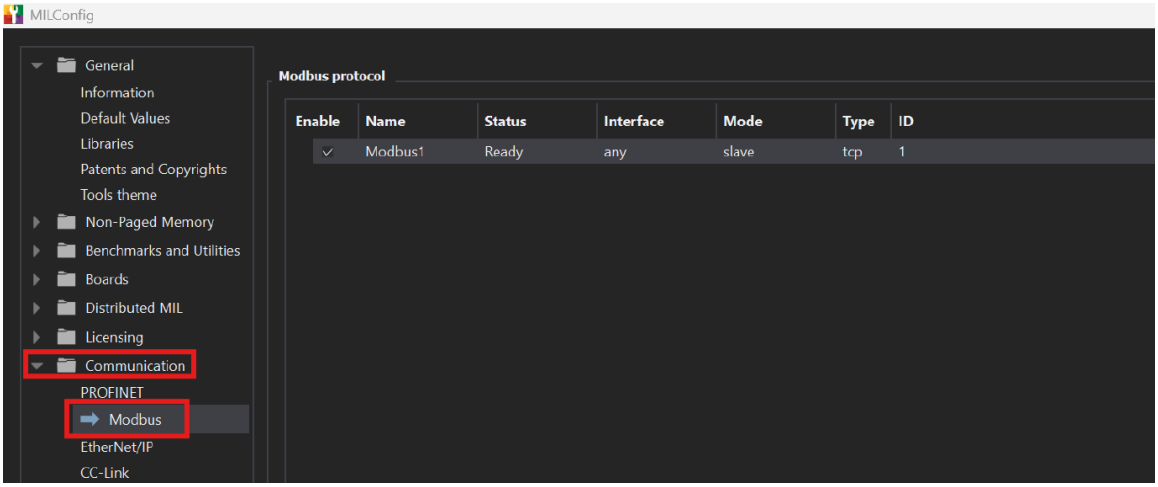
How to Integrate a Zebra EV7 Vision System using a 3S80 Camera to a UR Robot

This document provides step-by-step instructions for integrating a Zebra EV7 Vision System, a Universal Robots (UR) robot, and a Zebra 3S80 camera using MODBUS communication. It outlines how to configure the network, set vision parameters, transfer coordinate data, and establish trigger mechanisms between devices. The goal is to enable automated part detection and robotic interaction based on vision input, typically for industrial automation tasks such as inspection, sorting, or pick-and-place operations.

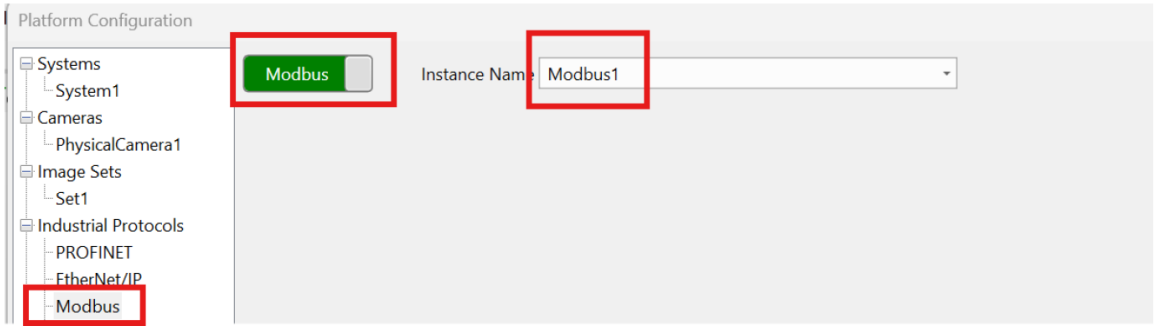
Vision Parameters

1. Set up MODBUS in MilConfig by clicking on Communication > Modbus > Add.

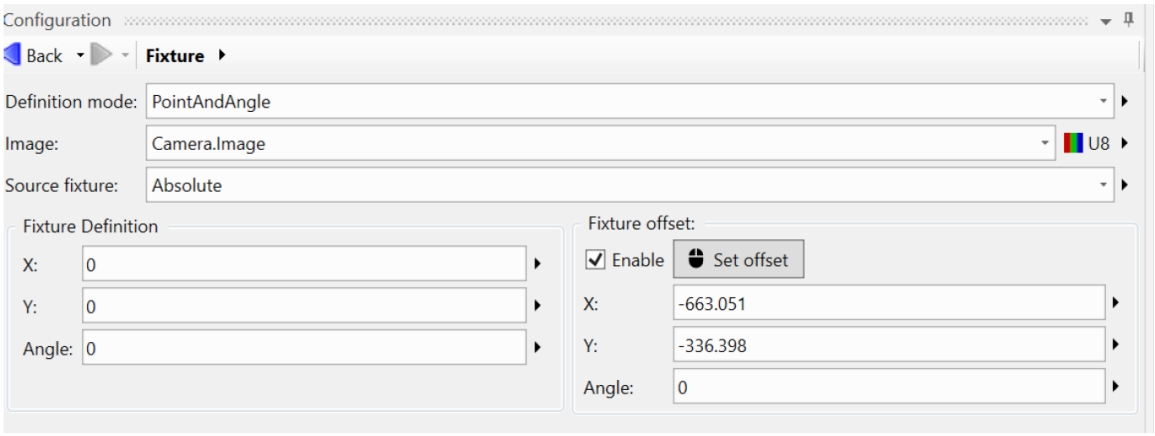


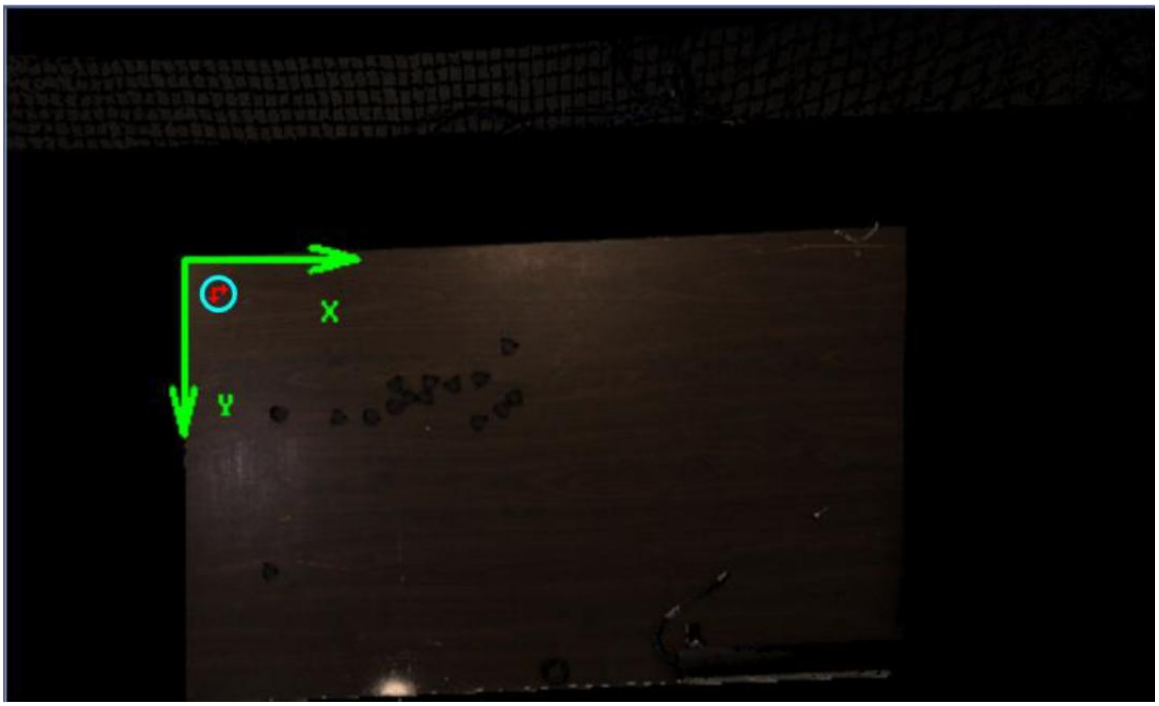
2. Set a name. The type should be “tcp,” the interface should be “any,” and the ID should be “1.”

3. Inside of Design Assistant, MODBUS needs to be enabled.



4. The robot requires an origin, a zero point to use as a reference for the coordinates generated by the 3S80 via ModelFinder. To achieve this, a fixture was created.



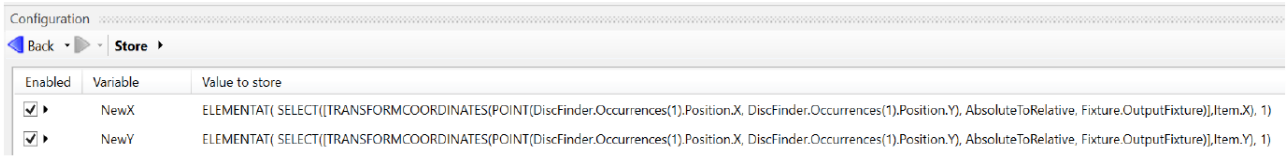


5. The coordinates of the model found by ModelFinder were transformed relative to the fixture we generated. The expressions to achieve this functionality are:

ELEMENTAT(SELECT([TRANSFORMCOORDINATES(POINT(DiscFinder.Occurrences(1).Position.X, DiscFinder.Occurrences(1).Position.Y), AbsoluteToRelative, Fixture.OutputFixture)], Item.X), 1)

ELEMENTAT(SELECT([TRANSFORMCOORDINATES(POINT(DiscFinder.Occurrences(1).Position.X, DiscFinder.Occurrences(1).Position.Y), AbsoluteToRelative, Fixture.OutputFixture)], Item.Y), 1)

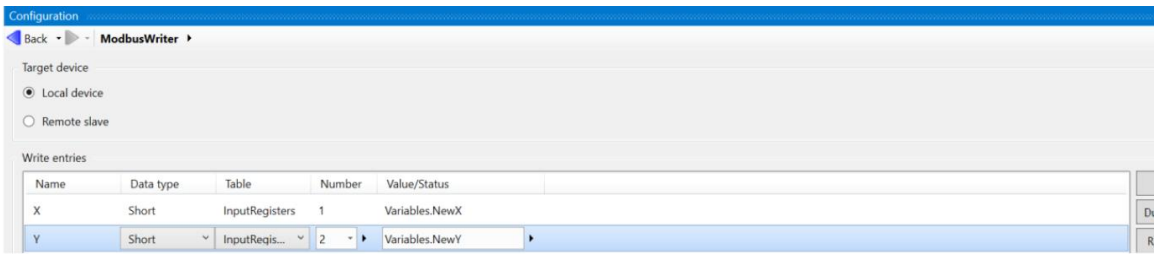
These values are then stored in two new variables to be sent to the robot via MODBUS.



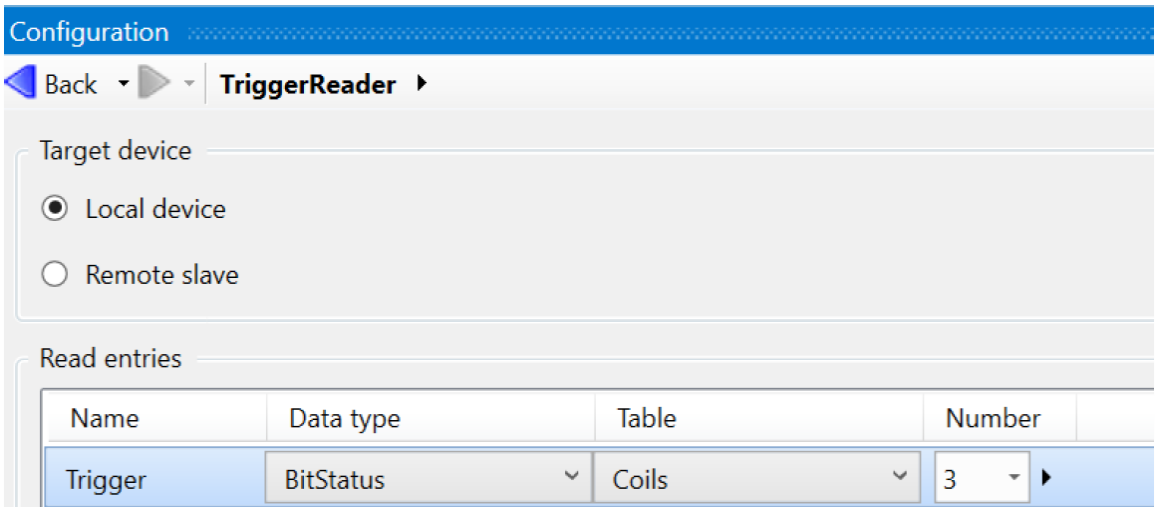
6. With the variables set, it's time to send the data to the robot using ModbusWriter. Set the target device to "Local Device," the data type to "short," the table to "input register," and assign a unique number for each entry.

⚠️ Note: The number must never be the same when sending or receiving data via MODBUS.

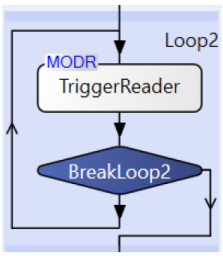
Finally, link the variables generated in step 5. Add a halt step of 100 ms after ModbusWriter to ensure data is written to the robot at a controlled rate.

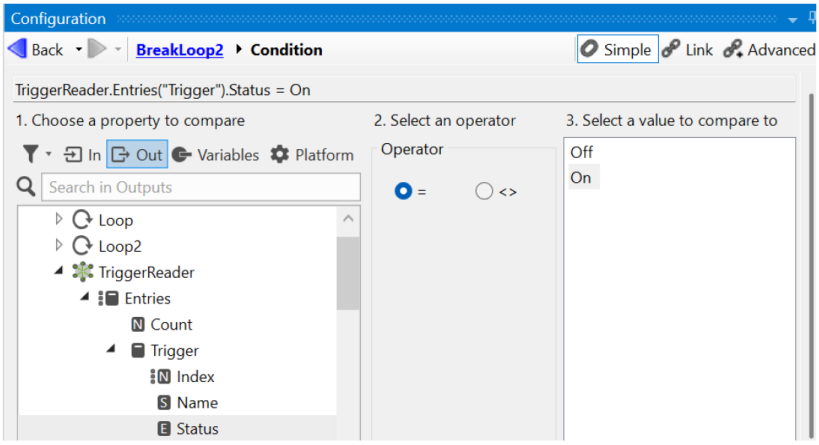


7. For our project, the robot will send a “1” or a “0” via MODBUS that DesignAssistant (DA) will interpret as a trigger. To read this signal, utilize ModbusReader. Set the target device to “Local Device,” assign a name, set the data type to “BitStatus,” the table to “Coils,” and the number to 3.

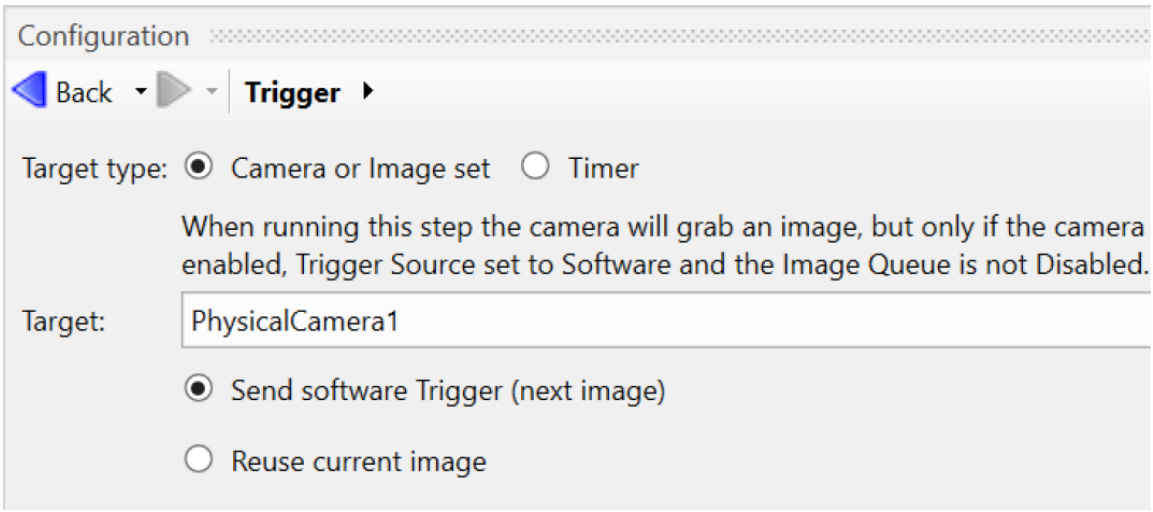


8. The trigger step can be set up in multiple ways. In this case, a loop was implemented. The flowchart does not execute unless a trigger is received via ModbusReader, which serves as the break condition for the loop.



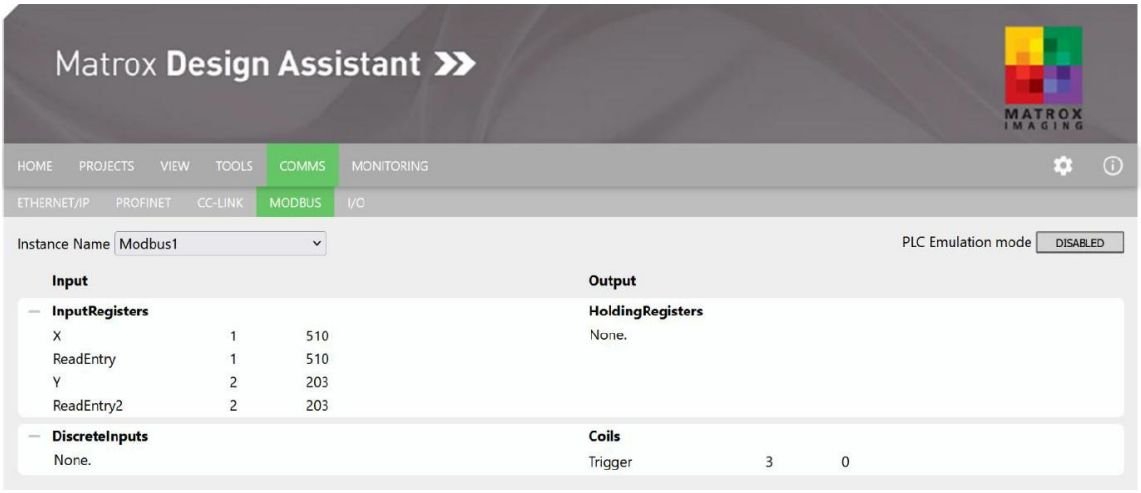


9. To finalize the vision parameters section, a trigger step was implemented to control the image acquisition of the 3S80.



10. When the ModbusWriter and ModbusReader steps are executed, the data can be visualized on localhost.

Navigate to Comms > MODBUS:



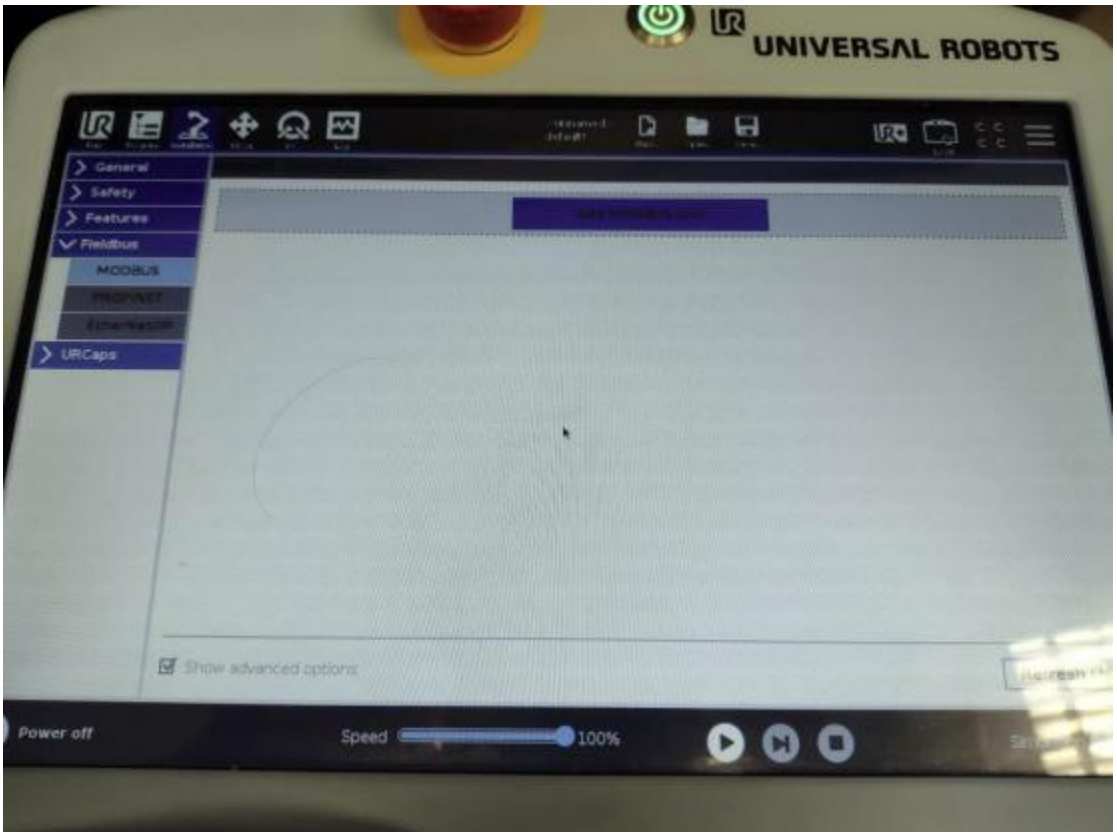
11. Before configuring the robot parameters, ensure that the IP addresses of the robot, EV7, and 3S80 are on the same network. A network switch is recommended. Connect three Ethernet cables to the switch:

- One from the EV7 to the switch
- One from the robot to the switch
- One from the 3S80 to the switch

Robot Configuration

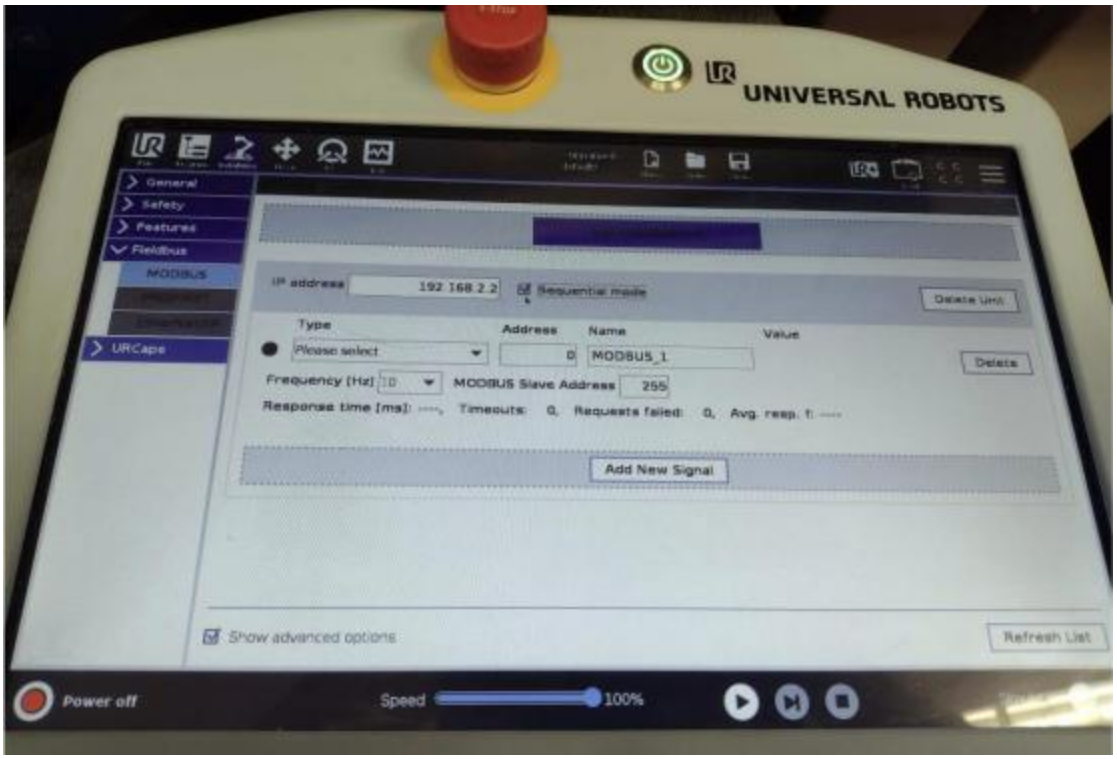
12. To start configuring the robot parameters, you must ensure that the robot is placed exactly where the fixture from Step 4 was made. Ideally, you can mark this fixture in your area before placing the robot.

13. After this has been done, you will need to configure the MODBUS communication on the UR's interface/Teach Pendant (Polyscope). Set up MODBUS by going to Installation>FieldBus>MODBUS



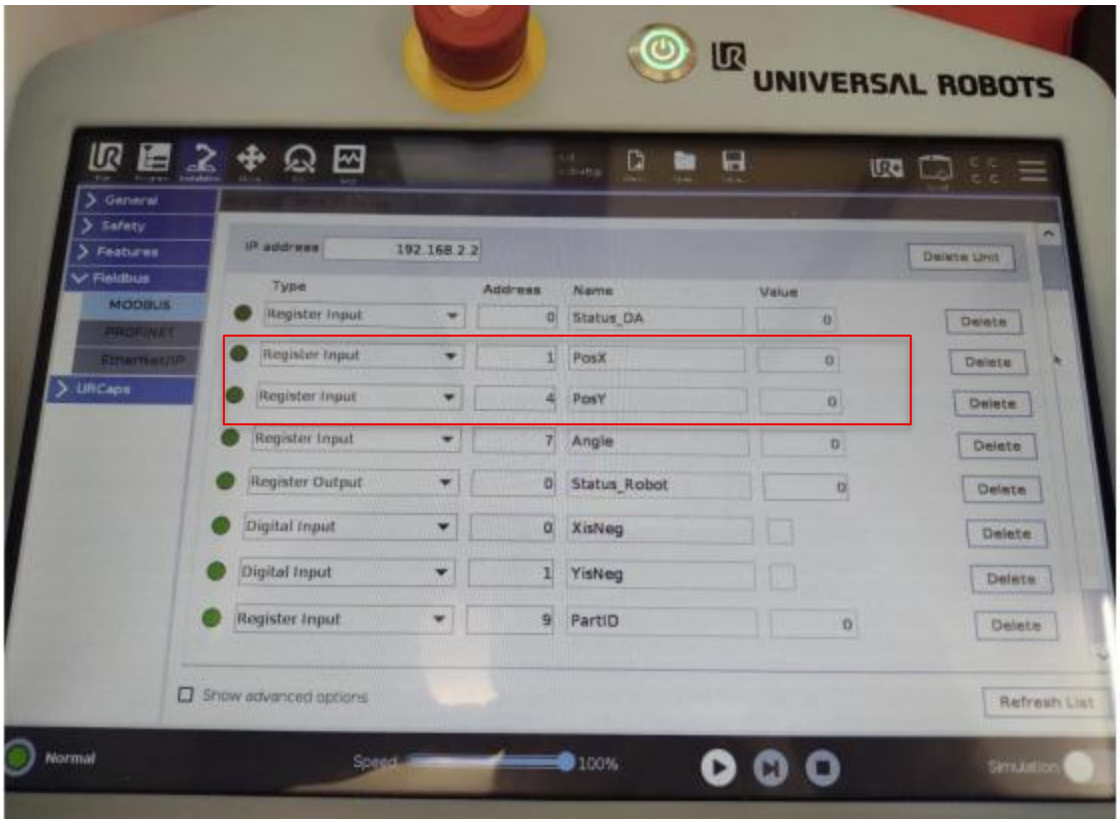
14. Press the Add MODBUS Unit and click on Show Advanced Options.

15. Enter the IP Address of your 3S80 System and click on Sequential Mode.

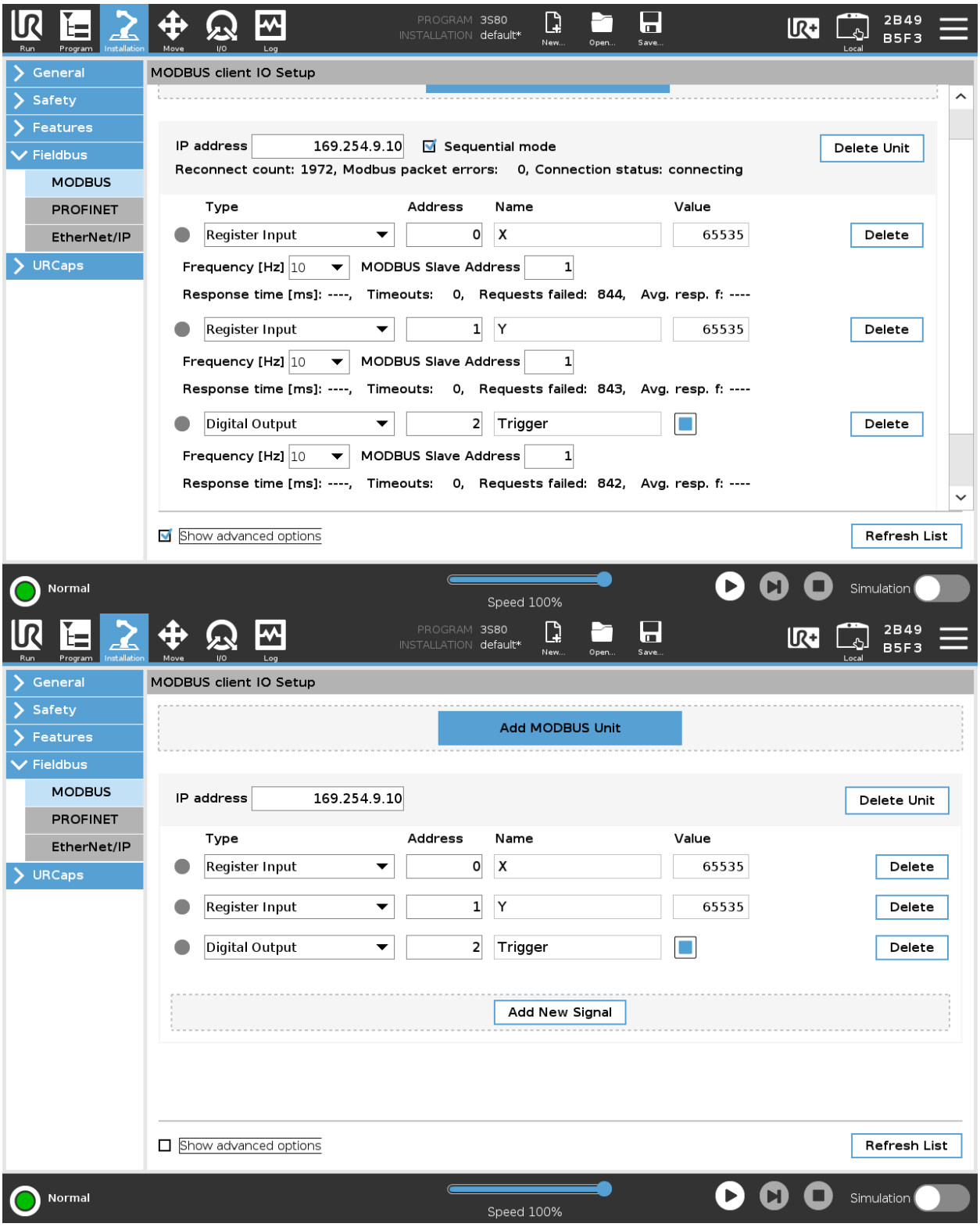


16. After this is set-up, you'll be able to add the signals that the 3S80/EV7 system has been configured to read, in our case, the X and Y positions. For this example, we would set-up two signals, one with the Address configured for the X Position, and the other with the Address configured for the Y Position. For clarity, you can rename the signals to what makes most sense to you.

Below is an example of how the finalized MODBUS set-up should look, highlighting what you would see if you set up the two X and Y signals.

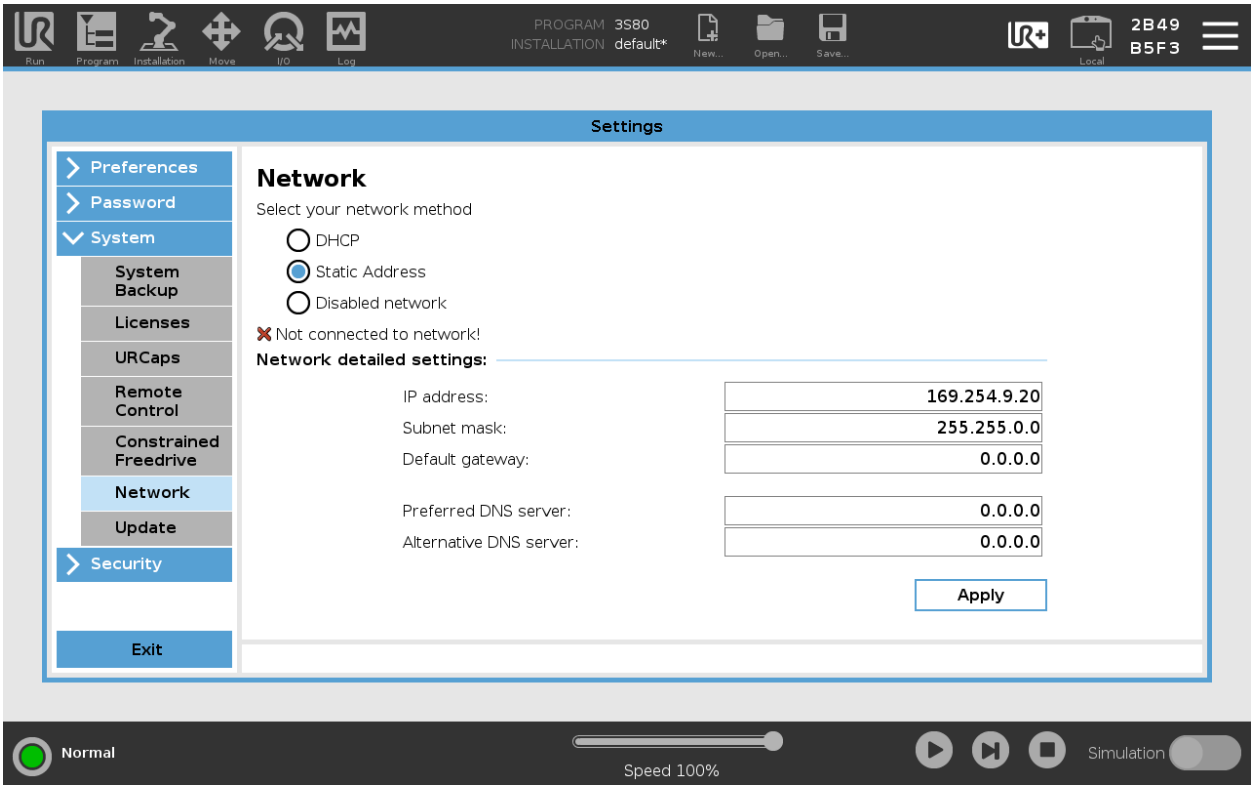


Below are the exact screenshots you would see if you follow the guide step-by-step for the MODBUS setup.


Run
Program
Installation
Move
I/O
Log
PROGRAM 3S80
INSTALLATION default*
New...
Open...
Save...
Local
2B49
B5F3
General
Safety
Features
Fieldbus
MODBUS
PROFINET
EtherNet/IP
URCaps
MODBUS client IO Setup
IP address 169.254.9.10
Sequential mode
Delete Unit
Reconnect count: 1972, Modbus packet errors: 0, Connection status: connecting
Type
Address
Name
Value
Register Input
0
X
65535
Delete
Frequency [Hz] 10
MODBUS Slave Address 1
Response time [ms]: ----, Timeouts: 0, Requests failed: 844, Avg. resp. f: ----
Register Input
1
Y
65535
Delete
Frequency [Hz] 10
MODBUS Slave Address 1
Response time [ms]: ----, Timeouts: 0, Requests failed: 843, Avg. resp. f: ----
Digital Output
2
Trigger
Delete
Frequency [Hz] 10
MODBUS Slave Address 1
Response time [ms]: ----, Timeouts: 0, Requests failed: 842, Avg. resp. f: ----
Show advanced options
Refresh List
Normal
Speed 100%
Simulation
Add MODBUS Unit
Add New Signal

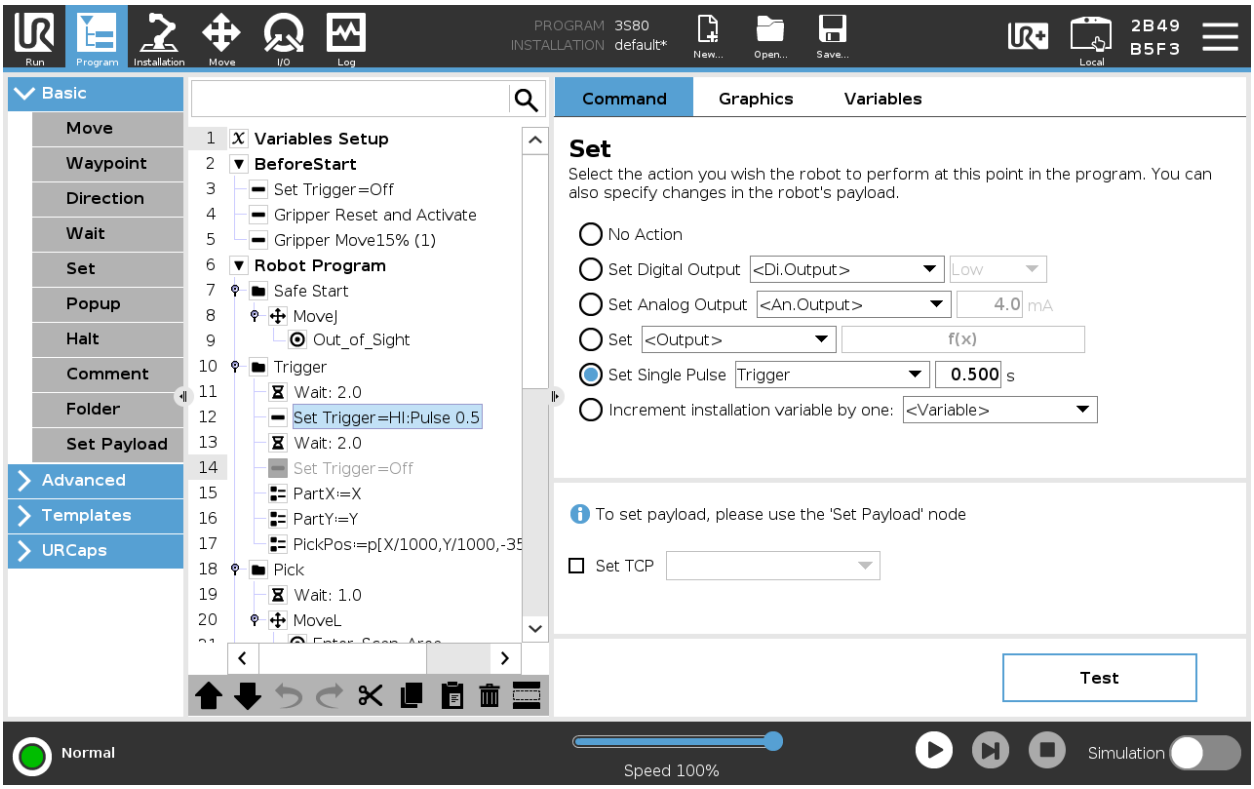
Notes:

Ensure that you have configured the robot's IP address to be a static one to match the configuration of your EV7/3S80. Below is a screenshot of the interface, you get there by clicking on the hamburger menu, going to Settings>System>Network.



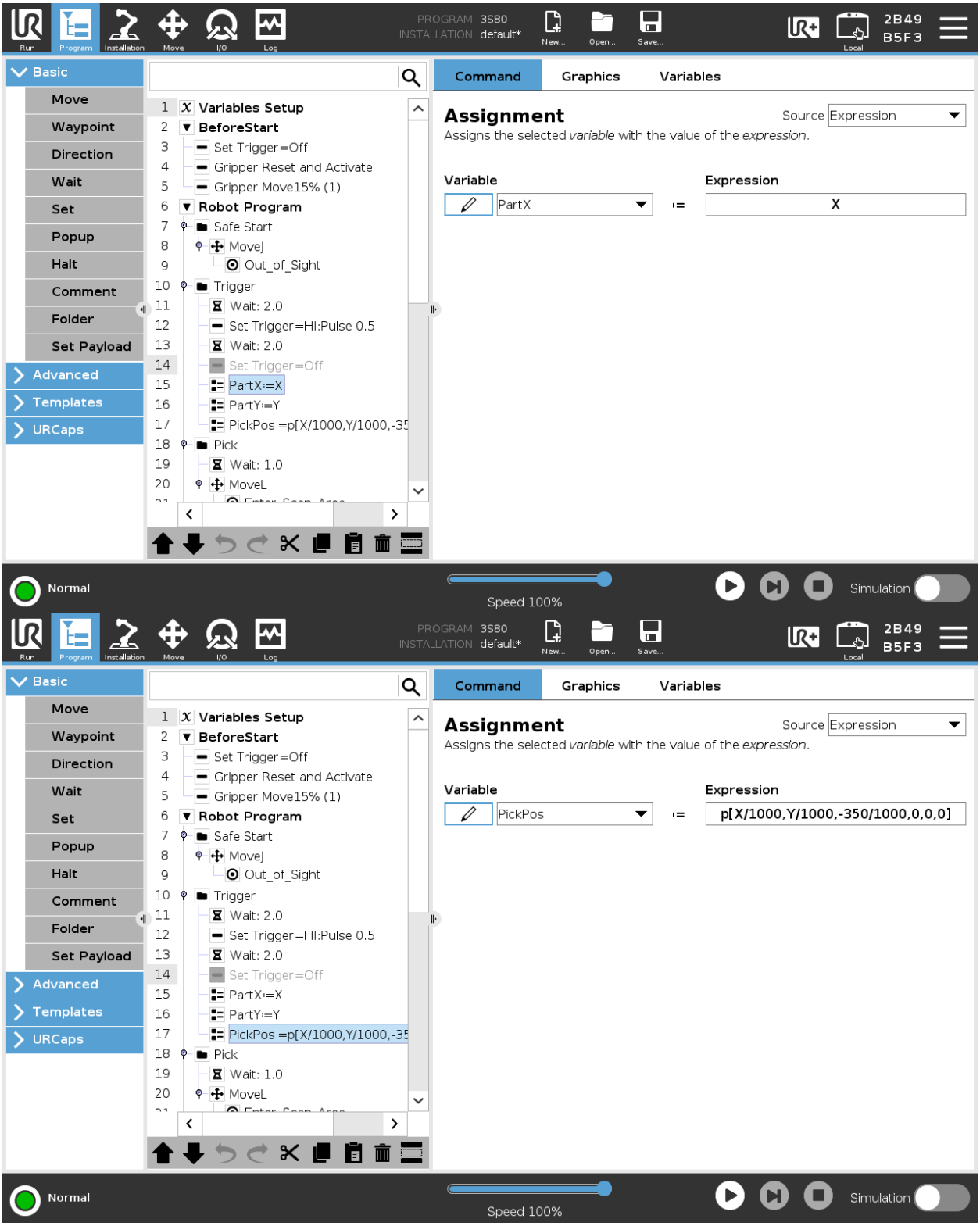
Robot Programming

Below are screenshots of the robot program made to enable for the communication from the EV7 to be integrated into the programming. After successfully setting up the MODBUS protocol and ensuring that you are receiving data from the snapshots, you can set-up the wavepoints corresponding to the data received through MODBUS.

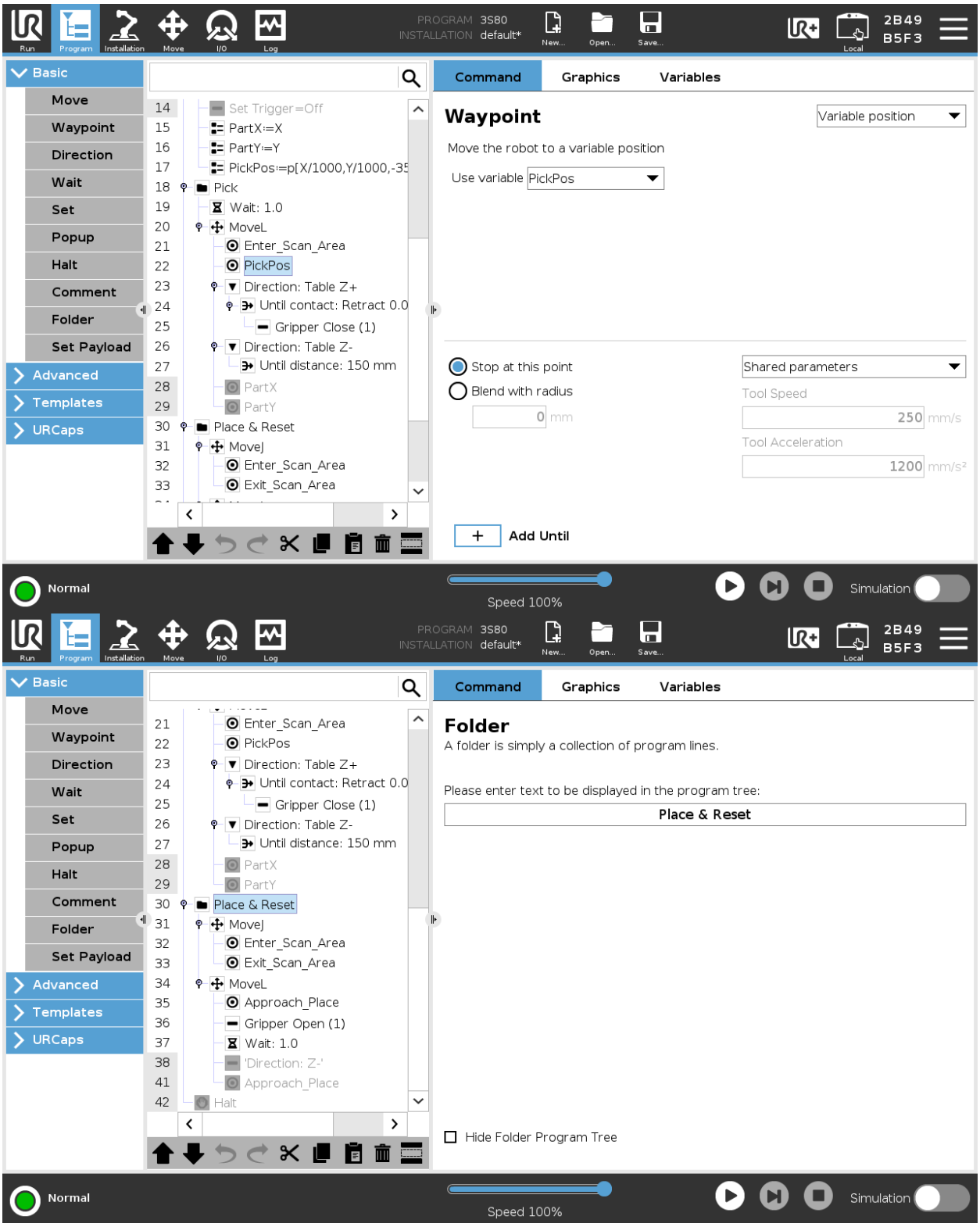


The screenshot above shows how to integrate a “Trigger” function, where the robot will send a pulse via MODBUS to the 3S80, which in turn will trigger the camera to take a new picture/capture. This is not necessary, but allows for the system to become fully autonomous, not requiring the user to take a new snapshot after every pick and place.

The bulk of the programming will change on a case to case basis, but the screenshots below show ways in which you could integrate the values received through MODBUS, like the X and Y values that the snapshot received. A good thing to note, is that UR and the 3S80 will have different values that they typically use in programming, like meters and millimeters. It is important to consider that translation when inputting those values through the Expression configurator.


Run
Program
Installation
Move
I/O
Log
PROGRAM 3S80
INSTALLATION default*
New...
Open...
Save...
Local
2B49
B5F3
Basic
Move
Waypoint
Direction
Wait
Set
Popup
Halt
Comment
Folder
Set Payload
Advanced
Templates
URCaps
Command
Graphics
Variables
Assignment
Source Expression
Assigns the selected variable with the value of the expression.
Variable
Expression
PartX
:=
X
1 Variables Setup
2 BeforeStart
3 Set Trigger=Off
4 Gripper Reset and Activate
5 Gripper Move15% (1)
6 Robot Program
7 Safe Start
8 MoveJ
9 Out_of_Sight
10 Trigger
11 Wait: 2.0
12 Set Trigger=HI:Pulse 0.5
13 Wait: 2.0
14 Set Trigger=Off
15 PartX:=X
16 PartY:=Y
17 PickPos:=p[X/1000,Y/1000,-35
18 Pick
19 Wait: 1.0
20 MoveL
Normal
Speed 100%
Simulation
PickPos
p[X/1000,Y/1000,-350/1000,0,0,0]

As mentioned previously, the X and Y values would have to be divided by 1000 to account for the change in programming. Making this assignment will make it to where you can make a relative waypoint that will have the robot move to wherever the 3S80 tells it to, thanks to our MODBUS communication. The Z value can be adjusted in different ways, the screenshot above has the robot approach the parts from a set Z distance, and then the screenshot below has the robot already right above the part, just ready to lower straight onto where it feels contact, which will then cause the gripper to close and pick the part.


Run
Program
Installation
Move
I/O
Log
PROGRAM 3S80
INSTALLATION default*
New...
Open...
Save...
Local
2B49 B5F3
Basic
Move
Waypoint
Direction
Wait
Set
Popup
Halt
Comment
Folder
Set Payload
Advanced
Templates
URCaps
Command
Graphics
Variables
Waypoint
Variable position
Move the robot to a variable position
Use variable PickPos
Stop at this point
Blend with radius
0 mm
Shared parameters
Tool Speed
250 mm/s
Tool Acceleration
1200 mm/s²
Add Until
14 Set Trigger=Off
15 PartX:=X
16 PartY:=Y
17 PickPos:=p[X/1000,Y/1000,-35
18 Pick
19 Wait: 1.0
20 MoveL
21 Enter_Scan_Area
22 PickPos
23 Direction: Table Z+
24 Until contact: Retract 0.0
25 Gripper Close (1)
26 Direction: Table Z-
27 Until distance: 150 mm
28 PartX
29 PartY
30 Place & Reset
31 MoveJ
32 Enter_Scan_Area
33 Exit_Scan_Area
Normal
Speed 100%
Simulation
Folder
A folder is simply a collection of program lines.
Please enter text to be displayed in the program tree:
Place & Reset
Hide Folder Program Tree
21 Enter_Scan_Area
22 PickPos
23 Direction: Table Z+
24 Until contact: Retract 0.0
25 Gripper Close (1)
26 Direction: Table Z-
27 Until distance: 150 mm
28 PartX
29 PartY
30 Place & Reset
31 MoveJ
32 Enter_Scan_Area
33 Exit_Scan_Area
34 MoveL
35 Approach_Place
36 Gripper Open (1)
37 Wait: 1.0
38 'Direction: Z-'
41 Approach_Place
42 Halt