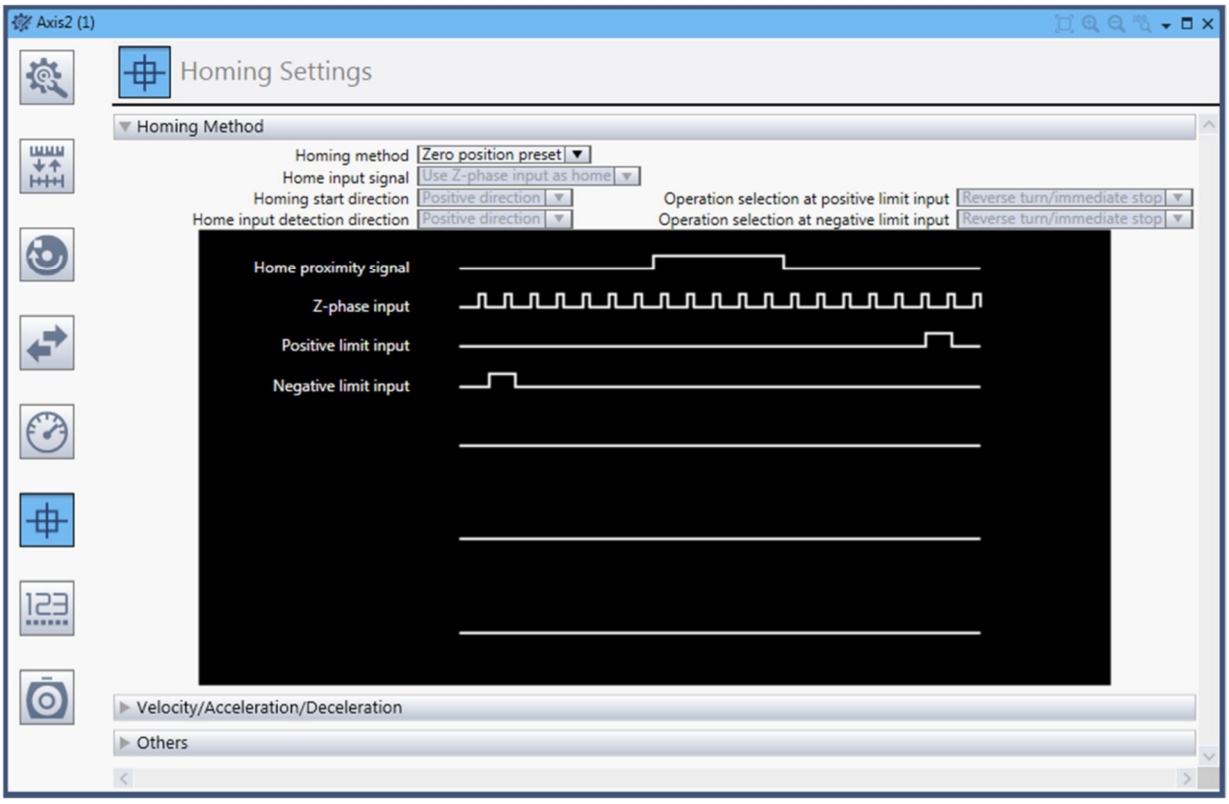



Figure 33. Homing settings – Axis 2

This means that when the homing instruction is executed, the current position of the axis is set to zero.

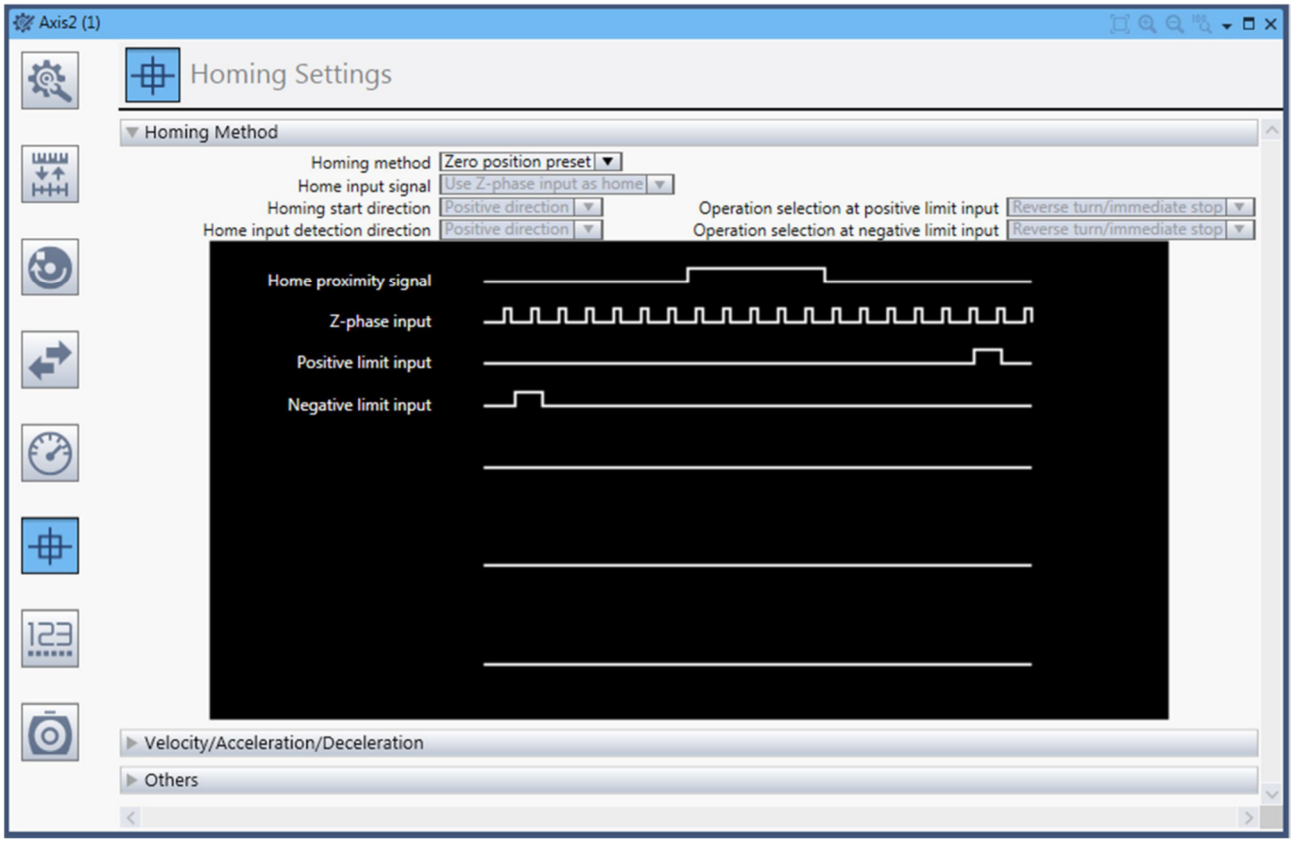



Figure 34. Homing settings – Axis 2

## 2.8 Position Count Settings

The count mode defines how the axis position is tracked. The position count mode must be set for the command positions of each axis. There are two count modes:

**Linear Mode** *(Default mode)*
The count mode is set to linear when an axis always moves within predefined limits that will never be exceeded.

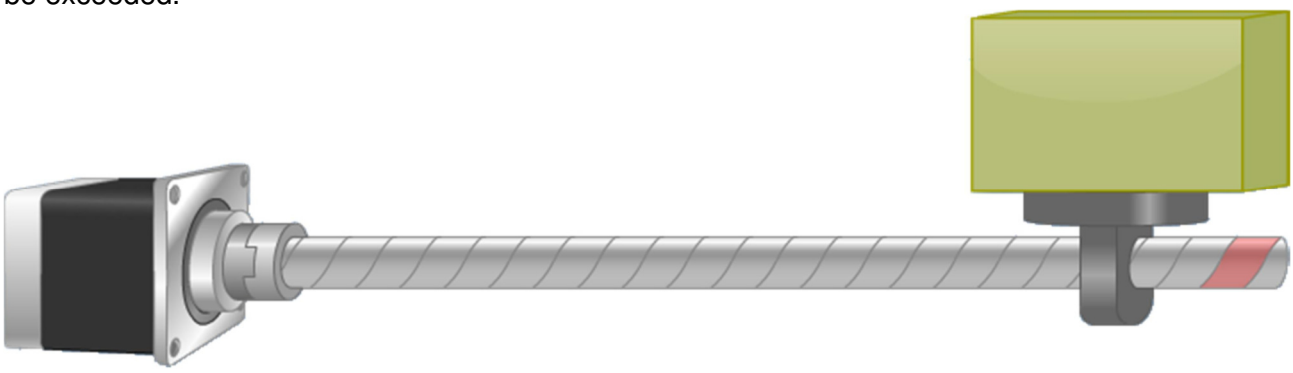


Figure 35. Linear mode

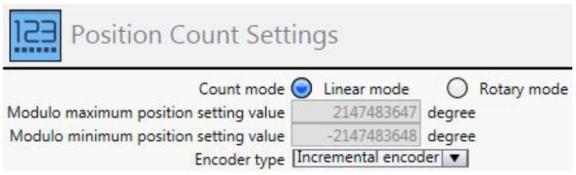



Figure 36. Linear mode example

**Rotary Mode**

The count mode is set to rotary when an axis performs continuous or infinite motion that exceeds the maximum or minimum value of the Position Module.

Figure 37. Rotary mode

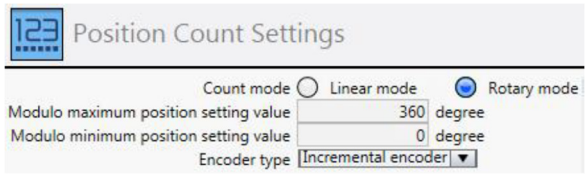



Figure 38. Rotary mode example

For **Axis2 [Disk]**, this means that we will have a new position reference every **90°** of rotation.

| 1st Rotation | | | | | 2nd Rotação | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0° | 90° | 180° | 270° | 360° | 450° | 540° | 630° | 720° | Linear Count Mode |
| 0° | 90° | 180° | 270° | 360° / 0° | 90° | 180° | 270° | 360° / 0° | Rotary Count Mode |

Accordingly, we apply the following configuration to **Axis2 [Disk]**:

**Position Count Settings**

1. **Rotary Mode**
2. **Maximum position:** 360°
3. **Minimum position:** 0°

Then execute the following steps:
**Project > Rebuild Controller > Online > Synchronize > Transfer to Controller**

Run a test using **MC Start Test Run** and verify the differences in the position count mode.

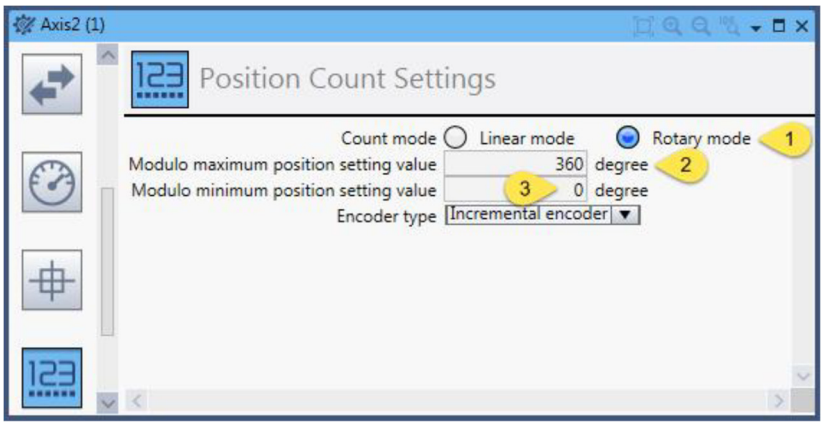



Figure 39. Position Count Settings

# 3 Motion: Start-up – Software

After assigning the hardware (servo drive) to the axis via **Axis Basic Settings** and validating correct operation via **MC Test Run**, we continue configuration via **Motion Controller Setup > Axis Settings**.

## 3.1 Start-up Procedure

To move any axis, only **3 commands** are required: MC_Power, MC_Home, and MC_Move, all linked to one of the previously created and configured axes.

*Global Variables view in Programming > Data*

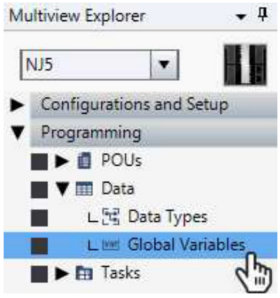
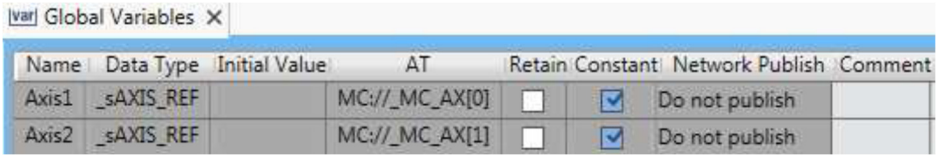



Figure 40. Global variables in *Programming* > *Data*

When we access global variables via **Programming > Data > Global Variables**, we see that two global variables of type _sAXIS_REF were automatically created, one for each axis.




So far, we have been working with local variables. To view them:

1. Go to **View**
2. Open the **Variable Manager**

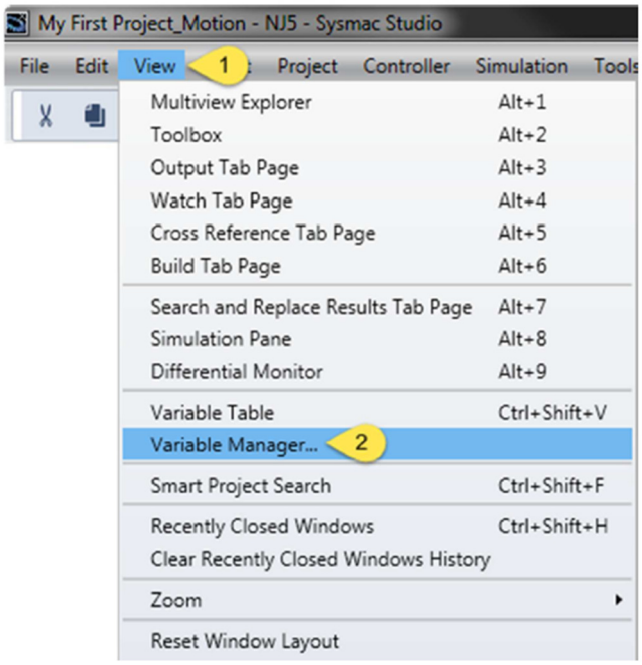



Figure 41. Working with local variables

*Switching between Global and Program0 Variables*

To convert a local variable to global, right-click the variable and select **Move Variable to Global**.

**Scope of Global Variables:**
They are accessible from any POU (Program, FB, or FUN). To avoid unintended access, a variable accessed by a POU should be declared as **external**.

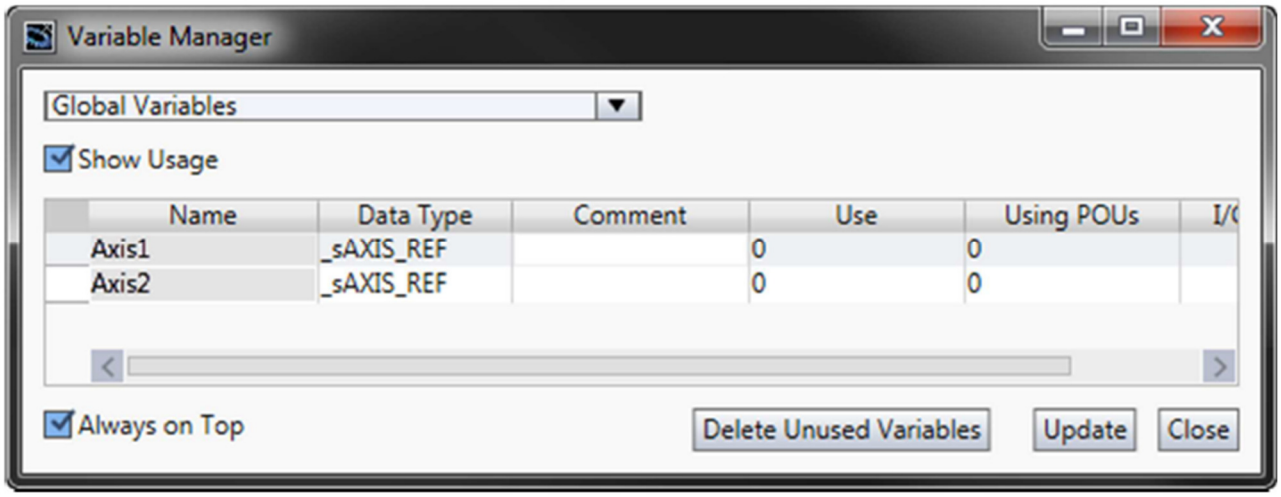



Figure 42. Global variables

To view the **local variables** associated with **Program0**:

1. Switch from **Global Variables** to **Program0**
2. If you want to change a variable from local to global, right-click the variable and select **Move Variable to Global**

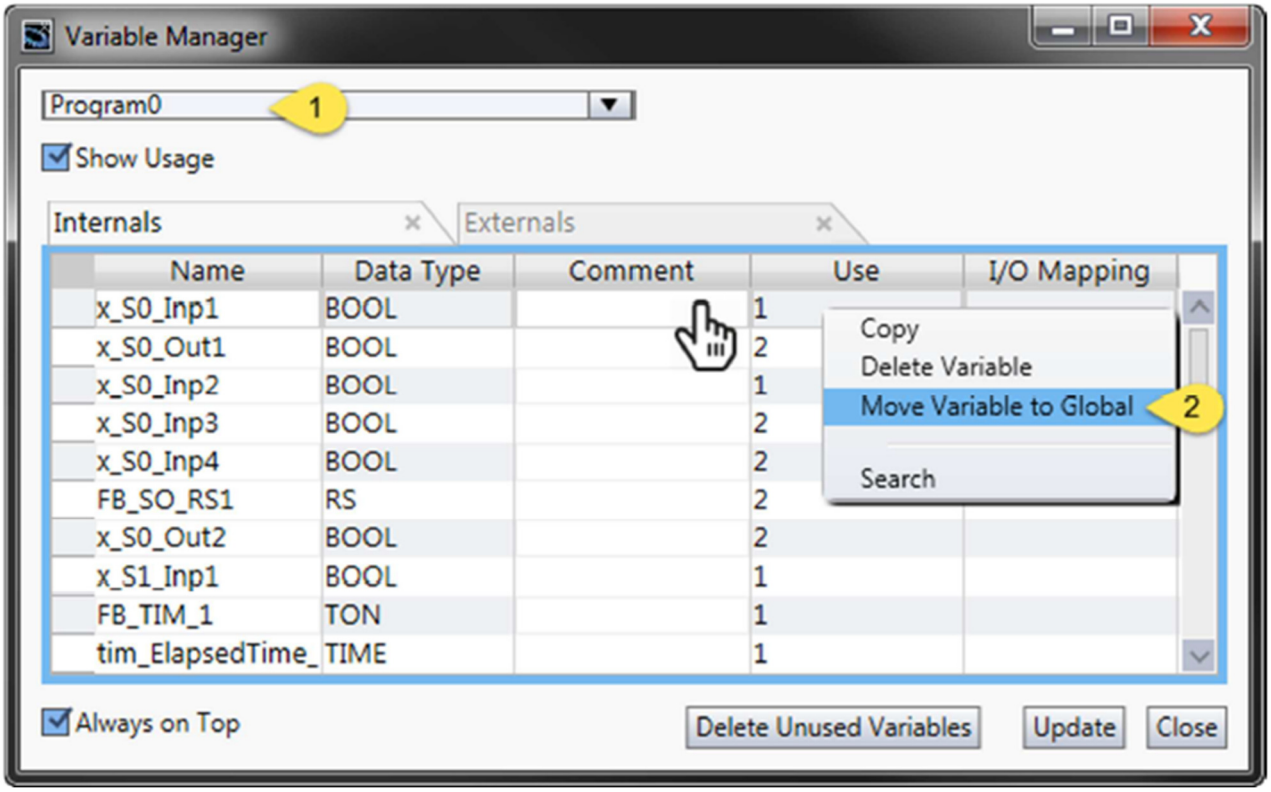



Figure 43. Global Variables to Program0

**Scope of Global Variables**

Global variables are accessible from all POUs (Programs, Function Blocks, and Functions).
To prevent unnecessary access and avoid potential issues, access to a POU should be defined as an **external variable**.

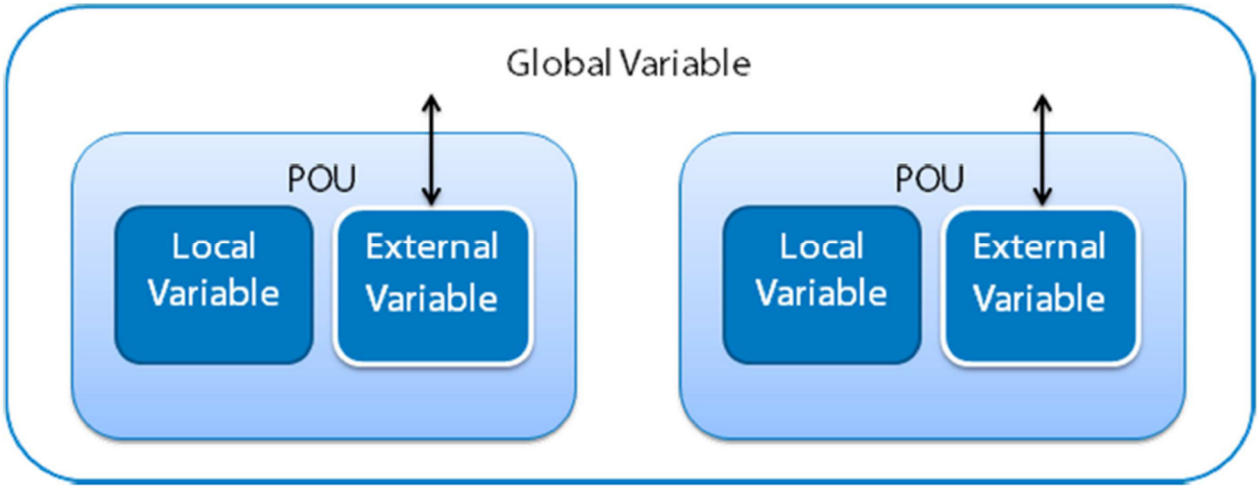



Figure 44. Global variables accessible from all POUs

## 3.2 _sAXIS_REF Structure

Each created axis is of type _sAXIS_REF. Using the **Watch Window**, we can inspect the structure of these axes.

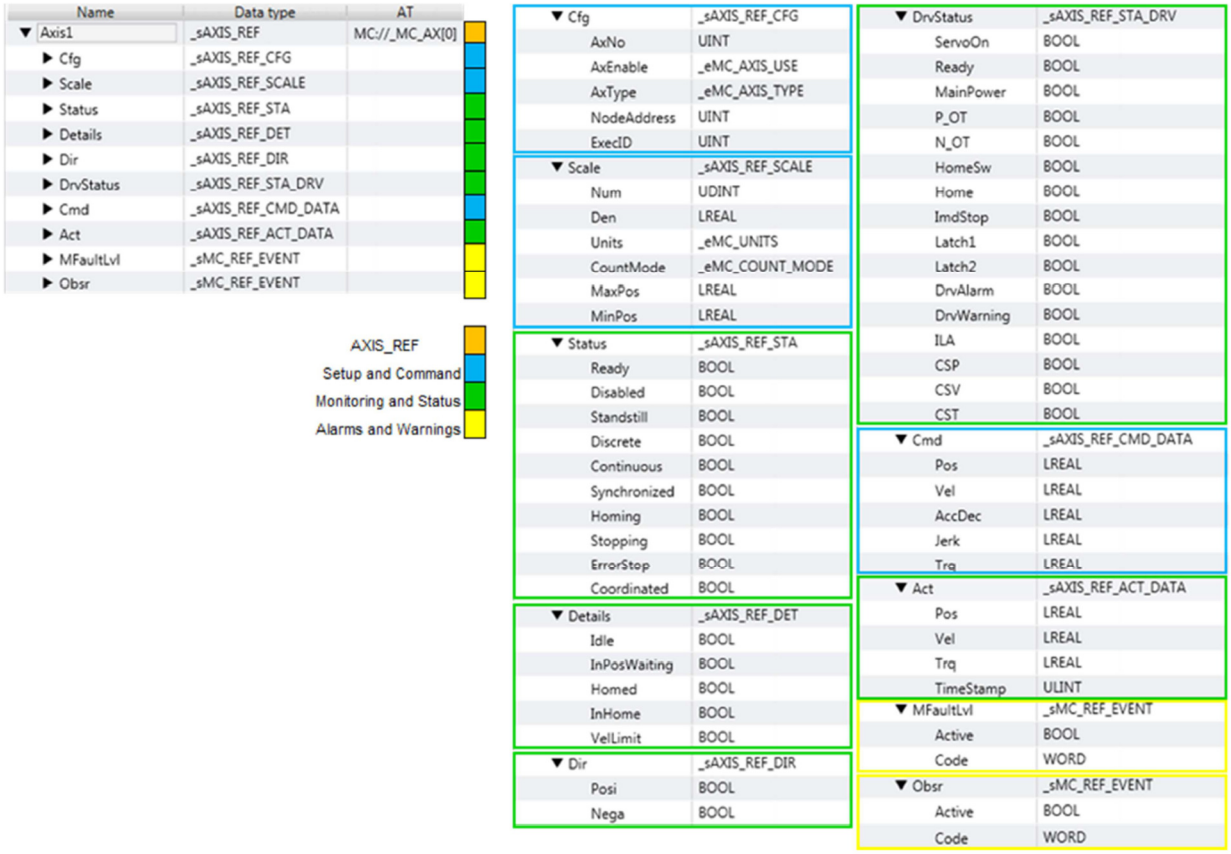



Figure 45. _sAXIS_REF Structure

## 3.3 Motion Basics

In **Programs**, add a new program in **Ladder**, naming it Program_Motion. Call the initial section: Section0_MotionBasics.

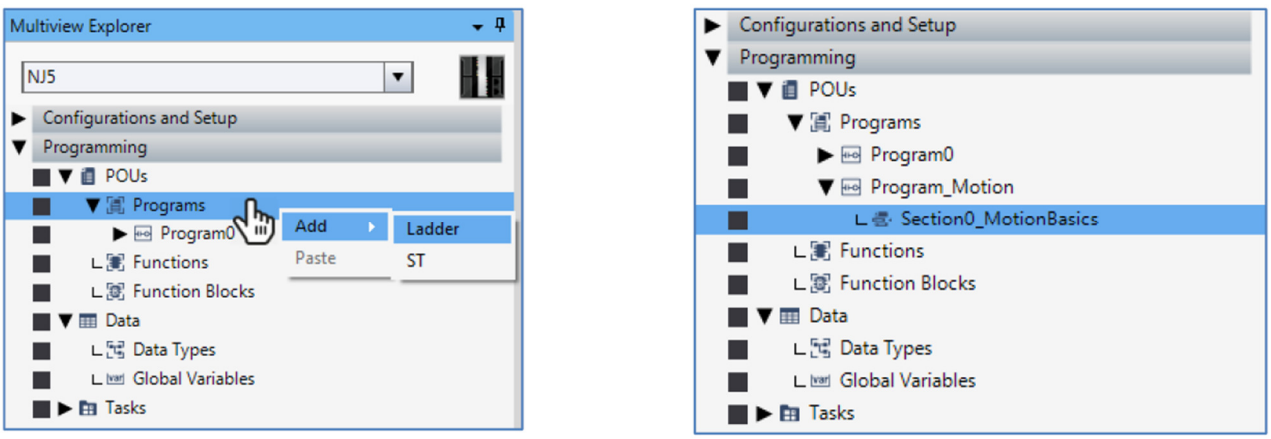



Figure 46. Section0_MotionBasics

## 3.4 Task

New programs must be assigned to a **Task**, otherwise the controller won't execute them.

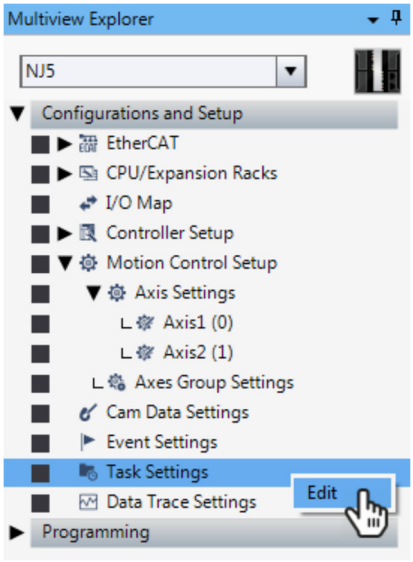



Figure 47. *Accessing Task Settings and creating assignments*

The **Task Settings** window opens. Here, you can create **three types of tasks**:

- **Primary task**
  - Priority level: **4**
- **Additional periodic tasks** can be created and assigned a lower priority
  - Priority levels: **16, 17, and 18**
- **Event tasks** can be configured to execute when a specific condition is met
  - Priority levels: **8 and 48**

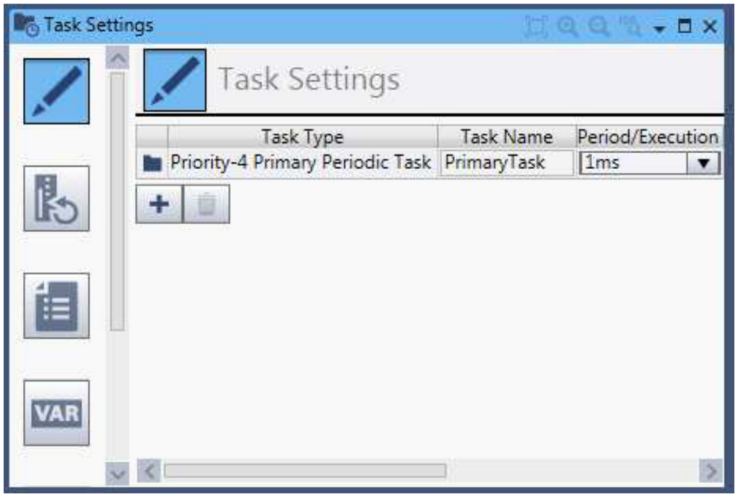



Figure 48. *Tasks and priorities*

For this example, we will use the existing **primary task**.

1. Select the **Program Assignment Settings** tab
2. Click **+** to add a program to the primary task
3. Click on the **drop-down list**
4. Select the previously created program, Program_Motion

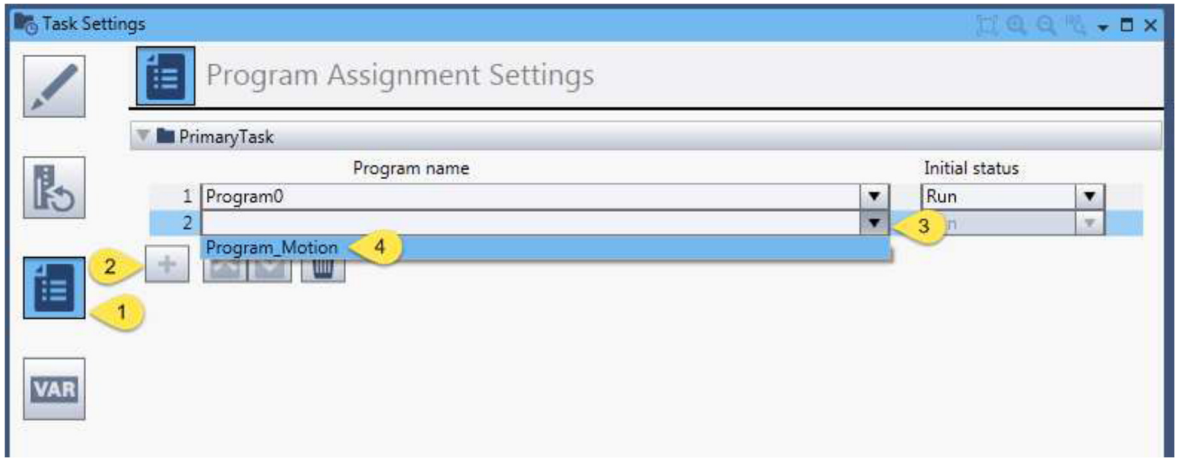
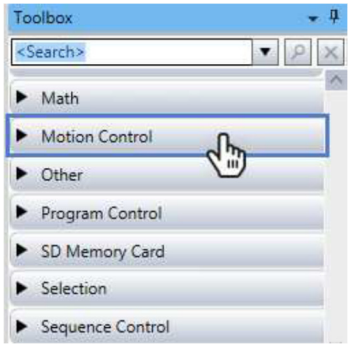



Figure 49. *Program Assignment Settings tab*

## 3.5 MC_Power

The first step in using an axis is to **power it on**.




To do this, use the MC_Power instruction, which can be found in the **Toolbox**, under the **Motion Control** tab.

Whenever the prefix MC_ is used in the system, it refers to **Motion Control**.

**Objective:** Power on the created axes using separate digital inputs.

| Name | Data Type |
|---|---|
| x_MS0_Axis1_PowerOn | BOOL |
| FB_MC_Power_Axis1 | MC_Power |
| x_MS0_Axis2_PowerOn | BOOL |
| FB_MC_Power_Axis2 | MC_Power |

**Naming convention:**
The following code is obtained.

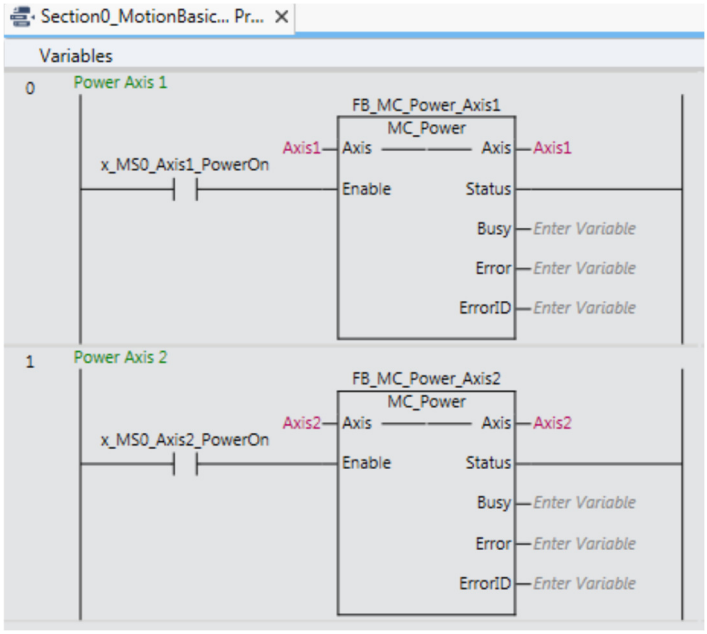



Figure 50. *Code for MC_Power*

Once the programming is complete, perform the following steps:
**Project > Check All Programs > Rebuild Controller > Online > Synchronize > Transfer To Controller**

After synchronizing the project with the controller, validate and verify the code operation.

1. Input x_MS0_Axis1_PowerOn = TRUE
   Powers on **Axis1 (Conveyor)**
2. Input x_MS0_Axis2_PowerOn
   Powers on **Axis2 (Disk)**

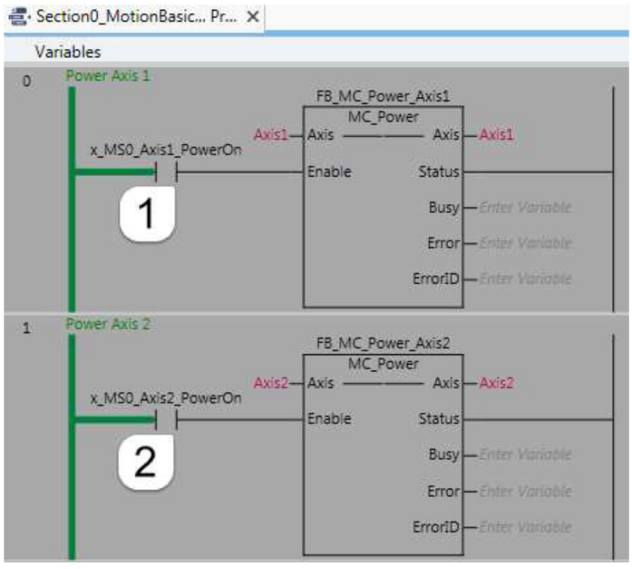
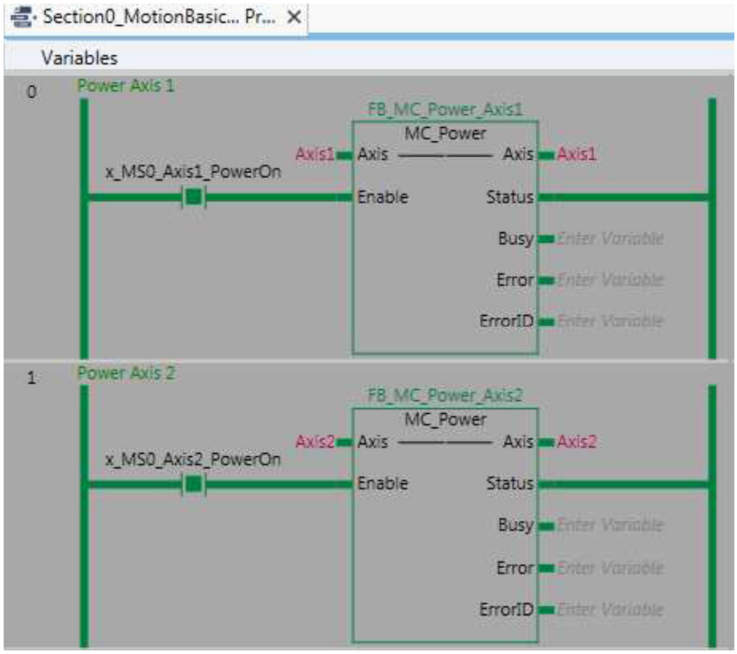



Figure 51. *Code for MC_Power - Test*

At this point, it is confirmed that both the conveyor and the disk are powered.




Since MC_Power has an **Enable-type** input, it means that it remains active **only while its input is TRUE**.

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Enable | Enable | BOOL | TRUE or FALSE | --- | The device is ready for operation when *Enable* is TRUE, and not ready when it is FALSE. |

## 3.6 MC_Home

Depending on the homing method defined for each axis, the MC_Home instruction will execute the homing operation.

In this case, when the homing instruction is executed, the current position is set to zero.

Objective: Apply the homing command to each of the axes

Naming convention:

| Name | Data Type |
|---|---|
| FB_MC_Power_Axis2 | MC_Power |
| x_MS0_Axis1_Homing | BOOL |
| FB_MC_Home_Axis1 | MC_Home |
| x_MS0_Axis2_Homing | BOOL |
| FB_MC_Home_Axis2 | MC_Home |

The following code is obtained.

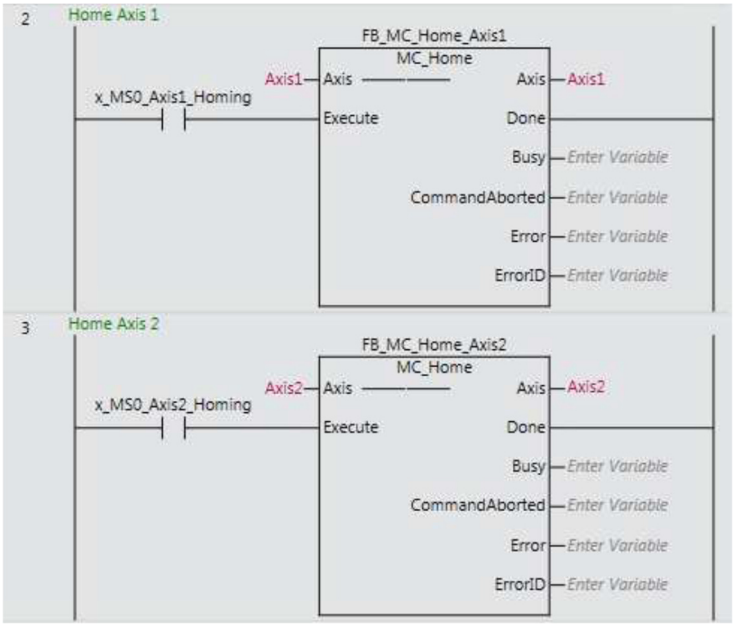



Figure 52. *Code for MC_Home*

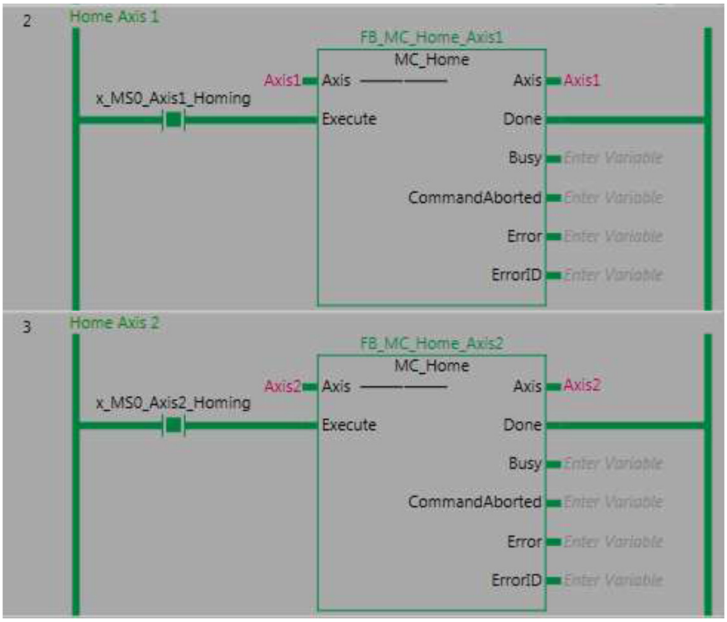
Once programming is complete, perform:
**Project > Check All Programs > Rebuild Controller > Online > Synchronize > Transfer To Controller**

After synchronizing the project with the controller, validate and test the code.

By executing MC_Power for each axis and then triggering the inputs:

- X_MS0_Axis1_Homing
- X_MS0_Axis2_Homing

You can verify via the **Watch Window** that the homing instruction is executed.



The position values of **Axis1 (Conveyor)** and **Axis2 (Disk)**, if different from zero, are then reset to **zero**.

Watch1

| Name | Online value |
|---|---|
| Axis1.DrvStatus.ServoOn | True |
| Axis2.DrvStatus.ServoOn | True |
| Axis1.Act.Pos | 0 |
| Axis2.Act.Pos | 0 |

Since MC_Home has an **Execute-type** input, this means the instruction is executed when the signal transitions to **TRUE**.

| Name | Meaning | Data type | Valid range | Default | Description |
|---|---|---|---|---|---|
| Execute | Execute | BOOL | TRUE or FALSE | FALSE | The instruction is executed when *Execute* changes to TRUE. |

## 3.7 MC_MoveVelocity

After executing MC_Power and MC_Home (if necessary), the next step is to use MC_Move to put the axis in motion.

**Objective:** Apply the MC_MoveVelocity instruction to each of the axes

**Naming convention:**

| Name | Data Type |
|---|---|
| x_MS0_Axis1_MoveVel | BOOL |
| FB_MC_MoveVel_Axis1 | MC_MoveVelocity |
| lr_Axis1_MoveVel | LREAL |
| lr_Axis1_Acc | LREAL |
| lr_Axis1_Dcc | LREAL |
| FB_MC_Stop_Axis1 | MC_Stop |
| x_MS0_Axis2_MoveVel | BOOL |
| FB_MC_MoveVel_Axis2 | MC_MoveVelocity |
| lr_Axis2_MoveVel | LREAL |
| lr_Axis2_Acc | LREAL |
| lr_Axis2_Dcc | LREAL |
| FB_MC_Stop_Axis2 | MC_Stop |

The following code is obtained:

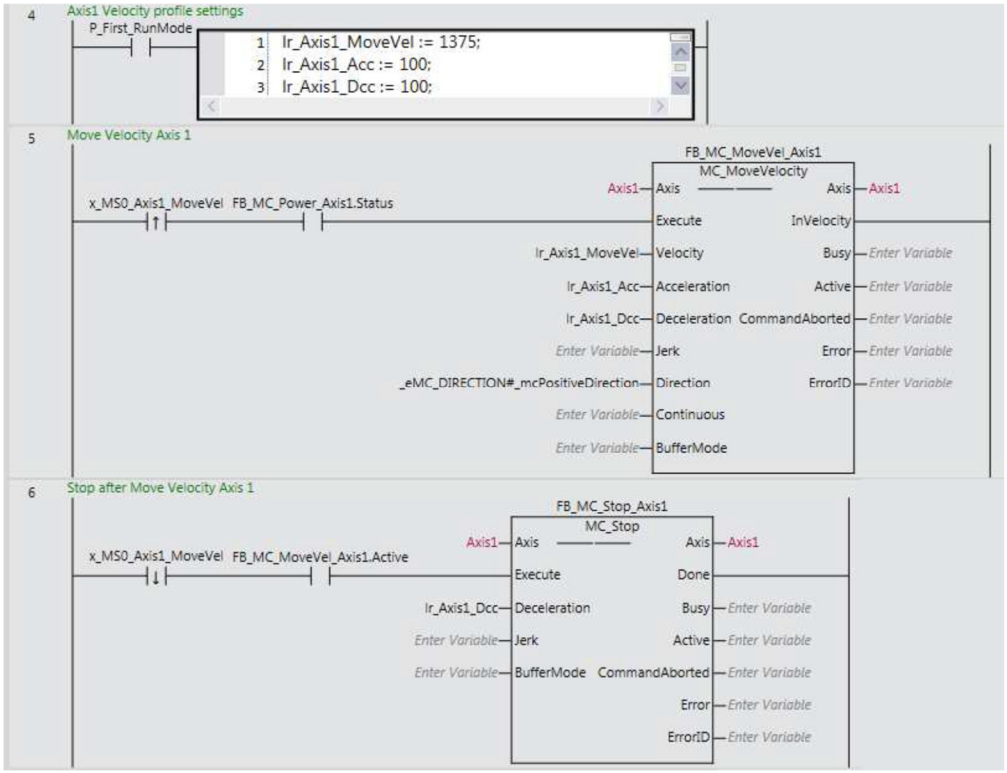



Figure 53. *MC_MoveVelocity code applied to Axis1 (Conveyor)*

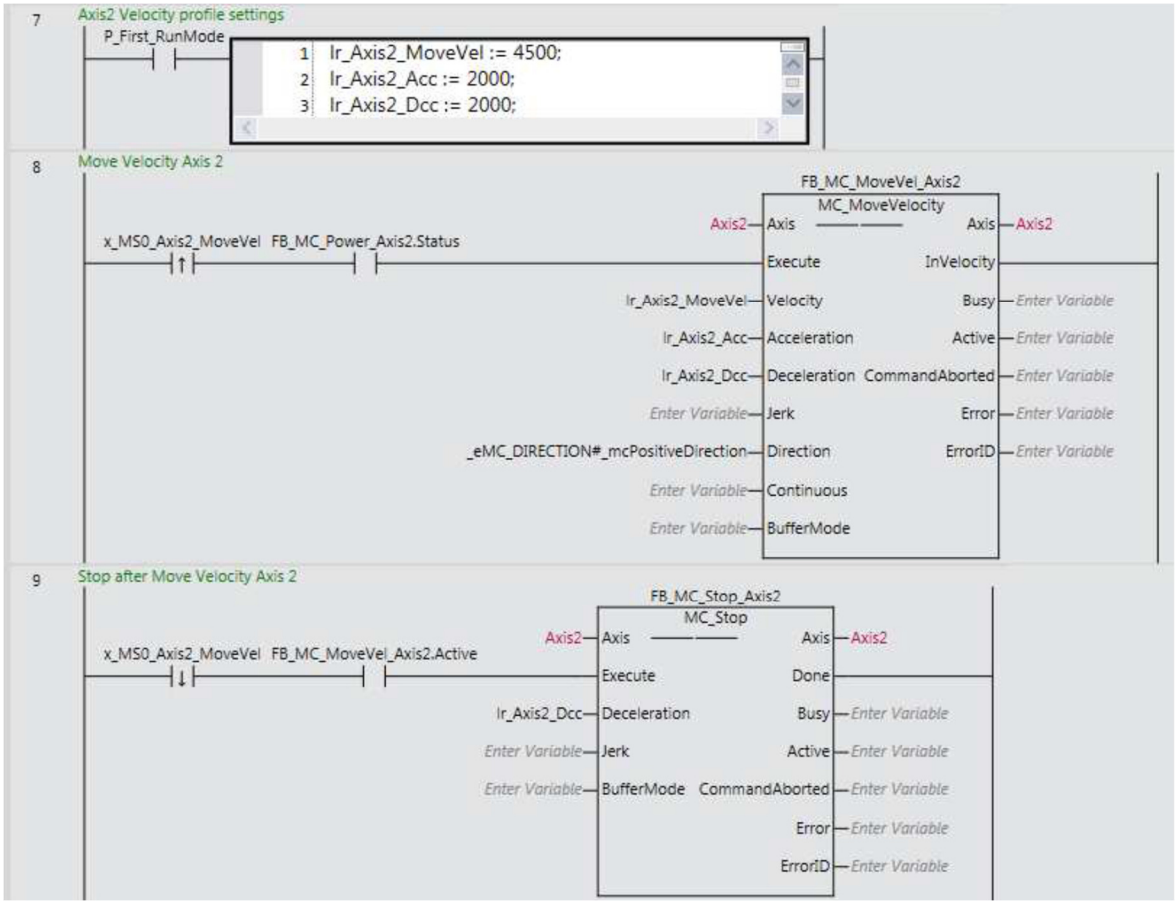
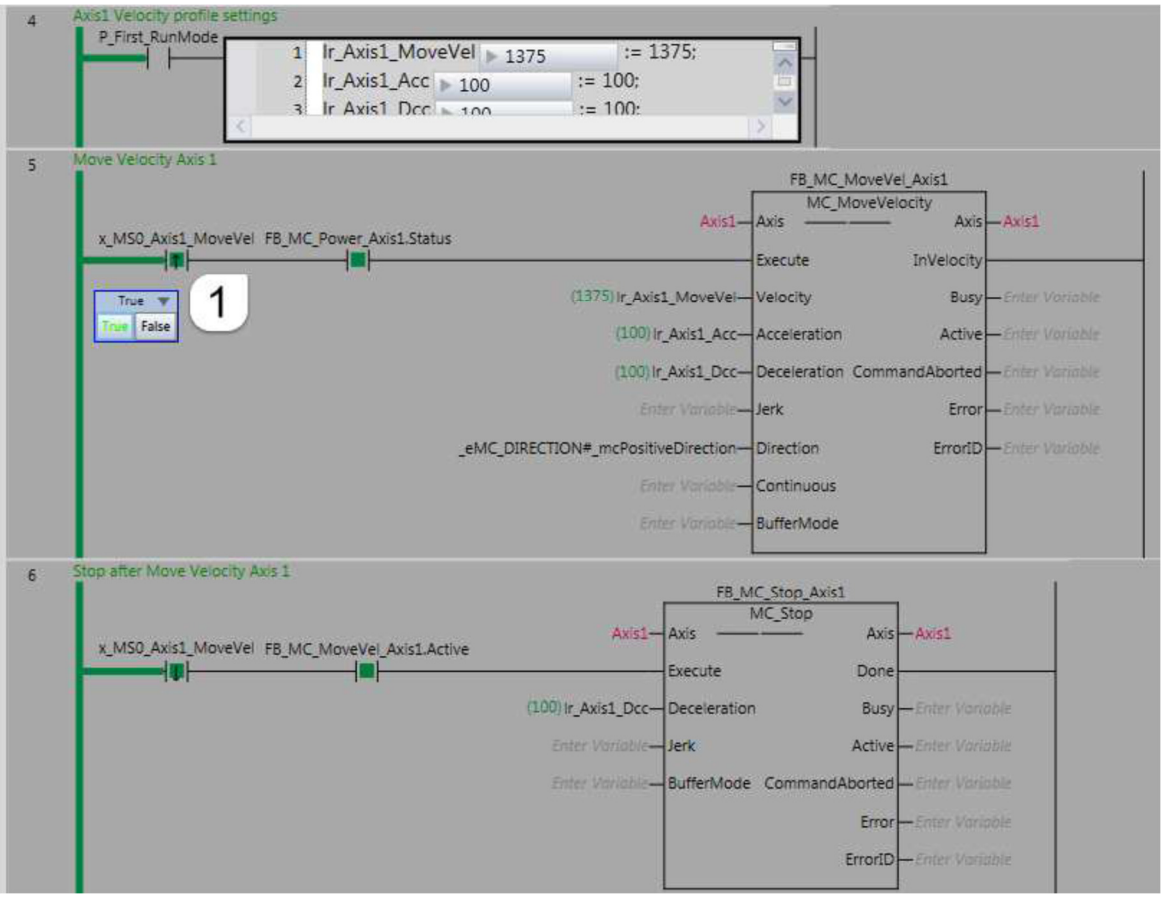



Figure 54. *MC_MoveVelocity code applied to Axis2 (Disk)*

Once the programming is complete, perform the following steps:
**Project > Check All Programs > Rebuild Controller > Online > Synchronize > Transfer To Controller**

After synchronizing the project with the controller, validate and verify the code operation.

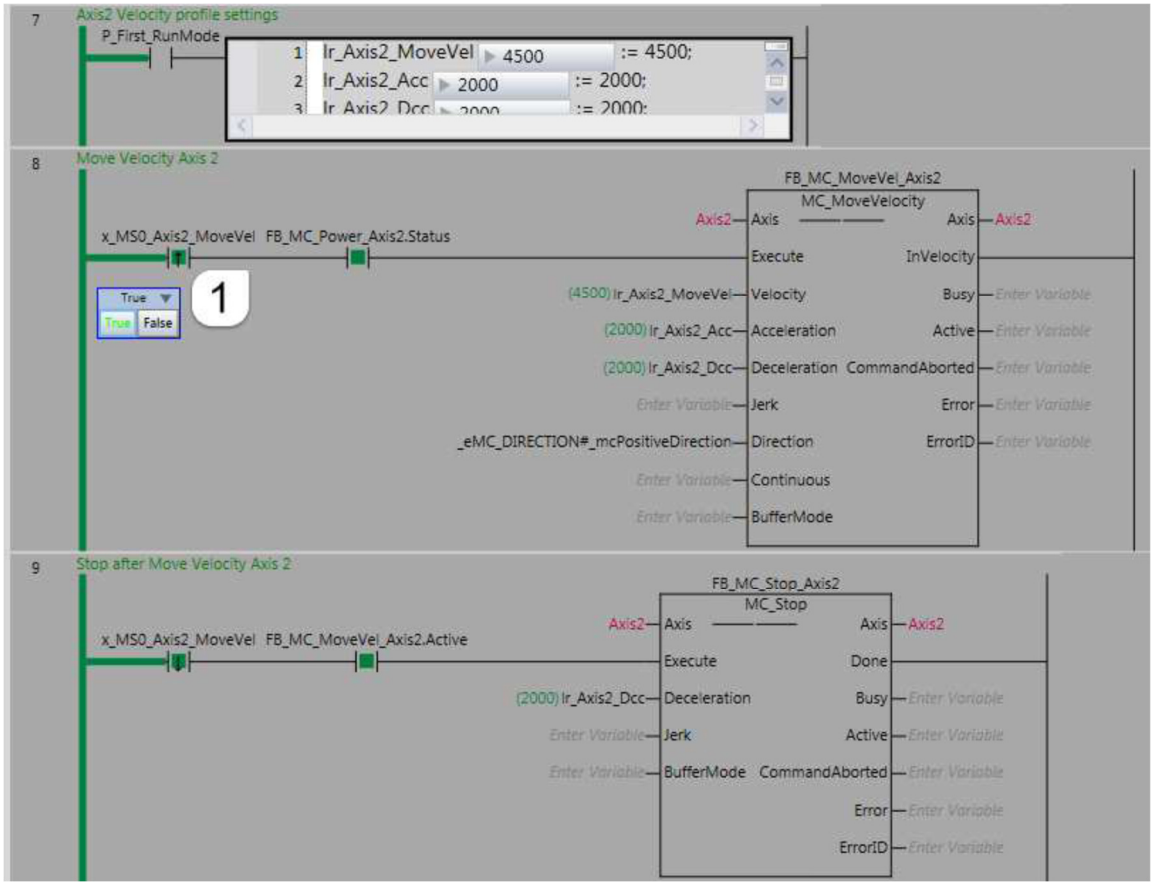
**Axis1 (Conveyor)**




(1) When x_MS0_Axis1_MoveVel = TRUE is set,
**Axis 1 moves in the positive direction at the defined speed.**

It continues moving until x_MS0_Axis1_MoveVel = FALSE.

**Axis2 (Disk)**




(1) When x_MS0_Axis2_MoveVel = TRUE is set,
**Axis 2 moves in the positive direction at the defined speed.**
It continues moving until x_MS0_Axis2_MoveVel = FALSE.

# 4 Motion: Exercise - Software

The goal is to create a program that controls a **conveyor belt** (**Axis1 - Conveyor**) and a **label applicator disk** (**Axis2 - Disk**).

## 4.1 Exercise

To move any axis, only **3 commands** are required: MC_Power, MC_Home, and MC_Move, all linked to one of the previously created and configured axes.

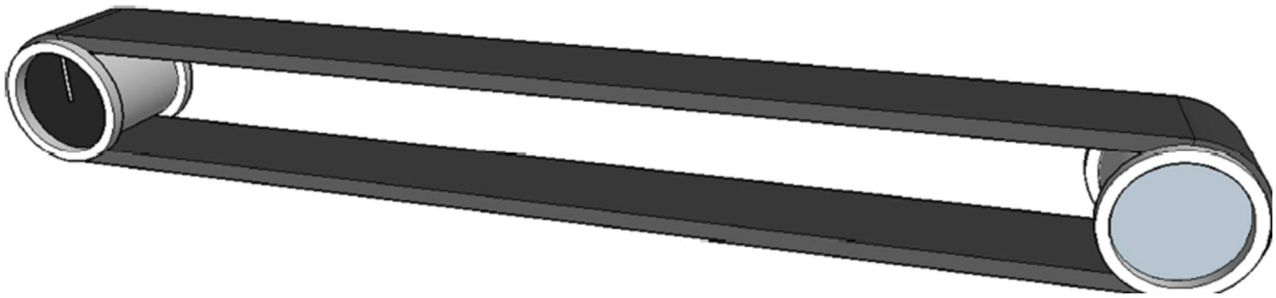


Figure 55. *Axis1 (Conveyor)*

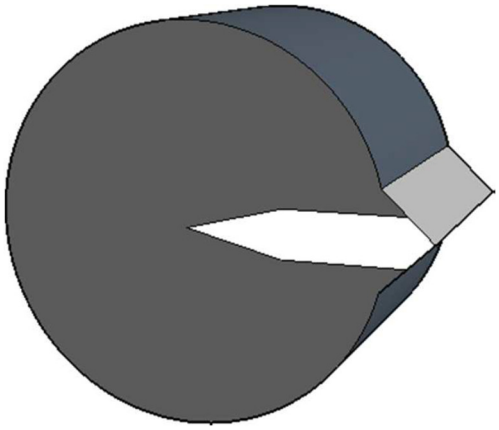


Figure 56. *Axis 2 (Disk)*

Components are transported along the conveyor, which has **two sensors** installed along its path.

If a component **triggers sensor 1** and, after **5 seconds**, does **not trigger sensor 2**, this means the component is **stuck at the entry zone**.
In this case, the system must **stop**.

The conveyor runs at a speed defined in the program code, and when the **detector** located at the **labeling station** is triggered, the **conveyor must stop** until the label is placed by the **disk**.

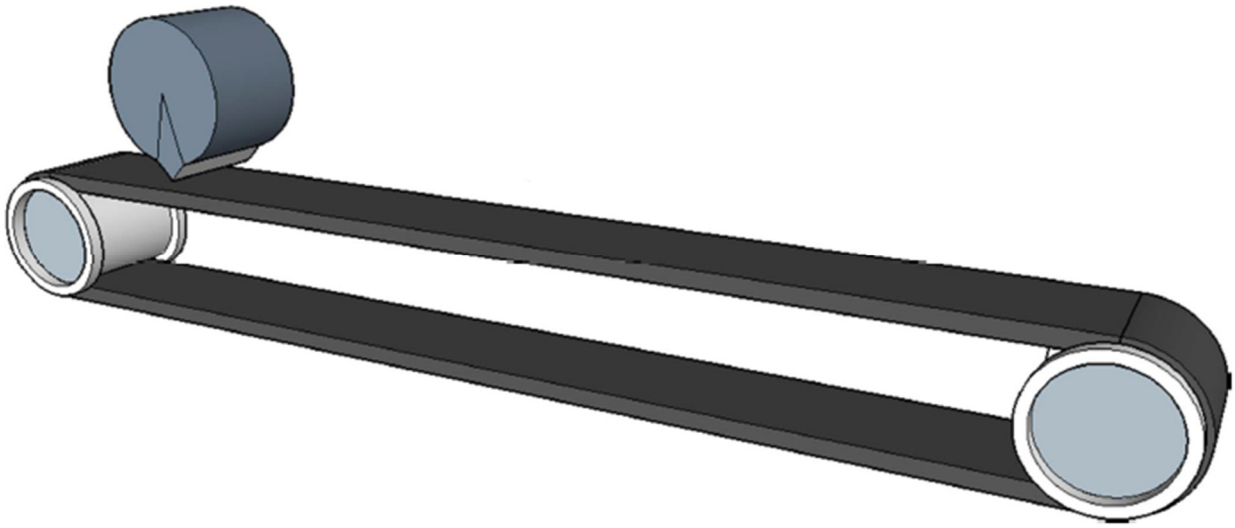
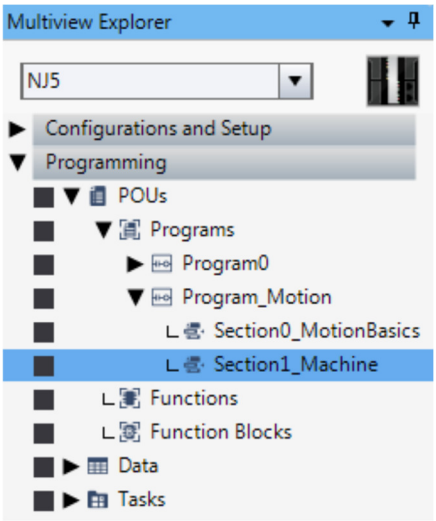


Figure 57. *Exercise*

## 4.2 Exercise - Solutions

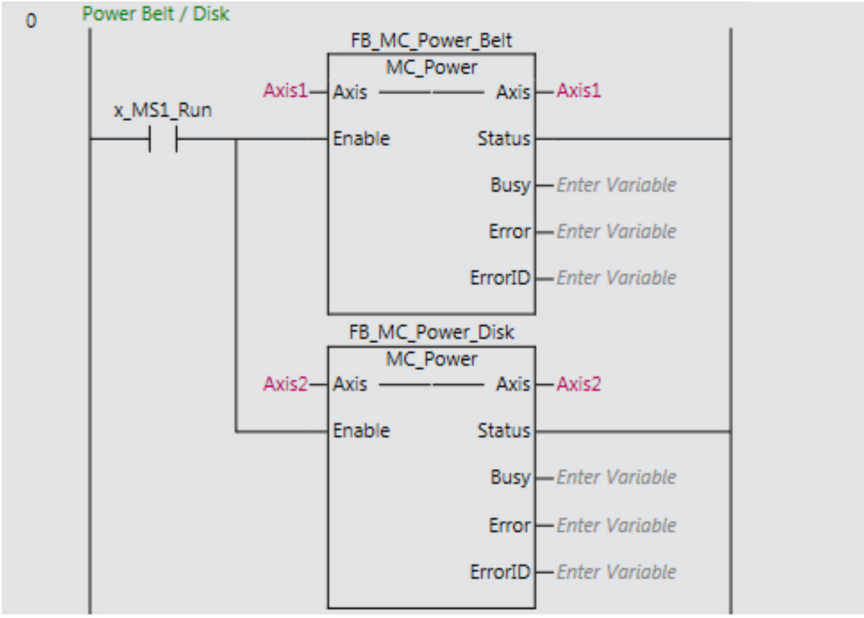
In **Programming > Programs > Program_Motion**, create a **new section** named: **Section1_Machine**




**Rung 0**

Activation of the servos based on the input **x_MS1_RUN**.

| Name | Data Type |
|---|---|
| x_MS1_Run | BOOL |
| FB_MC_Power_Belt | MC_Power |
| FB_MC_Power_Disk | MC_Power |

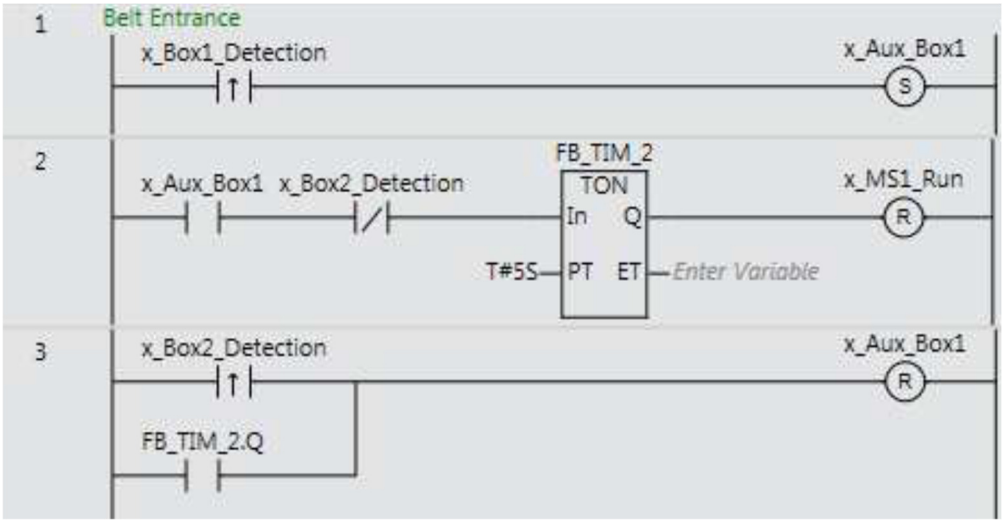



**Rungs 1 / 2 / 3**

**X_Box1_Detection** and **X_Box2_Detection** simulate the passage of components through **Sensor 1** and **Sensor 2**.

If **Sensor 1** is triggered and, within a maximum of **5 seconds**, **Sensor 2** is not triggered, the system's **run command is reset** (the system stops).

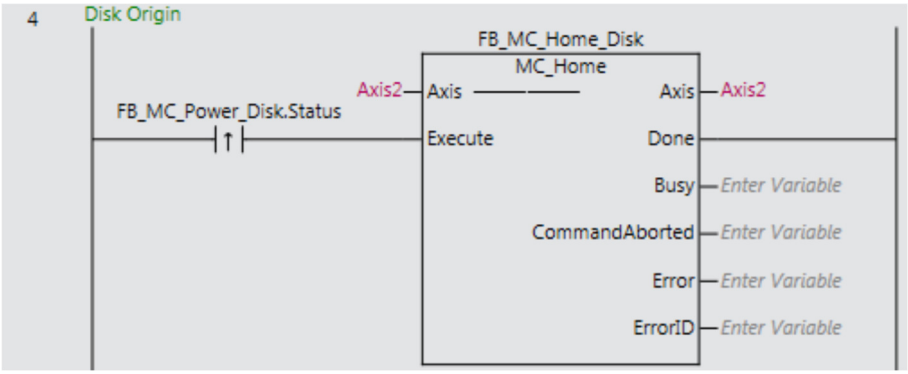
| Name | Data Type |
|---|---|
| x_Box1_Detection | BOOL |
| x_Aux_Box1 | BOOL |
| x_Box2_Detection | BOOL |
| FB_TIM_2 | TON |




**Rung 4**

Once **Axis2 (Disk)** is activated, the **homing procedure** is performed for the axis.

| Name | Data Type |
|---|---|
| FB_MC_Home_Disk | MC_Home |

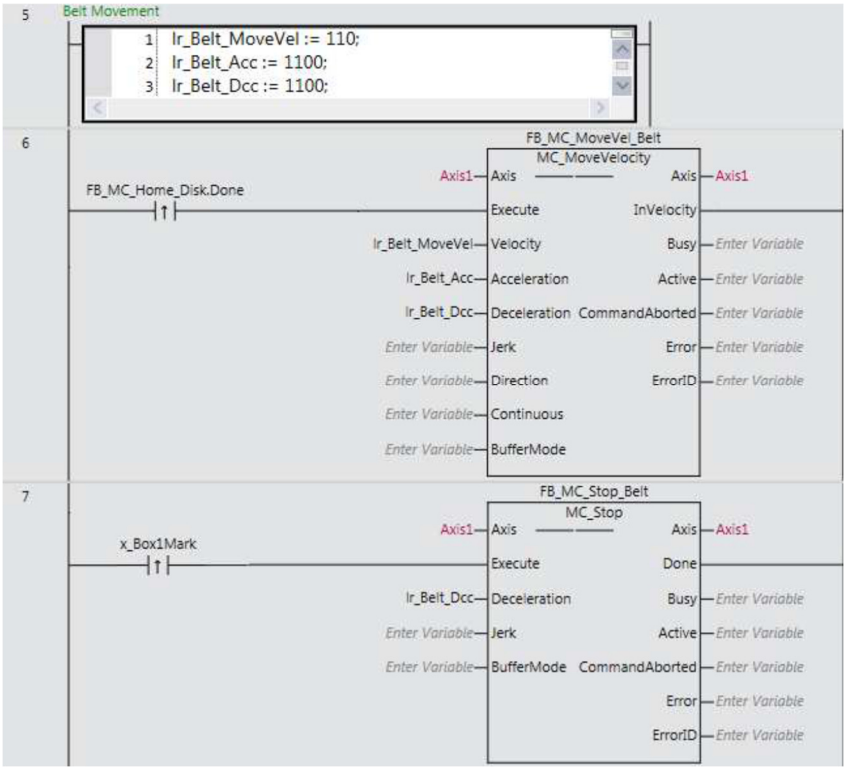



**Rungs 5 / 6 / 7**

After the homing of **Axis2** is completed, the **conveyor** starts moving until the **detector** located at the **labeling station** is triggered. This signal is represented by **x_Box1Mark**.

**Note:**
At the end, you must add **3 additional conditions** to **Rungs 6 and 7**.

| Name | Data Type |
|---|---|
| FB_MC_MoveVel_Belt | MC_MoveVelocity |
| lr_Belt_MoveVel | LREAL |
| lr_Belt_Acc | LREAL |
| lr_Belt_Dcc | LREAL |
| x_Box1Mark | BOOL |
| FB_MC_Stop_Belt | MC_Stop |

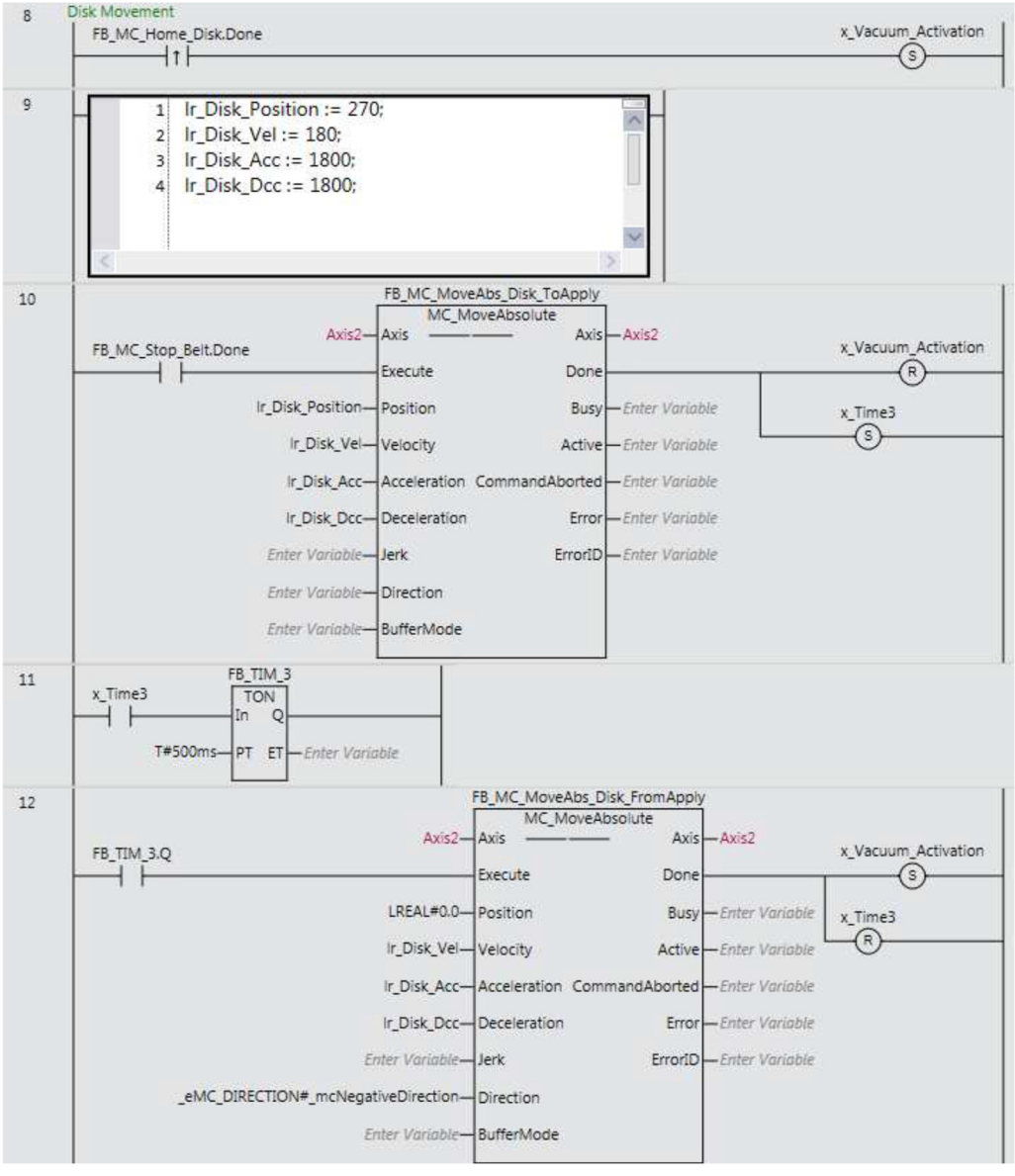


**Rungs 8 / 9 / 10 / 11 / 12**

When the conveyor receives a **stop command** because a component is present in the **labeling zone**, the **disk** moves from its **resting position** to the **labeling position**.

It then **waits for 3 seconds** to apply the label, and **returns to the home position**.

| Name | Data Type |
|---|---|
| x_Vacuum_Activation | BOOL |
| FB_MC_MoveAbs_Disk_ToApply | MC_MoveAbsolute |
| lr_Disk_Position | LREAL |
| lr_Disk_Vel | LREAL |
| lr_Disk_Acc | LREAL |
| lr_Disk_Dcc | LREAL |
| FB_TIM_3 | TON |
| x_Time3 | BOOL |
| FB_MC_MoveAbs_Disk_FromApply | MC_MoveAbsolute |



## 4.3 Extra Exercise - Solutions

**Complete Rungs 6 / 7**

- Add **point ○1** to **Rung 6**
- Add **points ○2 and ○3** to **Rung 7**

This ensures that the **conveyor resumes movement after the label is applied**.