# AutoVISION and Visionscape Industrial Protocol Manual

v9.0.1, June 2018

84-9220126-02 Rev C

## *Disclaimer*

The information and specifications described in this manual are subject to change without notice.

## *Latest Manual Version*

For the latest version of this manual, see the Download Center on our web site at:
**www.microscan.com**.

## *Technical Support*

For technical support, e-mail: **helpdesk@microscan.com**.

## *Warranty*

For current warranty information, see: **www.microscan.com/warranty**.

### Omron Microscan Systems, Inc.

**United States Corporate Headquarters**
+1.425.226.5700 / 800.762.1149

**United States Northeast Technology Center**
+1.603.598.8400 / 800.468.9503

**European Headquarters**
+31.172.423360

**Asia Pacific Headquarters**
+65.6846.1214

# Contents

PREFACE

# Welcome

## Purpose of This Manual

This manual contains detailed information about how to configure and deploy EtherNet/IP and PROFINET I/O-based applications using AutoVISION, Visionscape, and Omron Microscan smart cameras.

## Manual Conventions

The following typographical conventions are used throughout this manual.

- Items emphasizing important information are **bolded**.
- Menu selections, menu items and entries in screen images are indicated as: Run (triggered), Modify..., etc.

1

# Enabling Industrial Protocols

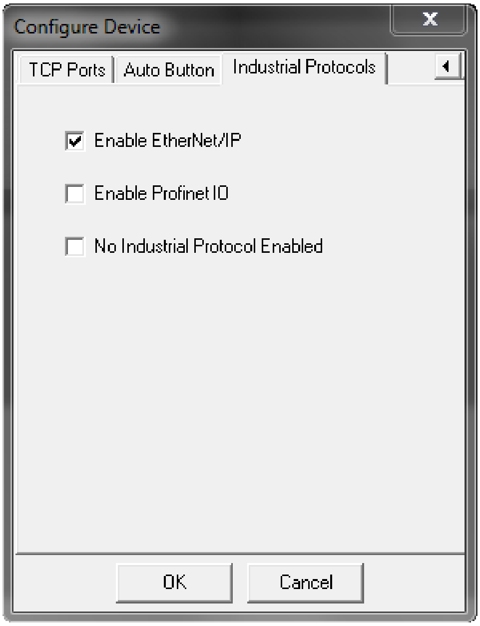
CHAPTER 1

This section describes how to enable EtherNet/IP for a Omron Microscan smart camera, and how to switch the camera's protocol between EtherNet/IP and PROFINET I/O.

# Enabling Protocols

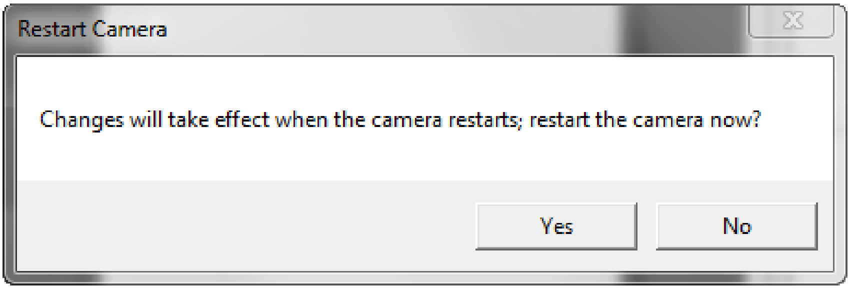
## FrontRunner

Go to the **File** menu and select **Configure Device**. Go to the **Industrial Protocols** tab. Select the industrial protocol you intend to use with the camera.



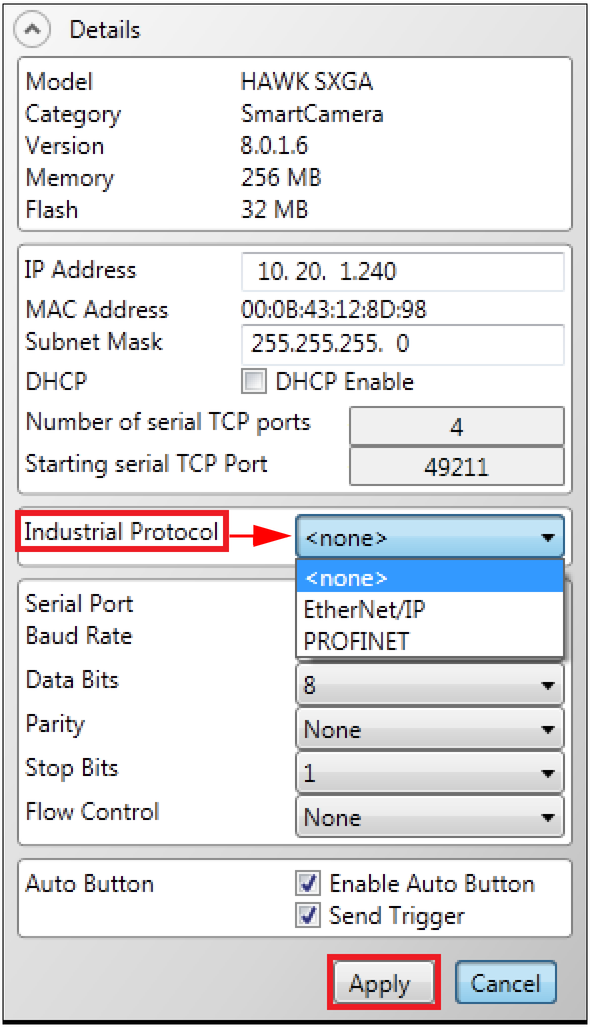

**Note:** When enabling, changing, or disabling protocols, the camera must be rebooted before the change will take effect. After clicking **OK**, you will be given the option to reboot the camera now or at a later time.

If you choose **No**, the change will not take effect until you manually reboot the camera.



## AutoVISION

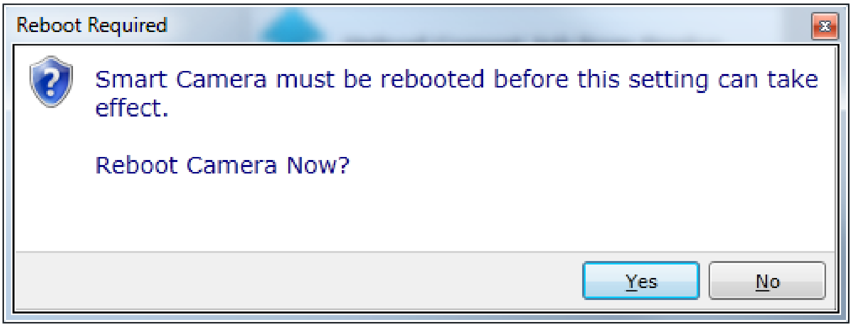
In the **Connect** view, and with a camera selected, click the button to the left of **Details** to view camera settings. Click **Modify** to change camera settings. Select the desired protocol from the **Industrial Protocol** dropdown menu, and then click **Apply**.



You will see the **Reboot Required** dialog after clicking Apply.

**Note:** A change to Industrial Protocol requires a reboot of the camera before the new setting can take effect.

Choose **Yes** and the camera will be rebooted for you. AutoVISION will be disabled while the reboot is in process. If you choose **No**, the change will not take effect until you manually reboot the camera.



2

CHAPTER 2

# Using EtherNet/IP

This section provides information necessary for using the HAWK MV-4000 and the MicroHAWK in an EtherNet/IP environment.

**Notes:**

- The camera communications protocol must be enabled for EtherNet/IP before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the camera, and information about switching camera communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate EtherNet/IP communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the Omron Microscan Link > Link Menus section of **Chapter 4** in the *AutoVISION Software User Manual*, and to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.

# EtherNet/IP

## Overview

The EtherNet/IP interface will be identified as Vendor Specific (100). The interface is designed to support Class 1 Implicit I/O data exchange, and Class 3 Explicit messages for serial commands not accessible with Implicit messaging.

## Necessary Tools

The following tools are helpful for configuring EtherNet/IP:

- AutoVISION and FrontRunner
- EtherNet/IP Messaging Tool – can be a PLC or Software Tool, must be capable ofsending explicit messages and establishing Class 1 connections. EIPScan from Pyramid Solutions is an example of such a tool.
- Terminal emulation or serial communication tool that can connect to serial uart and TCP socket, such as HyperTerminal or Putty.

## EtherNet/IP Terms of Use

EtherNet/IP Technology is governed by the Open DeviceNet Vendor Association, Inc (ODVA). Any person or entity that makes and sells products that implement EtherNet/IP Technology must agree to the Terms of Usage Agreement issued by ODVA. See **www.odva.org** for details.

## EtherNet/IP Object Model

The HAWK MV-4000 and MicroHAWK use Class 1 connected messaging to communicate most of their data and services in a single connection.

### EtherNet/IP Identity

**Device Type**

Device type is 100, Vendor Specific, Machine Vision Smart Camera.

**Vendor ID**

Microscan’s ODVA Vendor ID is 1095.

**Product Code**

MicroHAWK: 6901

HAWK MV-4000: 6902

**Interface Revision**

Major.Minor = 1.1

**Connection Properties: Class 1 Implicit Messaging**

**Input Assembly Instance (to PLC/client):** 102

**Output Assembly Instance (to camera):** 114

**Size:** Fixed, 320 bytes in both directions

**Input Trigger/Trigger Mode:** Cyclic

**RPI (Requested Packet Interval):** Greater than 20 ms recommended. 10 ms to 3.2 s allowed.

**Input Type/Connection Type:**

- Point-to-Point (PLC OUT, O > T)
- Point-to-Point (PLC IN, T > O)

**Connection Priority:** Scheduled

# Assembly Layout

## Input Assembly

The **input assembly** layout is described below.

| Bytes | Name | Description |
|---|---|---|
| **0...1** | STATUS | Status register of the camera, each bit of this register represents a different state item. See *Camera Status Register* for bit descriptions |
| **2...3** | ECHO | This 16 bit word value reflects back to the PLC the value that the PLC wrote to the output assembly ECHO register. The PLC can verify the output assembly has been written to the camera when this value matches the written value. |
| **4...7** | CmdCodeRslt | When Status.ExeCmdAck goes active in response to Control.ExeCmd, CmdCodeRslt reflects the result of the command invoked by Control.CmdCode. See *CmdCodeRslt* for definitions. |
| **8...11** | CmdRet | When Status.ExeCmdAck goes active in response to Control.ExeCmd, CmdRet contains the data returned from the command invoked by Control.CmdCode. See *CmdRet* for definitions. |
| **12...13** | reserved | Reserved for future use. |
| **14...15** | State | Device State register. Depending on the current state of the camera, certain STATUS and CONTROL features may or may not be operational. See *State* for definitions. |
| **16...17** | VIO | Each bit reflects the state of a virtual IO point. The least significant bit reflects vio point 145, the most significant bit vio point 160 |
| **18...19** | reserved | Reserved for future use. |
| **20...27** | bool1-64 | Each bit represents a bool value. The least significant bit of byte 20 reads the value of bool1. The most significant bit of byte 27 reads bool64. |
| **28...47** | int1-10 | Each pair of sequential bytes represents a 16 bit signed integer value. The 20 bytes represent 10 integers. From bytes 28 & 29 for the value of int1 through bytes 46 & 67 for the value of int10. |
| **48...87** | long1-10 | Each group of 4 bytes represents a 32 bit signed integer value. The 40 bytes represent 10 long integers. From bytes 48-51 for the value of long1 through bytes 84-87 for the value of long10. |
| **88...127** | float1-10 | Each group of 4 bytes represents a floating point value. The 40 bytes represent 10 floating point values. From bytes 88-91 for the value of float1 through bytes 124-127 for float10. |
| **128...223** | string1 | These 96 bytes can store a string of up to 92, 8 bit characters, with the first 4 bytes containing the length value. |
| **224...255** | string2 | Each of these 32 byte groups can store a string of up to 28, 8 bit characters, with the first 4 bytes containing the length value. |
| **256...287** | string3 | |
| **288...319** | string4 | |

The input assembly layout is shown here:

| Byte | | Byte | | Byte | | Byte | | Byte | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | STATUS | 64 | long5 | 128 | string1 | 192 | string1 (cont) | 256 | string3 |
| 2 | ECHO | 66 | | 130 | | 194 | | 258 | |
| 4 | CMD CODE RSLT | 68 | long6 | 132 | | 196 | | 260 | |
| 6 | | 70 | | 134 | | 198 | | 262 | |
| 8 | CMD RET | 72 | long7 | 136 | | 200 | | 264 | |
| 10 | | 74 | | 138 | | 202 | | 266 | |
| 12 | reserved | 76 | long8 | 140 | | 204 | | 268 | |
| 14 | STATE | 78 | | 142 | | 206 | | 270 | |
| 16 | VIO | 80 | long9 | 144 | | 208 | | 272 | |
| 18 | reserved | 82 | | 146 | | 210 | | 274 | |
| 20 | bool1...16 | 84 | long10 | 148 | | 212 | | 276 | |
| 22 | bool17...32 | 86 | | 150 | | 214 | | 278 | |
| 24 | bool33...48 | 88 | float1 | 152 | | 216 | | 280 | |
| 26 | bool49...64 | 90 | | 154 | | 218 | | 282 | |
| 28 | int1 | 92 | float2 | 156 | | 220 | | 284 | |
| 30 | int2 | 94 | | 158 | | 222 | | 286 | |
| 32 | int3 | 96 | float3 | 160 | | 224 | string2 | 288 | string4 |
| 34 | int4 | 98 | | 162 | | 226 | | 290 | |
| 36 | int5 | 100 | float4 | 164 | | 228 | | 292 | |
| 38 | int6 | 102 | | 166 | | 230 | | 294 | |
| 40 | int7 | 104 | float5 | 168 | | 232 | | 296 | |
| 42 | int8 | 106 | | 170 | | 234 | | 298 | |
| 44 | int9 | 108 | float6 | 172 | | 236 | | 300 | |
| 46 | int10 | 110 | | 174 | | 238 | | 302 | |
| 48 | long1 | 112 | float7 | 176 | | 240 | | 304 | |
| 50 | | 114 | | 178 | | 242 | | 306 | |
| 52 | long2 | 116 | float8 | 180 | | 244 | | 308 | |
| 54 | | 118 | | 182 | | 246 | | 310 | |
| 56 | long3 | 120 | float9 | 184 | | 248 | | 312 | |
| 58 | | 122 | | 186 | | 250 | | 314 | |
| 60 | long4 | 124 | float10 | 188 | | 252 | | 316 | |
| 62 | | 126 | | 190 | | 254 | | 318 | |

## Status: Camera Status Register (16-bit)

Each bit of this register represents a different state of the camera's operation. A high value of **1** indicates that state is active (true).

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | DATA VALID | INSP STAT | INSP BUSY | TRIGGER ACK | EXE CMD ACK | | RESET COUNT ACK | ERROR | TRIGGER READY | ACQ BUSY | EXP BUSY | ONLINE |
| Inspection 1 | | | | | | | | All Inspections | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| **0** | ONLINE | Inspections are running |
| **1** | EXP BUSY | The camera is busy capturing an image. The camera should not be triggered or the part under inspection moved during this time if illuminated. |
| **2** | ACQ BUSY | The camera is busy acquiring an image. The camera cannot be triggered while busy. |
| **3** | TRIGGER READY | The camera is ready to be triggered. This is equivalent to ONLINE == 1 and ACQ BUSY == 0. |
| **4** | ERROR | An error has occurred. Set the RESET ERROR control bit high to clear. |
| **5** | RESET COUNT ACK | This bit mirrors the RESET COUNT control bit. The PLC can be certain the reset command was received by the camera when this goes high. The PLC can then bring the RESET COUNT control signal back low. |
| **7** | EXE CMD ACK | This bit mirrors the EXE CMD control bit. |
| **8** | TRIGGER ACK | This bit mirrors the TRIGGER control bit. |
| **9** | INSP BUSY | This bit is high when inspection 1 is busy processing an image. |
| **10** | INSP STAT | This bit represents the inspection 1 status result. It is 1 if the inspection passes. It is only valid when DataValid goes high. |
| **11** | DATA VALID | This bit goes high when inspection 1 is complete. The PLC should clear this signal by setting RESET DV high once it has read results. |

## CmdCodeRslt (32-bit)

The value of **CmdCodeRslt** is only valid when **ExeCmdAck** is active (**1**), in response to **ExeCmd** being active.

| CmdCodeRslt value (base 16 hex) | Meaning |
|---|---|
| **0x0000_0000** | Success |
| **0x0100_0000** | Fail.<br>Possible reasons:<br>Camera under PC control.<br>Job cannot be changed. |
| **0x0200_0000** | Fail: No Job in slot. |
| **0x0300_0000** | Fail: Unknown cmd. |

## CmdRet (32-bit)

The value of **CmdRet** is only valid when **ExeCmdAck** is active (**1**), in response to **ExeCmd** being active, and **CmdCodeRslt** is **0** (**Success**). The following table shows which CmdCodes return data in the CmdRet register.

| CmdRet value (32 bit) | Associated CmdCode | Meaning |
|---|---|---|
| **0** | 0x1000_0000<br>to<br>0x1300_0000<br>(Job Change type) | Na |
| **1 – 255** | 0x1800_0000<br>(Query Active Job Slot) | Active Job Slot # |

## State (16-bit)

**State** reflects the following operational condition of the camera:

| State value (16 bit) | Meaning | Typical action required by the client (plc), or system operator |
|---|---|---|
| **0** | Offline | Perform job change or put camera online. |
| **1** | Online | Normal runtime operation: Monitor TriggerReady and DataValid signals. Trigger the camera. |
| **2** | Changing Vision Job | If camera is under pc control:<br>Wait until State changes to Offline or Online.<br>If plc is controlling the job change:<br>Use ExeCmd, CmdCode, ExeCmdAck, and CmdCodeRslt to complete the operation. |
| **3** | Booting* | Wait for camera to transition to Online or Offline. |
| **4** | Empty (no Vision Job) | Load a new job from AutoVISION or Front Runner. |

*Booting (3) State: This will rarely be seen by the PLC.

The value of State determines which **Control** and **Status** signals are available:

Control/Status Signal State

| | 0 (Offline) | 1 (Online) | 2 (Job Change) | 3 (Booting) | 4 (Empty) |
|---|---|---|---|---|---|
| Control.GO ONLINE | Y | | | | |
| ".GO OFFLINE | | Y | | | |
| ".RESET ERROR | | | | | |
| ".RESET COUNT | Y | Y | | | |
| ".EXE CMD | Y | Y | Y | | Y |
| ".TRIGGER | | Y | | | |
| ".RESET DATA VALID | | Y | | | |
| Status.ONLINE | Y | Y | Y | Y | Y |
| ".ERROR | | | | | |
| ".RESET COUNT ACK | Y | Y | | | |
| ".EXE CMD ACK | Y | Y | Y | | Y |
| ".EXP BUSY | | Y | | | |
| ".ACQ BUSY | | Y | | | |
| ".TRIGGER READY | | Y | | | |
| ".TRIGGER ACK | | Y | | | |
| ".INSP BUSY | | Y | | | |
| ".INSP STAT | | Y | | | |
| ".DATA VALID | | Y | | | |

Where:

**Y** = Signal is valid for this State

**Empty table cell** = Signal is not valid for this State

## VIO Register Bits

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v160 | v159 | v158 | v157 | v156 | v155 | v154 | v153 | v152 | v151 | v150 | v149 | v148 | v147 | v146 | v145 |

## Output Assembly

The **output assembly** layout is described below and shown in the following diagram.

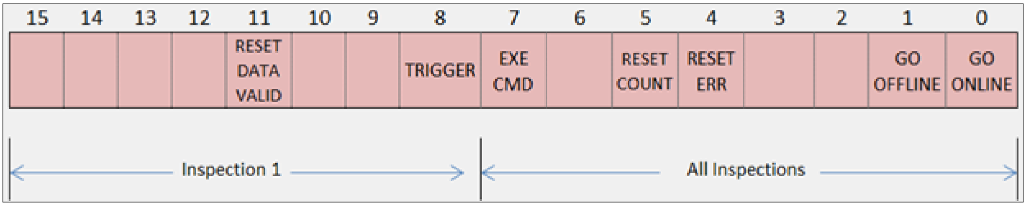
| Bytes | Name | Description |
|---|---|---|
| **0...1** | CONTROL | Control register of camera. Each bit of this register represents a different status item. See *Camera Control Register* for bit descriptions |
| **2...3** | ECHO | This 16 bit value is reflected back to the PLC in the input assembly ECHO register. The PLC can verify the output assembly has been written to the camera when the input assembly matches this written value. |
| **4...7** | CmdCode | Specifies the process invoked in the camera when Control.ExeCmd goes active. See *CmdCode* for definitions. |
| **8...11** | CmdArg | Additional argument data for the CmdCode. See *CmdArg* for definition. |
| **12...15** | reserved | Reserved for future use. |
| **16...17** | VIO | Each bit reflects the state of a virtual IO point. The least significant bit reflects vio point 129, the most significant bit is vio point 144 |
| **18...19** | Reserved | Reserved for future use. |
| **20...27** | bool | Each bit represents a bool value. The least significant bit of byte 20 writes the value of bool101. The most significant bit of byte 27 writes bool164. |
| **28...47** | int101-110 | Each pair of sequential bytes represents a 16 bit signed integer value. The 20 bytes represent 10 integers. From bytes 28 & 29 to write the value of int101 through bytes 46 & 67 for the value of int110. |
| **48...87** | long101-110 | Each group of 4 bytes represents a 32 bit signed integer value. The 40 bytes represent 10 long integers. From bytes 48-51 for the value of long101 through bytes 84-87 for the value of long110. |
| **88...127** | float101-110 | Each group of 4 bytes represents a floating point value. The 40 bytes represent 10 floating point values. From bytes 88-91 for the value of float101 through bytes 124-127 for the value of float110. |
| **128...223** | string101 | These 96 bytes can store a string of up to 92 bytes, with the first 4 bytes containing the length value. |
| **224...255** | string102 | Each of these 32 byte groups can store a string of up to 28 bytes, with the first 4 bytes containing the length value. |
| **256...287** | string103 | |
| **288...319** | string104 | |

The output assembly layout is shown here:

| Byte | | Byte | | Byte | | Byte | | Byte | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | CONTROL | 64 | long105 | 128 | string101 | 192 | string101 (cont) | 256 | string103 |
| 2 | ECHO | 66 | | 130 | | 194 | | 258 | |
| 4 | CMD CODE | 68 | long106 | 132 | | 196 | | 260 | |
| 6 | | 70 | | 134 | | 198 | | 262 | |
| 8 | CMD ARG | 72 | long107 | 136 | | 200 | | 264 | |
| 10 | | 74 | | 138 | | 202 | | 266 | |
| 12 | reserved | 76 | long108 | 140 | | 204 | | 268 | |
| 14 | | 78 | | 142 | | 206 | | 270 | |
| 16 | VIO | 80 | long109 | 144 | | 208 | | 272 | |
| 18 | reserved | 82 | | 146 | | 210 | | 274 | |
| 20 | bool101...116 | 84 | long110 | 148 | | 212 | | 276 | |
| 22 | bool117...132 | 86 | | 150 | | 214 | | 278 | |
| 24 | bool133...148 | 88 | float101 | 152 | | 216 | | 280 | |
| 26 | bool149...164 | 90 | | 154 | | 218 | | 282 | |
| 28 | int101 | 92 | float102 | 156 | | 220 | | 284 | |
| 30 | int102 | 94 | | 158 | | 222 | | 286 | |
| 32 | int103 | 96 | float103 | 160 | | 224 | string102 | 288 | string104 |
| 34 | int104 | 98 | | 162 | | 226 | | 290 | |
| 36 | int105 | 100 | float104 | 164 | | 228 | | 292 | |
| 38 | int106 | 102 | | 166 | | 230 | | 294 | |
| 40 | int107 | 104 | float105 | 168 | | 232 | | 296 | |
| 42 | int108 | 106 | | 170 | | 234 | | 298 | |
| 44 | int109 | 108 | float106 | 172 | | 236 | | 300 | |
| 46 | int110 | 110 | | 174 | | 238 | | 302 | |
| 48 | long101 | 112 | float107 | 176 | | 240 | | 304 | |
| 50 | | 114 | | 178 | | 242 | | 306 | |
| 52 | long102 | 116 | float108 | 180 | | 244 | | 308 | |
| 54 | | 118 | | 182 | | 246 | | 310 | |
| 56 | long103 | 120 | float109 | 184 | | 248 | | 312 | |
| 58 | | 122 | | 186 | | 250 | | 314 | |
| 60 | long104 | 124 | float110 | 188 | | 252 | | 316 | |
| 62 | | 126 | | 190 | | 254 | | 318 | |

## Control: Camera Control Register (16-bit)

Each bit of this register controls a function on the camera. Transitions from a low state of **0** to a high state of **1**, initiates the associate operation. The PLC should return the state of the control bit back to **0** after it has acknowledged the camera has processed the control. Unused bits should remain **0**.




| Bit | Name | Description |
|---|---|---|
| **0** | GO ONLINE | Start all inspections running |
| **1** | GO OFFLINE | Stop all inspections |
| **4** | RESET ERROR | Reset ERROR in the Status register |
| **5** | RESET COUNT | Reset all inspection counts |
| **7** | EXE CMD | Execute the command specified by Control.CmdCode |
| **8** | TRIGGER | Trigger Inspection 1. The inspection must be configured for a triggered image acquisition. |
| **11** | RESET DATA VALID | Reset the Data Valid signal of the Status register |

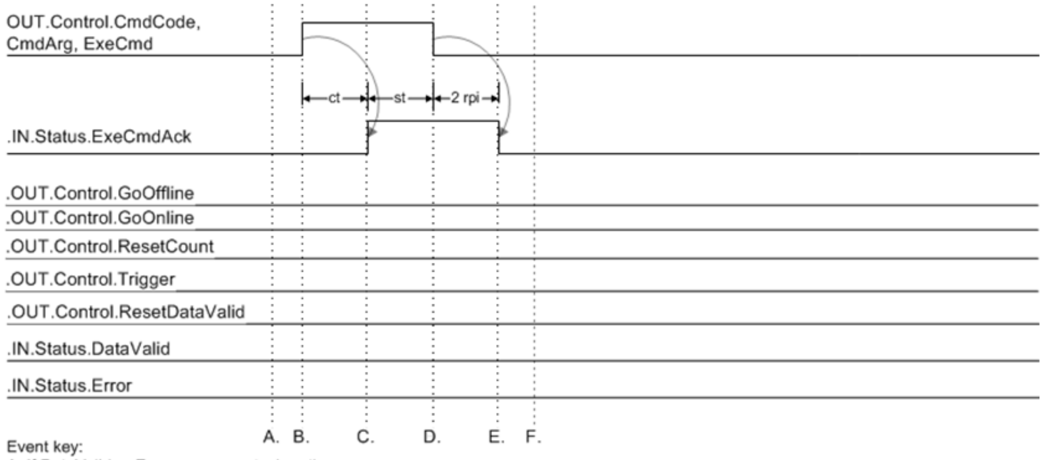
## CmdCode and CmdArg (32-bit)

Specifies the process invoked in the camera when **Control.ExeCmd** goes active.

List of available CmdCodes, and associated CmdArg:

| **CmdCode value** | **CmdArg** | **Operations performed** |
|---|---|---|
| **0x1000_0000** | Job Slot (1-255) | Go Offline, Load job from specified slot |
| **0x1100_0000** | Job Slot (1-255) | Go Offline, Load job from specified slot, Go Online |
| **0x1200_0000** | Job Slot (1-255) | Go Offline, Load job from specified slot, Make it the boot job |
| **0x1300_0000** | Job Slot (1-255) | Go Offline, Load job from specified slot, Make it the boot job, and Go Online |
| **0x1800_0000** | na | Query active job slot. *CmdRet* will contain the active job slot number when the operation is done. |

## CmdCode and ExeCmd Operation




Event key:

A. If DataValid or Error are present, clear them.
   Set the following control signals idle, and keep them idle while the command is processed by the camera:
   GoOffline, GoOnline, Trigger, ResetDataValid, ResetCount, ResetError.
   If the command operation is a job change, populate the output tags required to configure the new job (bool, int, long, float, string).

B. Populate CmdCode and CmdArg, then activate ExeCmd.

C. Camera executes the command (may take up to a minute). While processing a Job Change command, State will be 2. Camera activates ExeCmdAck when it is done processing the command.

D. When the PLC sees an active ExeCmdAck, verify CmdCodeRslt is 0, and Error is 0. Process CmdRet if needed, then clear ExeCmd.

E. Camera clears ExeCmdAck when ExeCmd goes inactive. When ExeCmdAck goes inactive, CmdCodeRslt and CmdRet are no longer valid, and it may take a few seconds for the camera State and Online signals to settle to a final value (typically Online or Offline).

F. Camera can now be put online and triggered.

Notes:
st = PLC program scan time
ct = Command processing time in the camera. May take up to a minute for some commands.
rpi = Requested Packet Interval. Configured in the plc's EIP module connection properties. Allowed rpi is 10 ms to 3.2 s.
All signals represent the state of plc tags.

## VIO Register Bits

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v144 | v143 | v142 | v141 | v140 | v139 | v138 | v137 | v136 | v135 | v134 | v133 | v132 | v131 | v130 | v129 |

# Connection Properties: Class 3 Explicit Messaging

All Class 1 I/O assembly data and additional data are accessible via Explicit message. Input data (camera to PLC/Client) occupies attributes **1** to **100** of the classes. Output data (PLC/Client to camera) occupies attributes **101** to **200**.

## Service:

- Get Attribute Single (0xE)
- Set Attribute Single (0x10)

## Classes:

- bool = 104 (0x68)
- int = 105 (0x69)
- long = 106 (0x6A)
- float = 107 (0x6B)
- string = 108 (0x6C)
- control/status (mixed data types) = 109 (0x6D)

## Instance: 1

### Attribute:

- 1 to 100 = In to PLC/Client
- 101 to 200 = Out to Camera

### Attribute Layout

When using explicit EtherNet/IP messaging, all global data objects can be read or written. Each data type is stored in its own class object and an instance of 1 to read the global data. For example, to read **float2**, the EtherNet/IP request would be for **Service Code 14 (0xE)**, **Class 107 (0x6B)**, **Instance 1**, **Attribute 2**.

| Attr# | Class 104 | Attr# | Class 105 | Attr# | Class 106 | Attr# | Class 107 | Attr# | Class 108 | Attr# | Class 109 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | bool1 | 1 | int1 | 1 | long1 | 1 | float1 | 1 | string1 | 1 | CONTROL |
| 2 | bool2 | 2 | int2 | 2 | long2 | 2 | float2 | 2 | string2 | 2 | STATUS |
| 3 | bool3 | 3 | int3 | 3 | long3 | 3 | float3 | 3 | string3 | 3 | |
| 4 | bool4 | 4 | int4 | 4 | long4 | 4 | float4 | 4 | string4 | 4 | |
| 5 | bool5 | 5 | int5 | 5 | long5 | 5 | float5 | 5 | string5 | 5 | |
| 6 | bool6 | 6 | int6 | 6 | long6 | 6 | float6 | 6 | string6 | 6 | ECHO |
| 7 | bool7 | 7 | int7 | 7 | long7 | 7 | float7 | 7 | string7 | 7 | CMD CODE |
| 8 | bool8 | 8 | int8 | 8 | long8 | 8 | float8 | 8 | string8 | 8 | CMD ARG |
| 9 | bool9 | 9 | int9 | 9 | long9 | 9 | float9 | 9 | string9 | 9 | CMD CODE RSLT |
| 10 | bool10 | 10 | int10 | 10 | long10 | 10 | float10 | 10 | string10 | 10 | CMD RET |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | 11 | STATE |
| | | | | | | | | | | ... | |
| 199 | bool199 | 199 | int199 | 199 | long199 | 199 | float199 | 199 | string199 | 199 | |
| 200 | bool200 | 200 | int200 | 200 | long200 | 200 | float200 | 200 | string200 | 200 | |

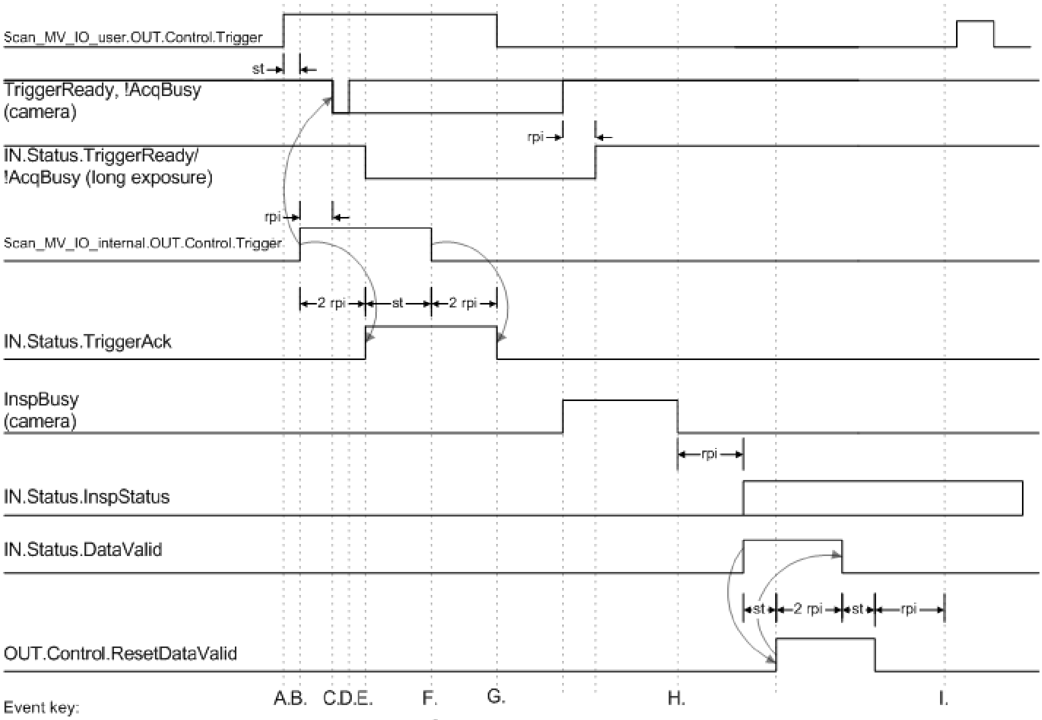
The value received in response to **Get Attribute Single** depends on the type:

- **Bool** will return a 16-bit word with **0** for false or **1** for true
- **Ints** will return a 16-bit signed integer
- **Longs** will return a 32-bit signed integer
- **Floats** will return a 32-bit floating point number
- **Strings** will return a counted string. Total size of a string data item is **2048 bytes**. This includes a 4 byte “length” field followed by 2044 eight bit characters. When accessing strings explicitly, they are not limited to the size in the I/O assemblies. For example, **string3** is limited to **28 bytes** in the input assembly. If the actual string is longer than 28 bytes, it will be truncated when reading via the assembly, but not truncated when reading the same string via an attribute explicitly.

**Assembly Class 109** can be used to read and write special EtherNet/IP-specific registers.

| Attr# | Name | Description |
|---|---|---|
| **1** | CONTROL | The control register (16 bit). See *Camera Control Register* for bit definitions. |
| **2** | STATUS | The status register (16 bit). See *Camera Status Register* for bit definitions. |
| **6** | ECHO | The ECHO register (16 bit) (read only if implicit write is enabled) |
| **7** | CMD CODE | The command code register (32 bit). See *CmdCode*. |
| **8** | CMD ARG | The command argument register (32 bit). See *CmdArg*. |
| **9** | CMD CODE RSLT | The command code result register (32 bit). See *CmdCodeRslt*. |
| **10** | CMD RET | The command return value register (32 bit). See *CmdRet*. |
| **11** | STATE | The device state register (16 bit). See *State* for definitions. |

# EtherNet/IP Control/Status Signal Operation




Event key:
A. On rising edge of system trigger, the user app activates Scan_MV_IO_user.OUT.Control.Trigger to trigger the demo code.
B. Demo code detects rising edge of Scan_MV_IO_user.OUT.Control.Trigger, and if the camera is ready, sends a trigger to the camera.
C. Camera acquisition begins (may be delayed by one rpi).
D. If the camera's exposure time is shorter than the rpi, no change will be seen in TriggerReady and AcqBusy plc IN tags.
E. Camera firmware acks the trigger. The demo code may not see the ack until two rpi after the trigger was sent (event B).
F. Demo code detects TriggerAck and clears the Trigger.
G. Demo code detect falling edge of TriggerAck and clears the user Trigger.
H. Camera internal signal DataValid will go high when InspBusy goes low
I. Plc logic must delay one rpi time before re-asserting ResetDataValid

Notes:
1. The chart shows the workings of the Trigger and ResetDataValid Control signals, and the TriggerAck and DataValid Status signals.
2. st = plc program scan time
3. rpi = Requested Packet Interval. Configured in the plc's EIP module connection properties. Allowed rpi is 10 ms to 3.2 s.
4. All signals represent the state of plc tags, except where noted as "(camera)". The cam signals shown are visible in the EIP interface, but the state of the plc tags and internal firmware signals will be different for at least one or two requested packet intervals (rpi).
5. The plc is running the demo code distributed with the camera. The demo code and user app use the Scan_MV_IO_user tag set as the primary control, status, and data interface for the user app. All signal operations are still true even if the plc demo code is not used.
6. TriggerReady/!AcqBusy: Camera exposure times can range from less than 1 ms, up to 100 ms.

# Data Type Descriptions and Equivalents in PLC and EDS/CIP Environments

| AV | Description | RSLogix equivalent | Description | EDS / EIP equivalents | Description |
|---|---|---|---|---|---|
| **Bool** | 1 bit | BOOL | 1 bit | BOOL | 1 bit |
| | | | | WORD | 16 BOOLs |
| | | | | LWORD | 64 BOOLs |
| **Int** | 16 bit signed integer | INT | 16 bit signed integer | INT | 16 bit signed integer |
| **Long** | 32 bit signed integer | DINT | 32 bit signed integer | DINT | 32 bit signed integer |
| **Float** | 32 bit floating point | REAL | 32 bit floating point | REAL | 32 bit floating point |
| **String** | 32 bit length field followed by 8 bit ASCII characters | STRING | 32 bit length field followed by 8 bit ASCII characters | DINT + USINT[] | DINT (length) + USINT array of characters. USINT = 8 bit integer |

# PLC Tags and Serial Command Names

PLC tags are separated into **IN** and **OUT** for data direction. Within the IN and OUT groups, the tags are sub-divided into fixed **Status** and **Control** fields, plus user-defined linked data fields. This table shows how PLC tag names correspond to serial commands.

| IN | | | OUT | | |
|---|---|---|---|---|---|
| **PLC tag prefix** | **Serial cmd prefix** | **Tag name** | **PLC tag prefix** | **Serial cmd prefix** | **Tag name** |
| IN.Status. | eip.status. | Online (1) | OUT.Control. | eip.control. | GoOnline[i] |
| IN.Status. | eip.status. | Online (0) | OUT.Control. | eip.control. | GoOffline[ii] |
| IN.Status. | eip.status. | Error | OUT.Control. | eip.control. | ResetError |
| IN.Status. | eip.status. | ResetCountAck | OUT.Control. | eip.control. | ResetCount |
| IN.Status. | eip.status. | TriggerAck | OUT.Control. | eip.control. | Trigger |
| IN.Status. | eip.status. | DataValid | OUT.Control. | eip.control. | ResetDataValid |
| IN.Status. | eip.status. | ExeCmdAck | OUT.Control. | eip.control. | ExeCmd |
| IN.Status. | eip.status. | TrigReady[iii] | - | - | - |
| IN.Status. | eip.status. | AcqBusy | - | - | - |
| IN.Status. | eip.status. | ExpBusy | - | - | - |
| IN.Status. | eip.status. | InspBusy | - | - | - |
| IN.Status. | eip.status. | InspStat | - | - | - |
| IN.Status. | eip. | Echo | OUT.Control. | eip. | Echo |
| IN.Status. | eip. | CmdCodeRslt | OUT.Control | eip. | CmdCode |
| IN.Status | eip. | CmdRet | OUT.Control | eip. | CmdArg |
| IN.Status. | eip. | State | - | - | - |
| IN.vio. | io. | v[145-160] | OUT.vio. | io. | v[129-144] |
| IN.bool. | eip. | bool[1-100] | OUT.bool. | eip. | bool[101-200][iv] |
| IN.int. | eip. | int[1-100] | OUT.int. | eip. | int[101-200][v] |
| IN.long. | eip. | long[1-100] | OUT.long. | eip. | long[101-200] |
| IN.float. | eip. | float[1-100] | OUT.float. | eip. | float[101-200] |
| IN.string. | eip. | string[1-100] | OUT.string. | eip. | string[101-200] |

[i] When GoOnline is changed from 0 to 1, Online goes to 1.
[ii] When GoOffline is changed from 0 to 1, Online goes to 0.
[iii] TrigReady, AcqBusy, ExpBusy, InspBusy, and InspStat are all IN-direction data only.
[iv] bool1-bool64 are mapped to PLC tags in the IN assembly. Bool101-bool164 are mapped to PLC tags in the OUT assembly. Bool members numbered 65-100 and 165-200 are accessible via Explicit Message only.
[v] For int, long, float, and string data:
Data members numbered 1-10 are mapped to PLC tags in the IN assembly.
Data members numbered 101-110 are mapped to PLC tags in the OUT assembly.
Data members numbered 11-100 and 111-200 are accessible via Explicit Message only.

CHAPTER 3

# Allen-Bradley AOI (Add-On Instructions) for EtherNet/IP Operation

This section provides additional instructions helpful for using a smart camera in an EtherNet/IP environment.

**Notes:**

- The camera communications protocol must be enabled for EtherNet/IP before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the camera, and information about switching camera communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate EtherNet/IP communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.

## Rockwell RSLogix 5000 AOI (Add-On Instructions) for Omron Microscan Devices

The AOI file has been created as basic instructions and as a main rung import. The AOI instructions read and write data when called. The main rung import has global tags and the AOI itself to demonstrate how the AOI is used for beginners with the RSLogix system. The AOI can be used with the EDS file.

**Note:** Examples in this section have been created using RSLogix 5000 version 20.

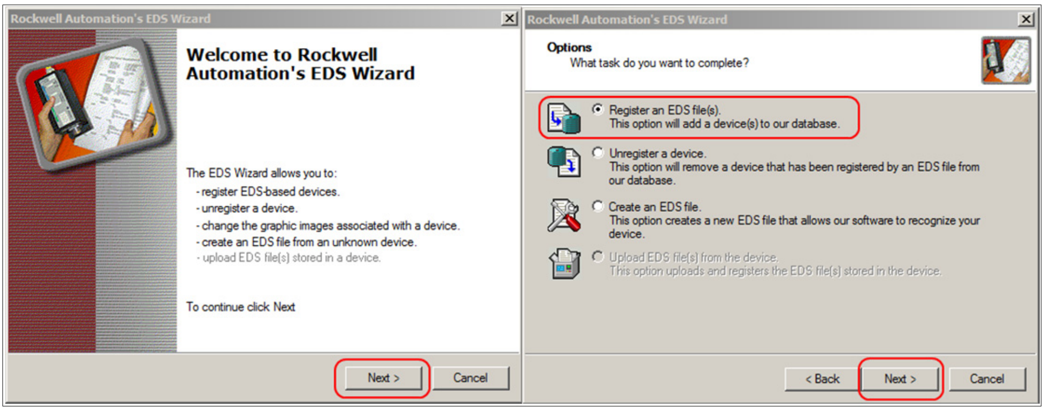
## Steps

- Install EDS File
- Import AOI File
- Test Communications and Review Data

### Install EDS File

In RSLogix 5000, select the **EDS Hardware Installation Tool** under the main menu **Tools**.

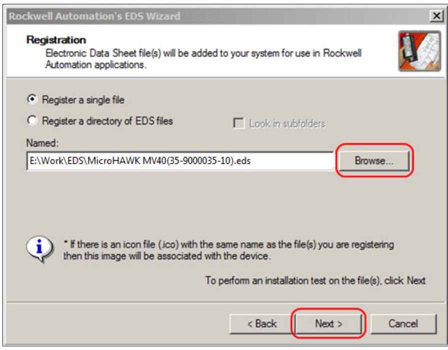
Click the **Next** button. Make sure the **Register an EDS Files(s)** radio button is selected.

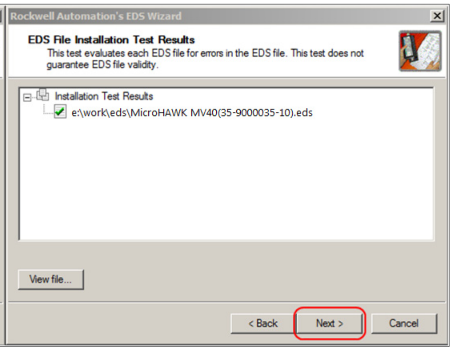


Click the **Browse** button to locate the EDS file (C:\Microscan\Vscape\Firmware\eds\). Once the EDS file is located and selected, click the **Next** button.

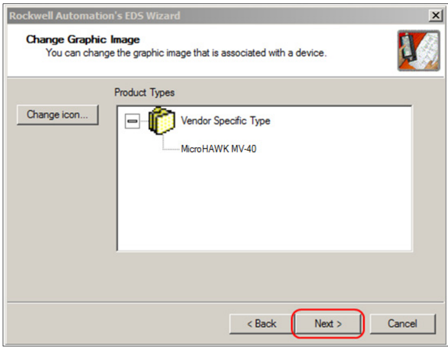
The MicroHAWK's EDS file is: **MicroHAWK (35-9000035-10).eds**

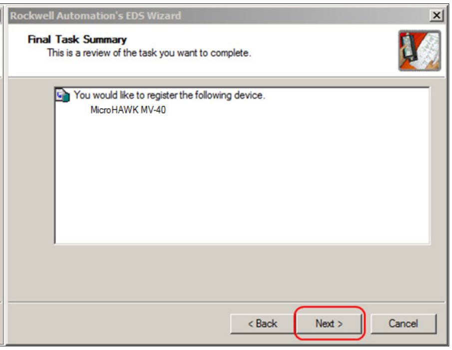
The HAWK MV-4000's EDS file is: **HAWK_MV4000(8ABS-LFFA-LPPP).eds**



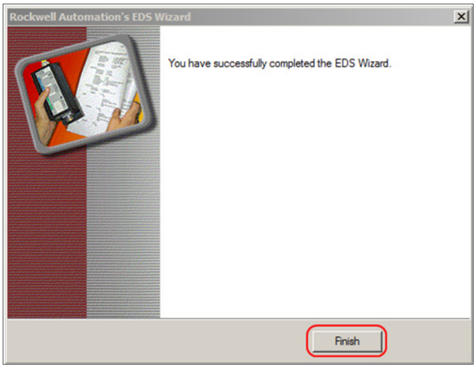




Click the **Next** button for the image. Click the **Next** button for the summary.

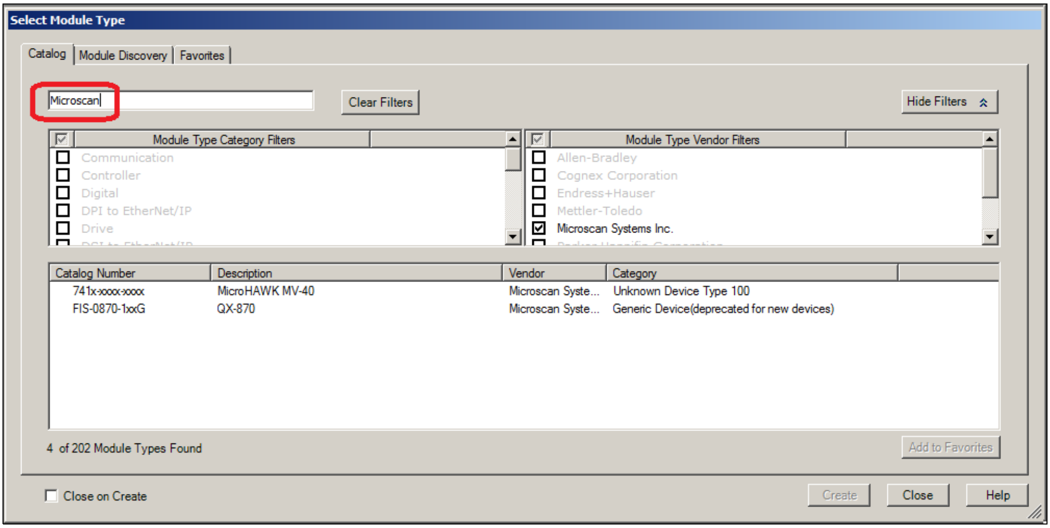


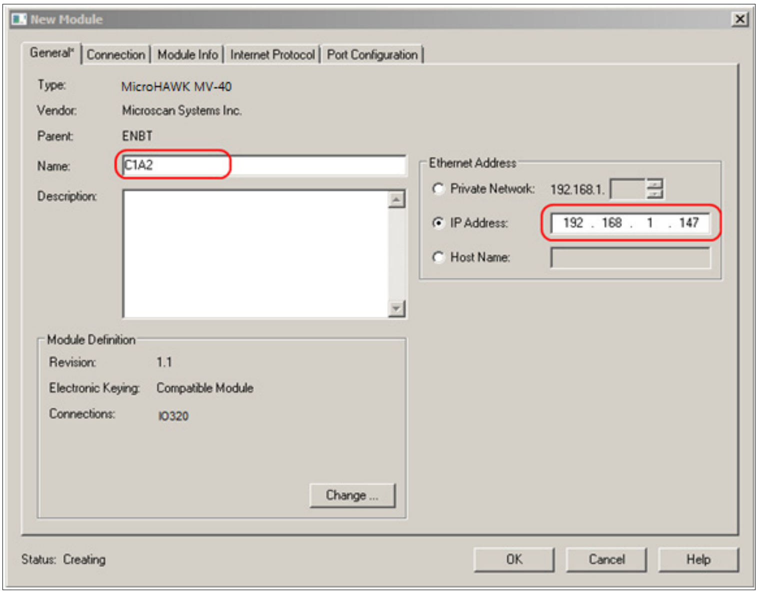

Click the **Finish** button to complete the EDS installation.



Right-click the Ethernet node on the left pane and select **New Module**. Type **Microscan** in the filter box to list the device. Select the device from the newly added EDS file. Double-click the device or select and click the Create button to add to the project.

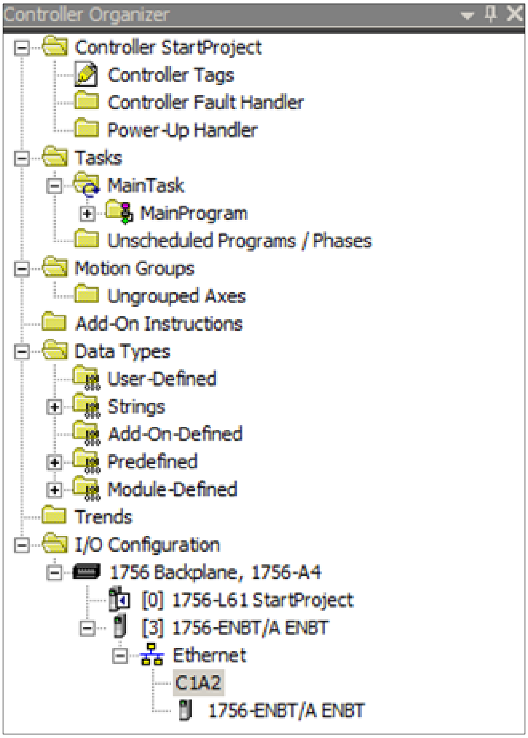



Enter the name for your device and the IP address, then click **OK**.



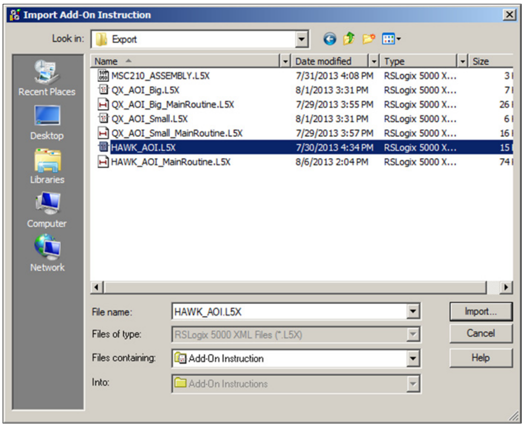

If the camera has multiple assembly sizes, the **Change** button allows you to select the other assembly formats.

Click the **Close** button on the module selection dialog to continue. Now the device has been added to the project and will be visible in the tree view under the **Ethernet** node.



## Import AOI File

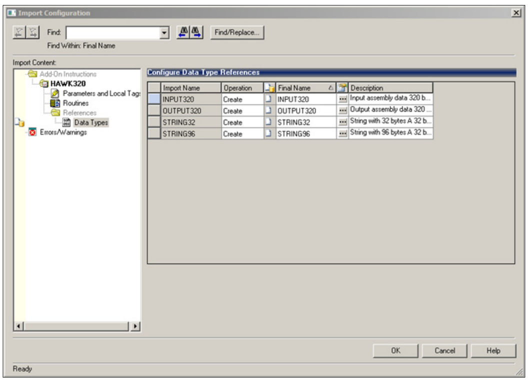
Right-click the **Add-On Instructions** node in the tree view in the left pane and select **Import Add-On Instruction**. Locate the **L5X file** (C:\Microscan\Vscape\Firmware\aoi\) and click the **Import** button.



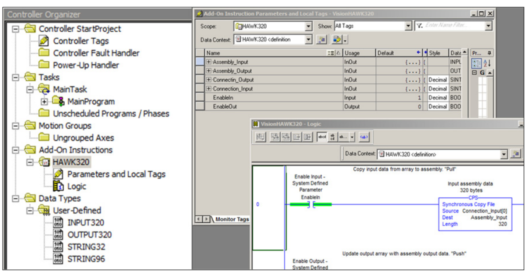

The MicroHAWK's AOI file name is: **MicroHAWK_MV_AOI.L5X**.

The HAWK MV-4000's AOI file name is: **HAWK_MV4000_AOI.L5X**.

The **Import Configuration** dialog will prompt you for information regarding the AOI file. Select the **Data Types** to view the new tags and their attributes. Click **OK** when ready to continue.




New tags and logic will now be added to the project.

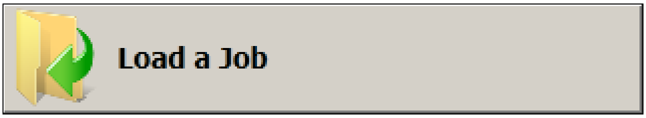


## Test Communications and Review Data

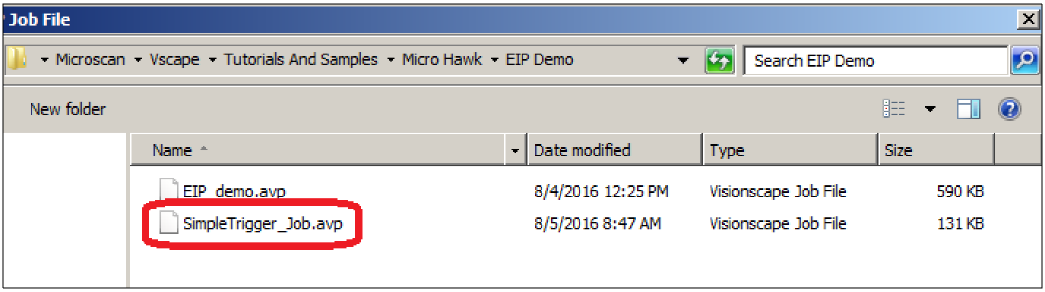
After the reader module has been installed, you can start with a basic ladder logic program to test the data to and from the device. Two steps are involved to test the communication: (1.) Download an autovision job; (2.) Add logic to the PLC ladder logic.

Start by opening up **AutoVISION** and select the camera.

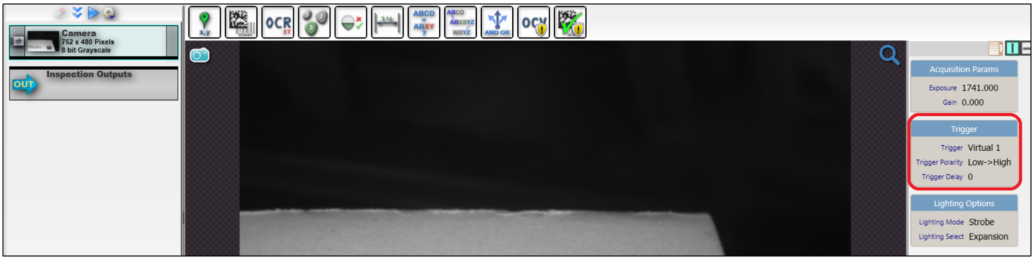
Select **Load a Job**.



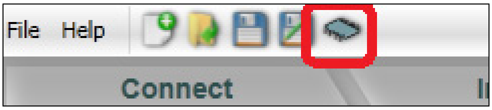

Navigate to **Omron Microscan > Vscape > Tutorials and Samples > MicroHAWK > EIP Demo > SimpleTrigger_Job.avp**.



This job has no tools loaded and is a simple job that allows the PLC to trigger the camera. You can also set another Trigger action in the Trigger panel on the right side of the screen.

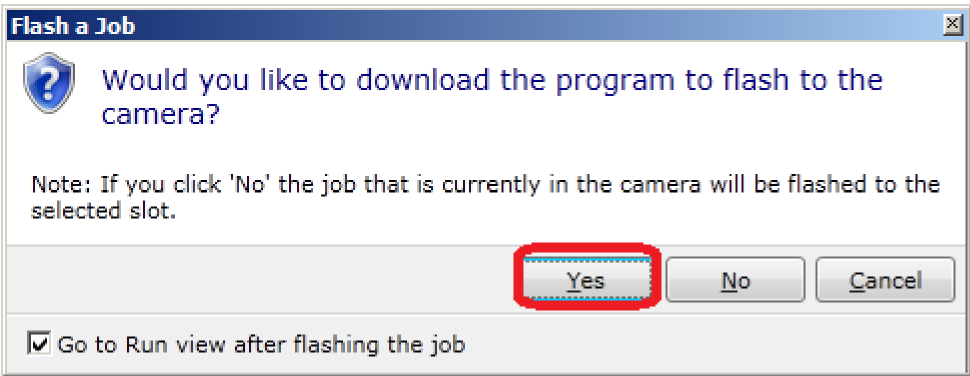



Now download the job to the camera by selecting the Save Job to Camera icon and selecting **New Slot**.

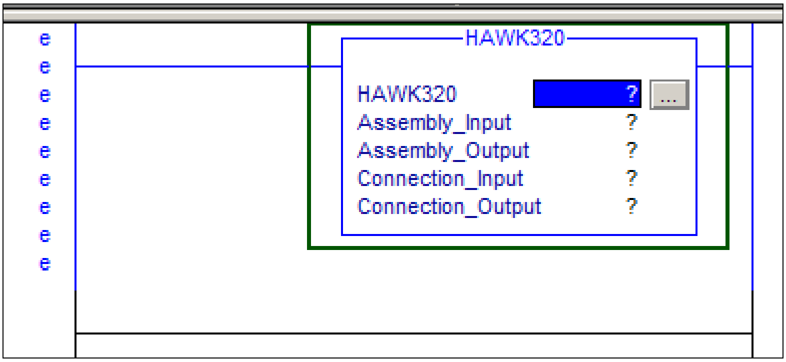




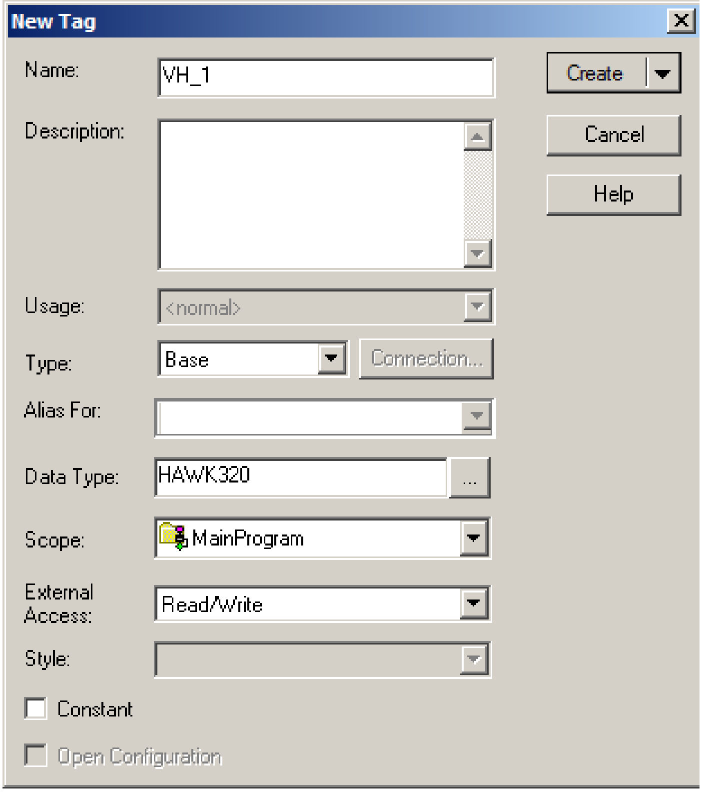
Select **Yes** to downloading the program to flash to the camera.



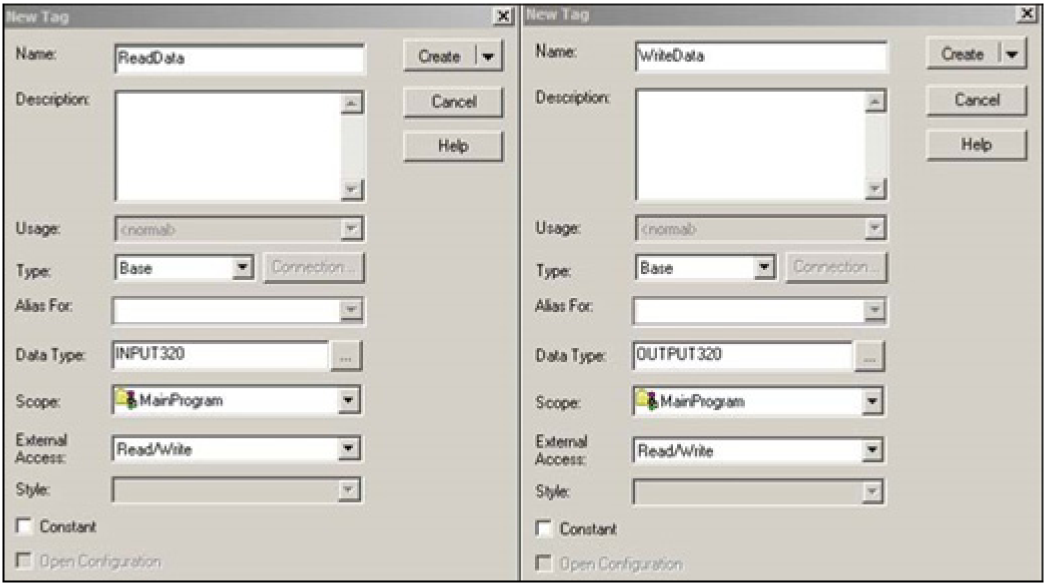
Then add the logic to the PLC. Do this by opening the **MainRoutine** editor and create an **Add-On-Instructions**.



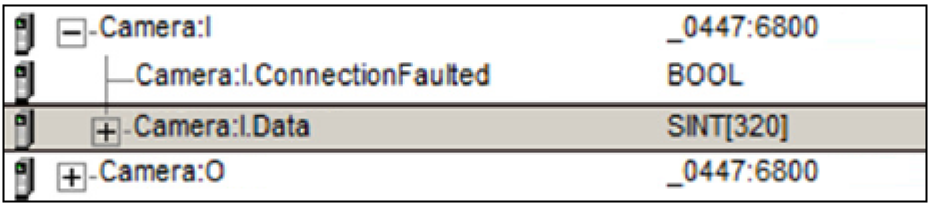

Right-click in the first element and create a **New Tag**.



Right-click the second element and create a new tag for the **Assembly_Input**. It will default to the **AOI INPUT320** data type. Right-click the third element and create a new tag for the **Assembly_Output**. It will default to the **AOI OUTPUT320** data type.

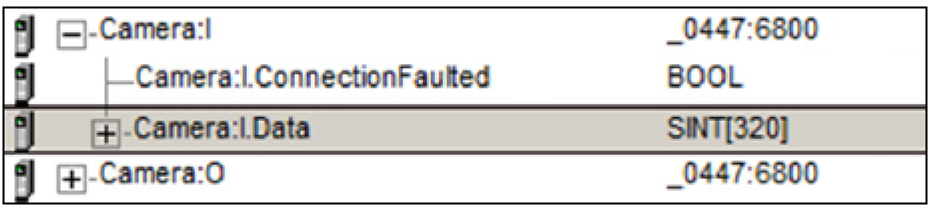



Double-click the fourth element and click the down arrow in the combo box to link the **Connection_Input** to the reader input data. The link should be the **[reader name]:I.Data**.

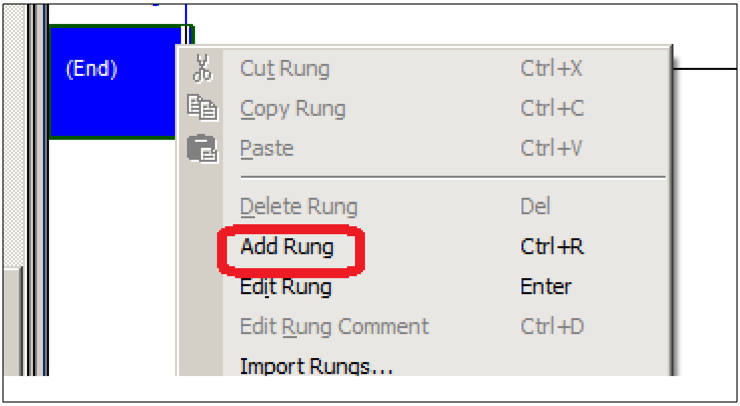



**Note:** Do not connect to the **ConnectionFault** item.

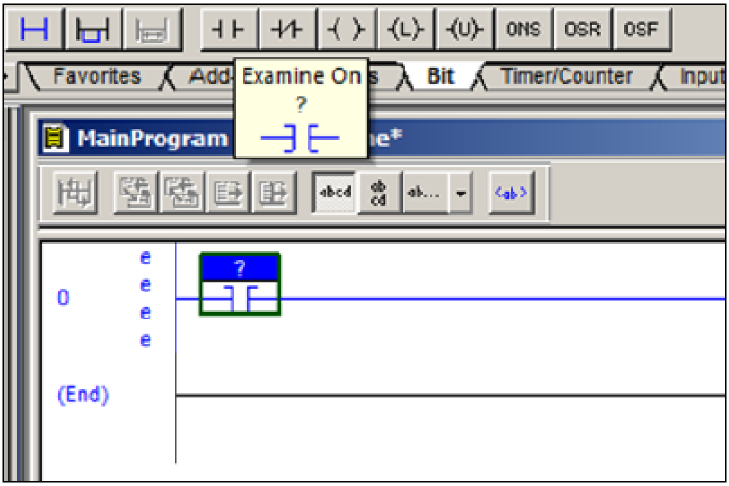
Double-click the fifth element and click the down arrow in the combo box to link the **Connection_Output** to the reader output data. The link should be the **[reader name]:O.Data**.



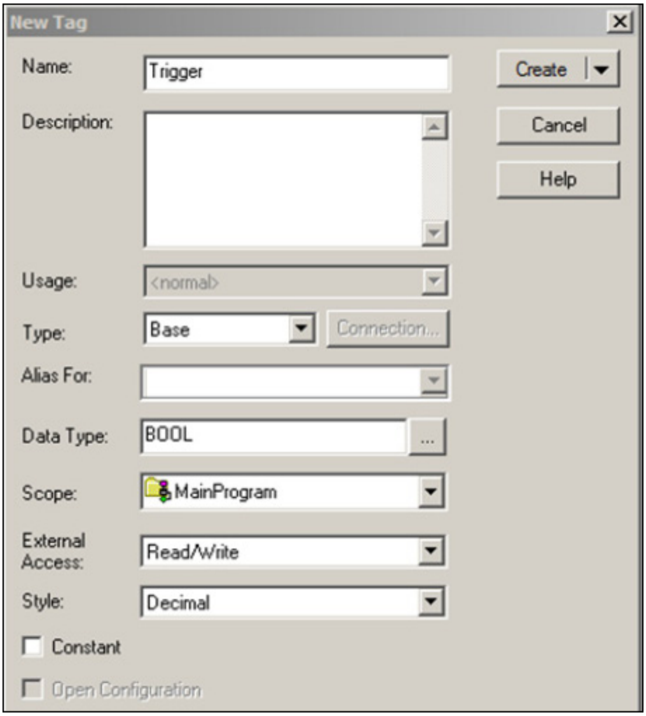
Add a new rung by right clicking and selecting **Add Rung**.



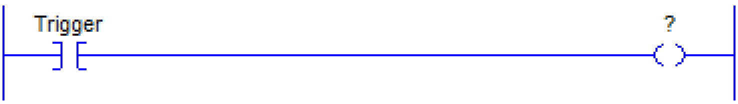

Create an **Examine On** trigger tag.



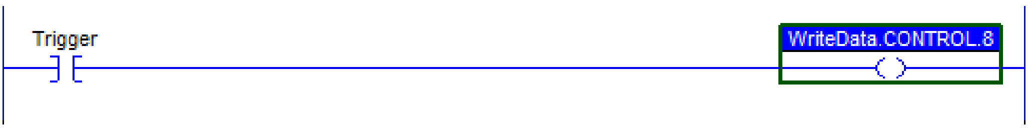
Right-click and create a **Boolean Trigger**.



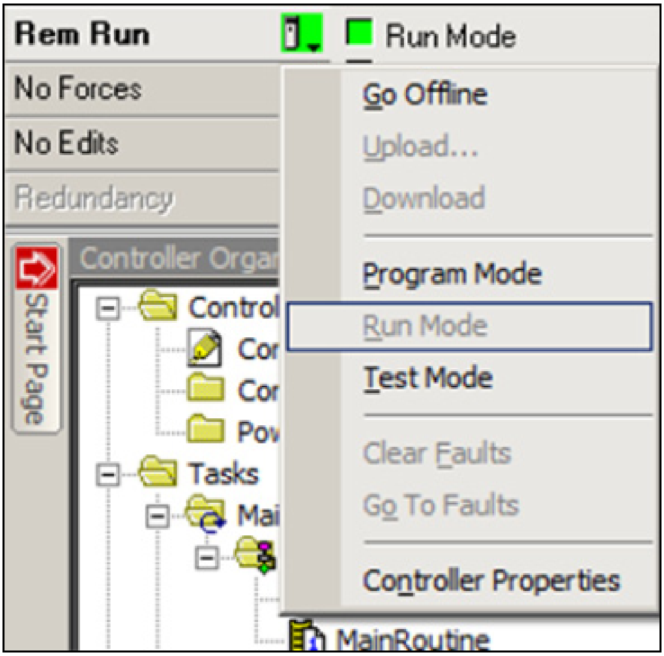

Add an **Output Energized** Tag to the end of the rung.




Assign the **Output Energized** tag to the **WriteData.CONTROL.8** bit

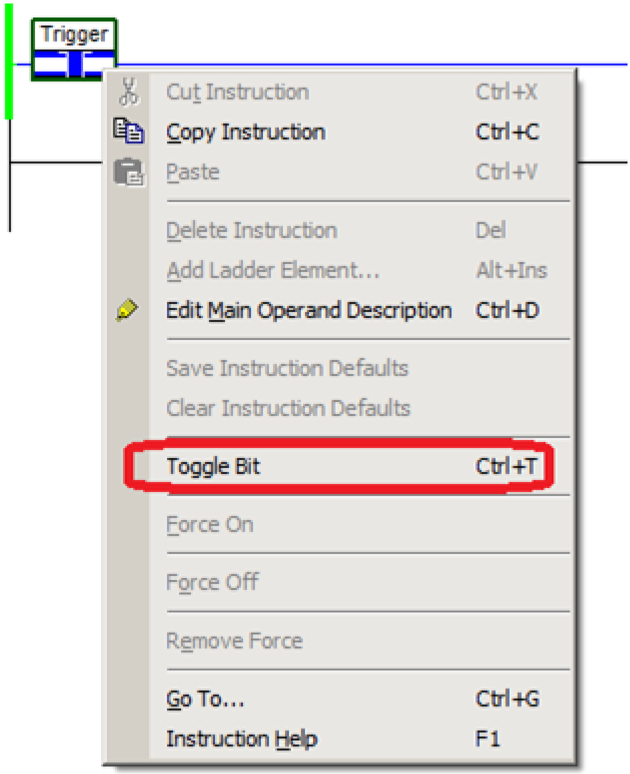


Now download the program to the PLC. Once the program has downloaded, set the PLC to **Run Mode**.




**Note:** Be sure the path to the PLC has been set in the project so that communications to the PLC can be established.

Toggle the bit **Trigger** by right-clicking and selecting **Toggle Bit** or by pressing **Ctrl + T**.




This action will set the **Trigger** tag high causing the trigger output bit to go high in the Output Assembly. The camera will trigger on each rising edge (ie when the **Trigger** tag is set to 1). This will confirm that communication is established between the PLC and the camera.

4

CHAPTER 4

# Allen-Bradley PLC Setup via EDS for EtherNet/IP Operation

This section describes how to use an EDS file to set up an Allen-Bradley PLC for EtherNet/IP operation.

**Notes:**

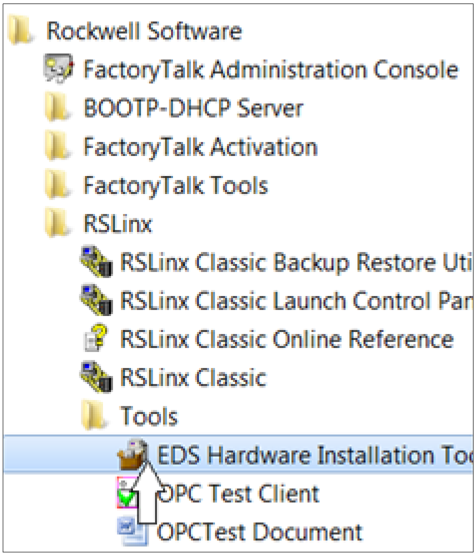
- The camera communications protocol must be enabled for EtherNet/IP before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the MicroHAWK, and information about switching MicroHAWK communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate EtherNet/IP communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.

# AB Rockwell RSLogix 5000 v20 PLC Integration with EDS

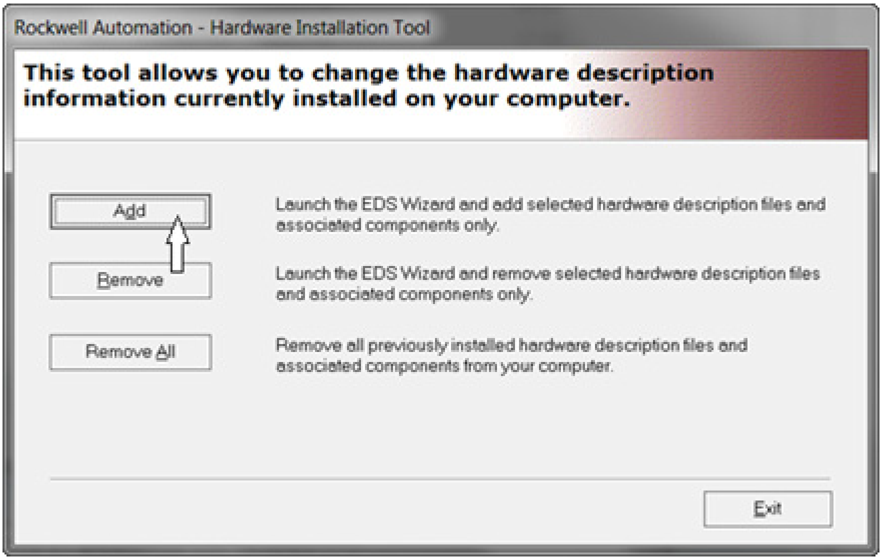
This section was created and run on the following Allen Bradley/Rockwell components:

- RSLogix 5000 Version 20.00.00 (CPR 9 SR 5)
- 756-L61 ControlLogix5561 Controller, firmware rev 20.11
- 1756-ENBT/A EtherNet/IP interface card, firmware rev 4.1

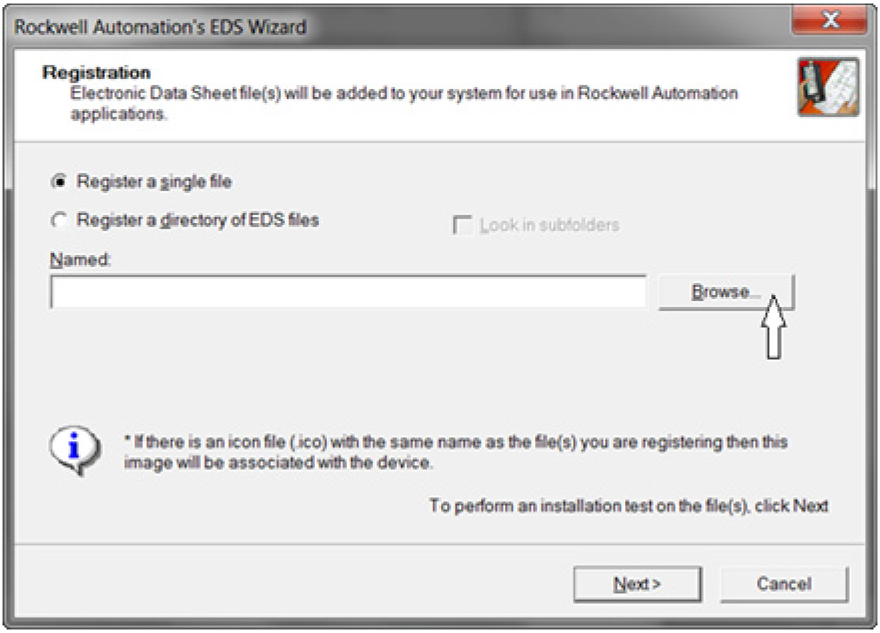
Run the Rockwell **EDS Hardware Installation Tool**.



Select **Add**.



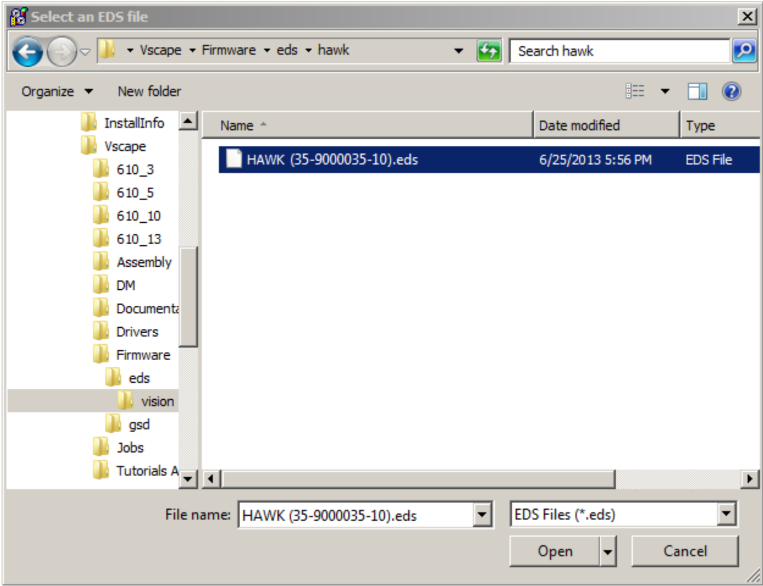

Select **Browse**.



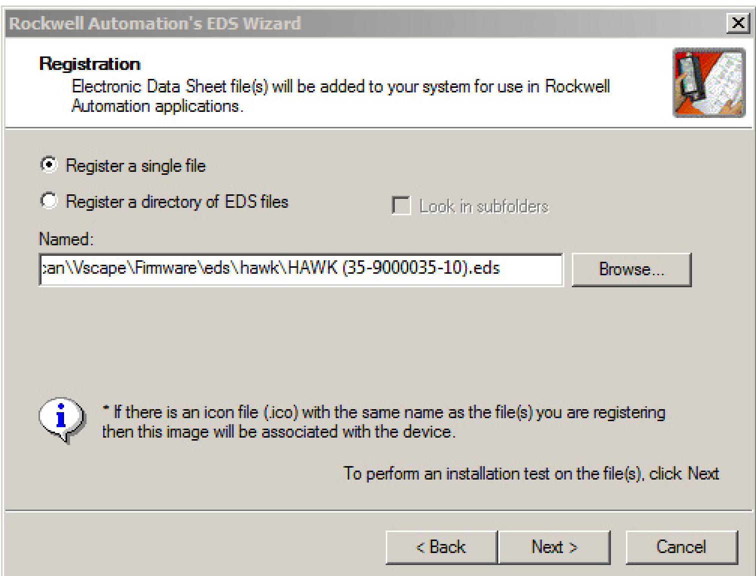
Navigate to the EDS file, then **Open** it. The default install location is:

**For MicroHAWK: C:\Microscan\Vscape\Firmware\eds\MicroHAWK**

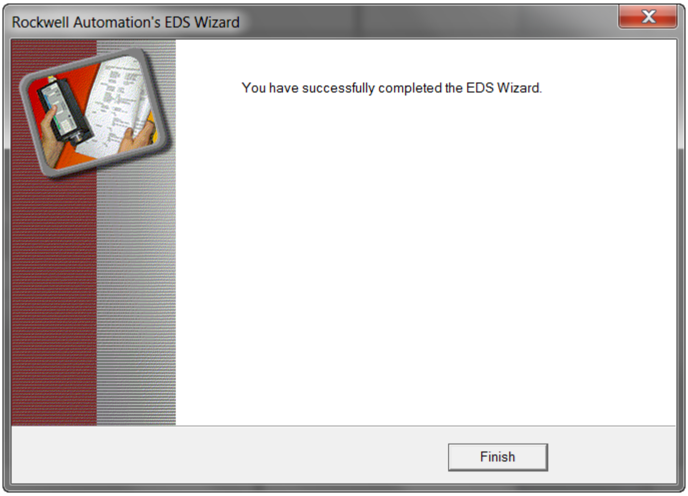
**For HAWK MV-4000: C:\Microscan\Vscape\Firmware\eds\HAWK MV4000**



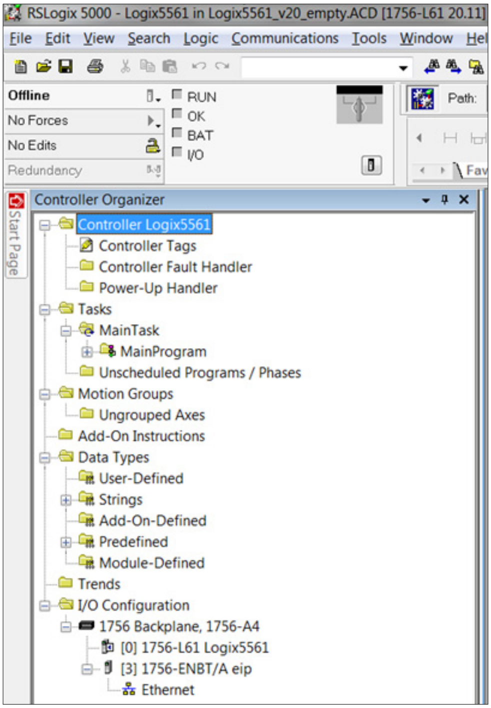

Keep clicking **Next** until the **Finish** button is displayed.



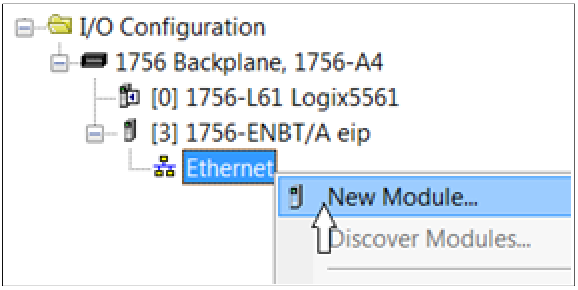
Click **Finish**.



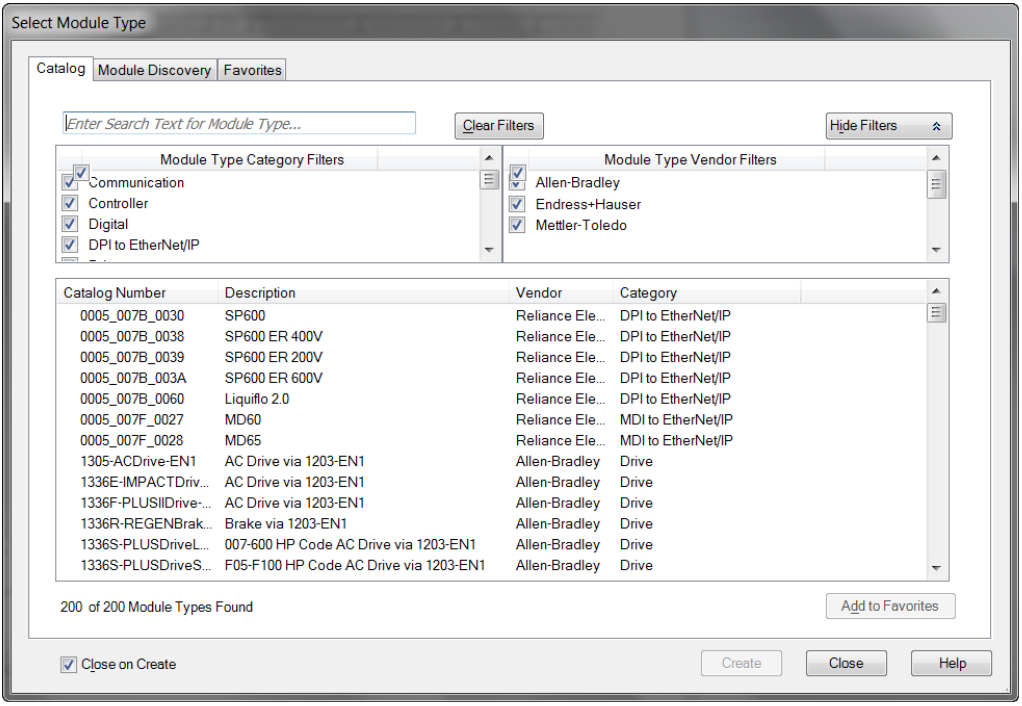

Open RSLogix 5000 v20 and create the **I/O Configuration** for the base system, including the system's Ethernet interface.



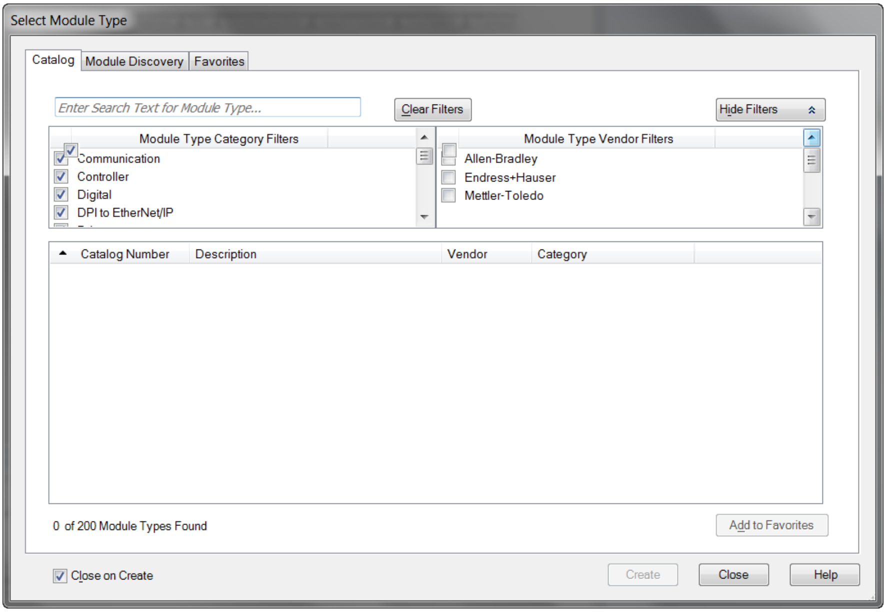
Right-click **Ethernet** and select **New Module**.



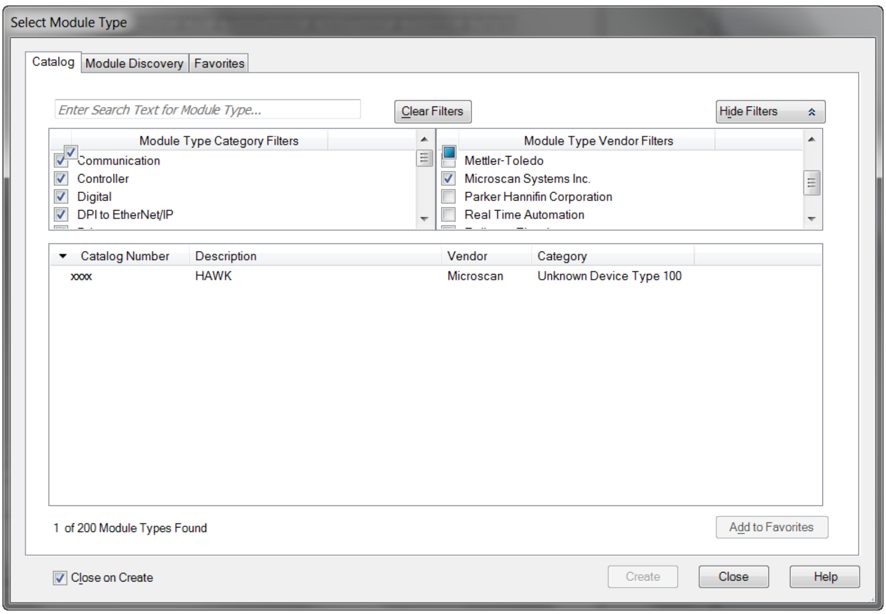

The **Select Module Type** dialog will be displayed.



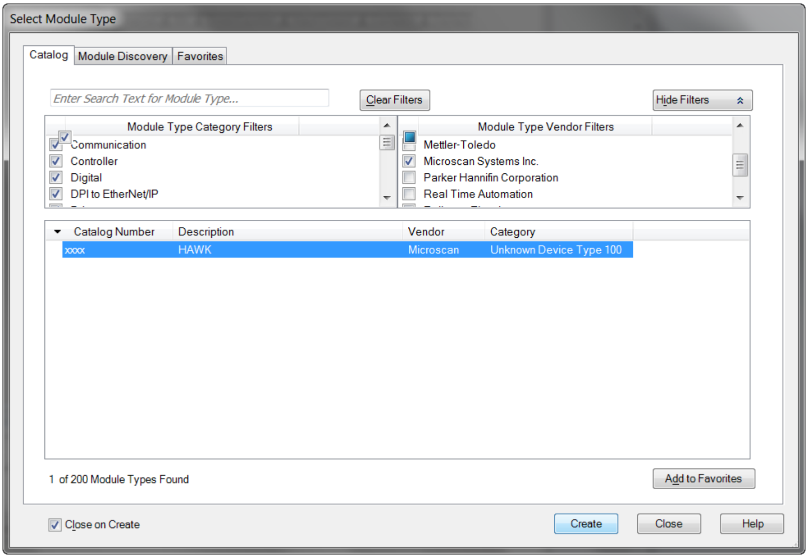
Clear the **Module Type Vendor Filters**.



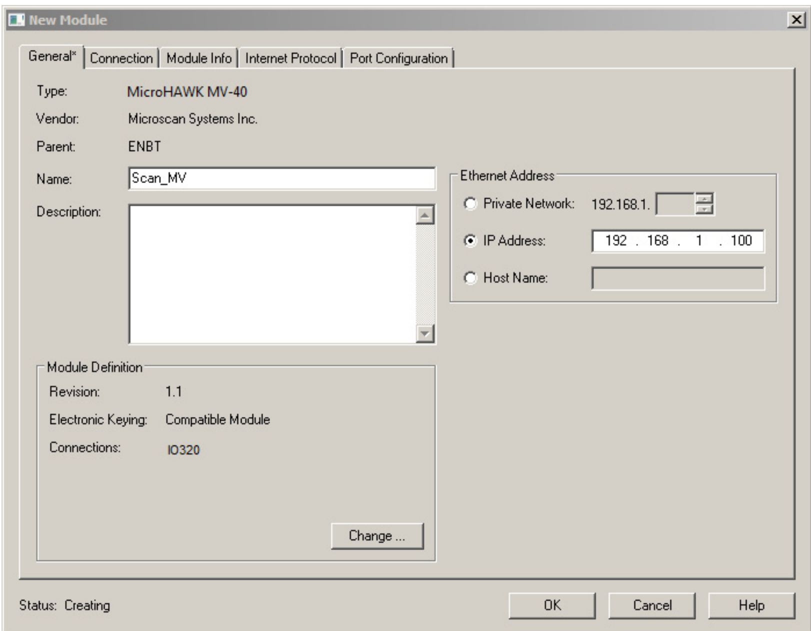

Scroll down the Module Type Vendor Filters until **Omron Microscan** comes into view, then select Microscan.



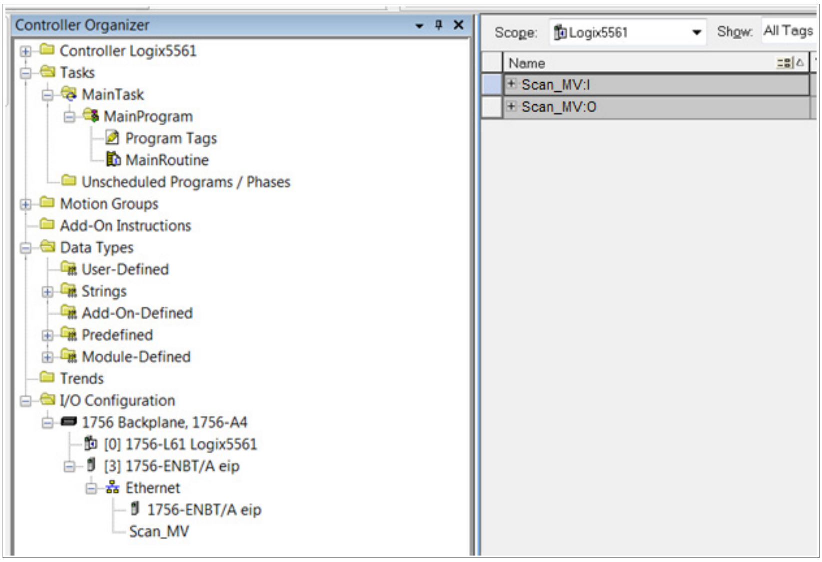
Click the required camera and click **Create**.



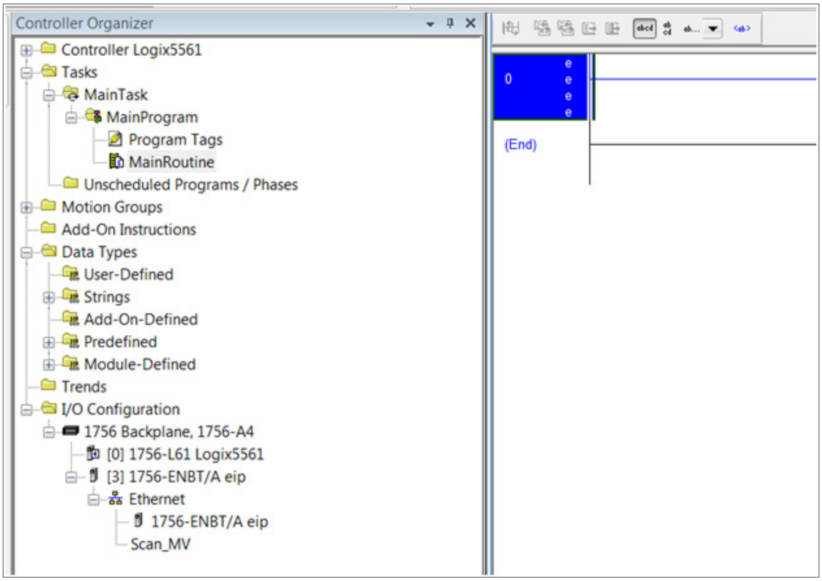

The **New Module** dialog is displayed. Type a unique name for this camera and its IP address.



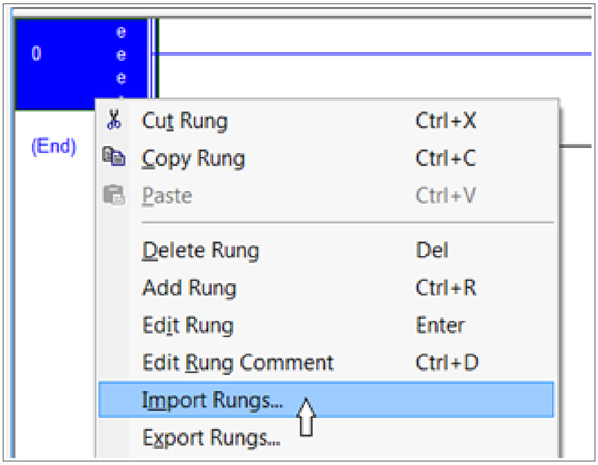
Click **OK**, verify the camera was added to the Ethernet network, then open the controller tags to verify that **:I** and **:O** tag sets were created.




Open the **Main Routine**.

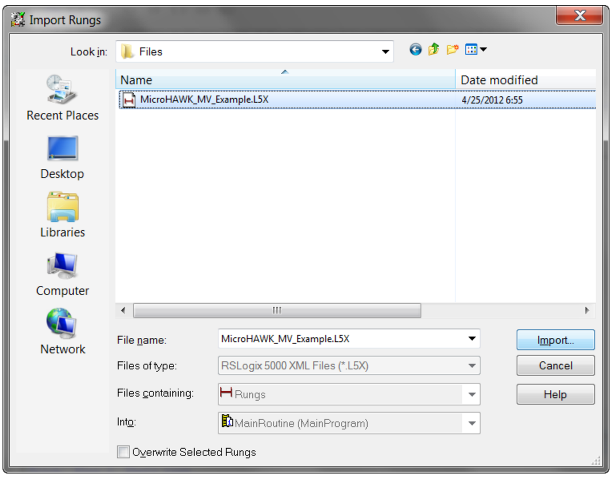


Right-click rung **0**, and select **Import Rungs**.

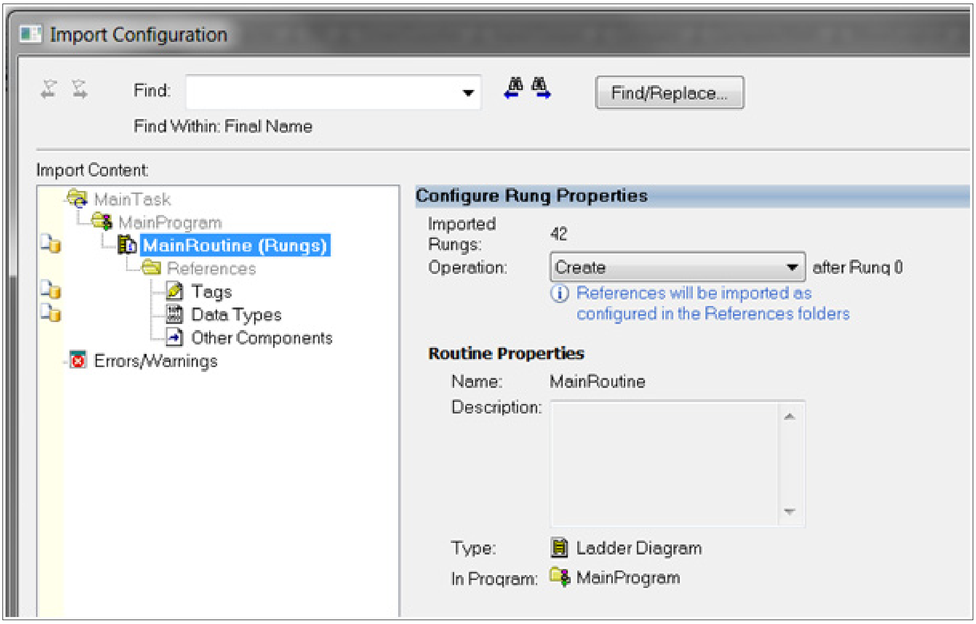



Navigate to the **MicroHAWK_MV_Example.L5X** file and select **Import**. The default install directory is: **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\EIP Demo**.

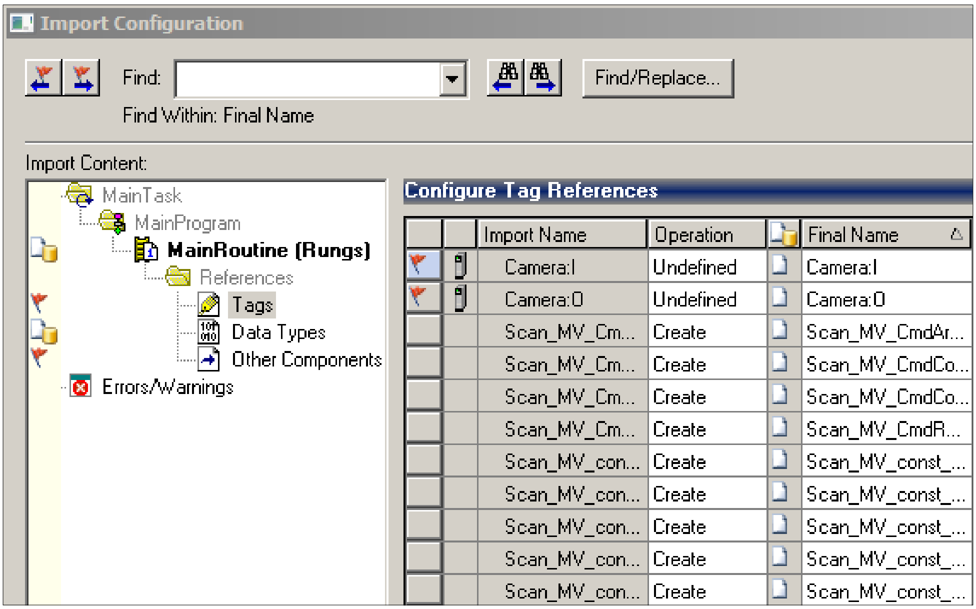
**Note:** If you are using a HAWK MV-4000 Smart Camera, you can use this example routine as well.



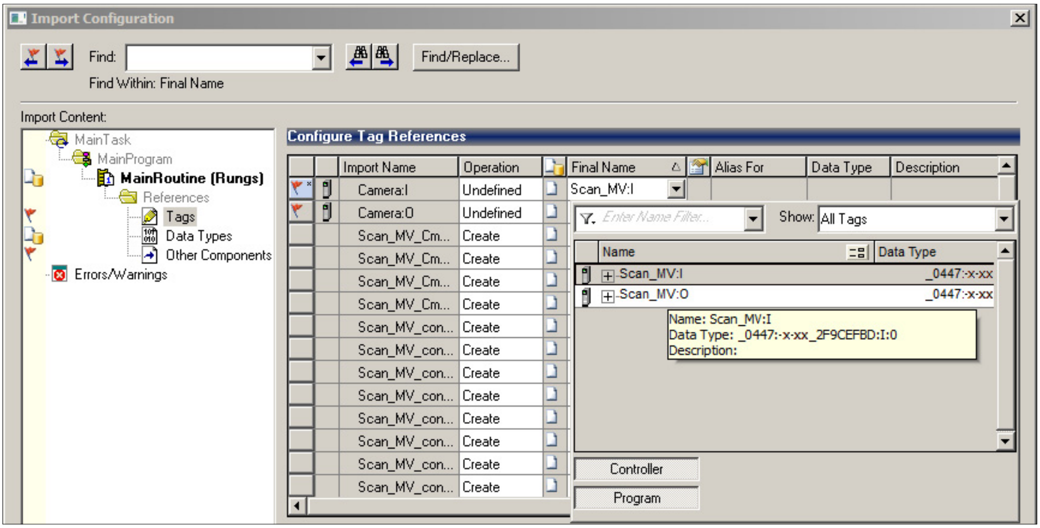
The **Import Configuration** dialog will be displayed.



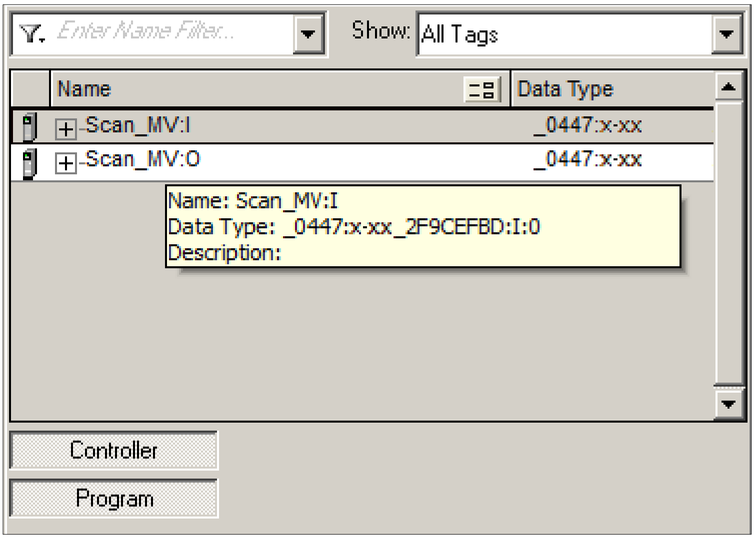

Select **Tags**.



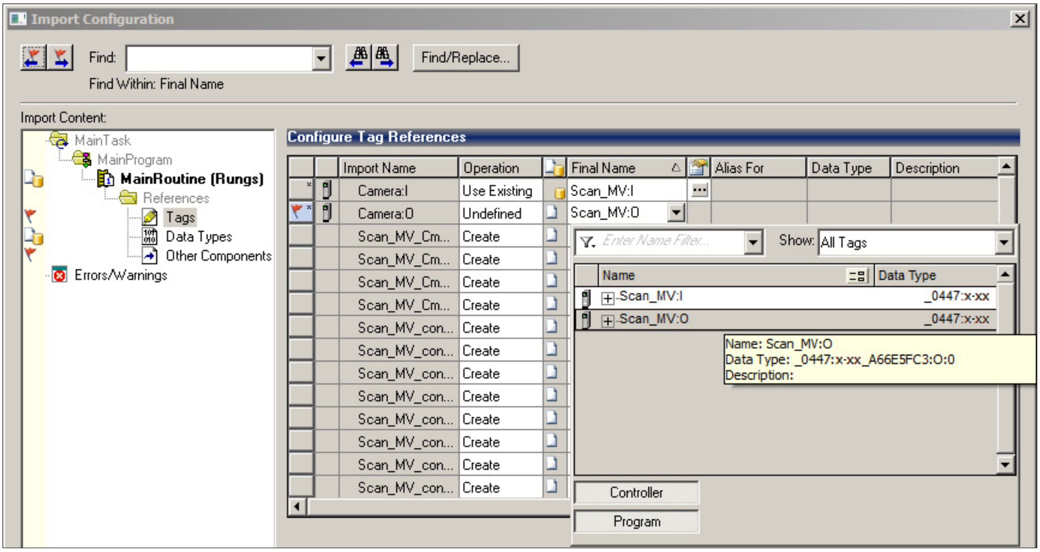
In the **Final Name** column, click **Camera:I**, then click the down arrow that appears on the right.



Double-click the camera name input tag assigned earlier.

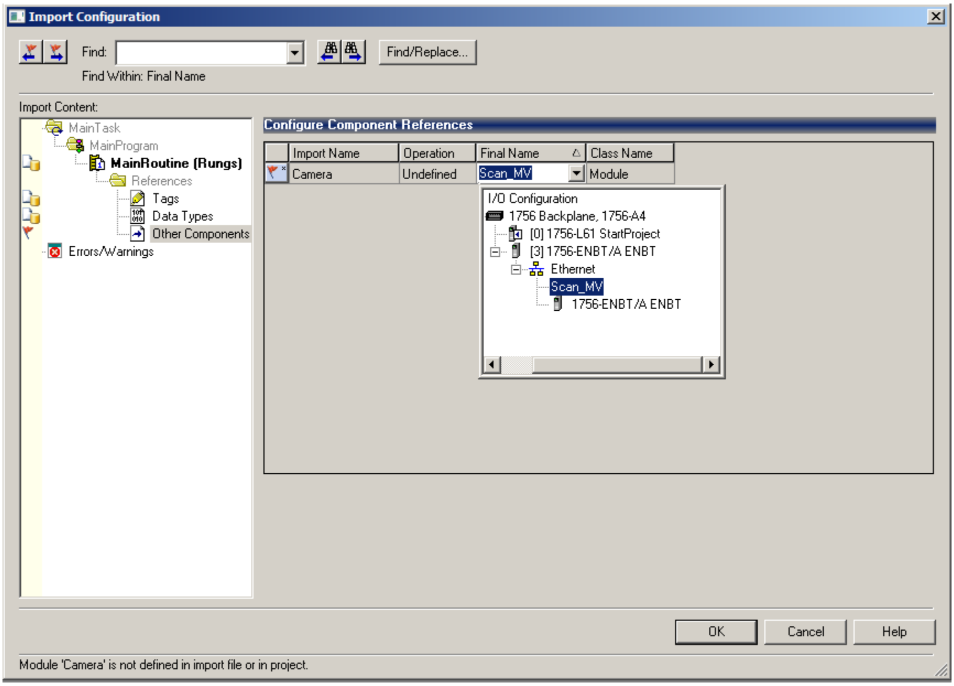


In the **Final Name** column, click **Camera:O**, then click the down arrow that appears on the right.

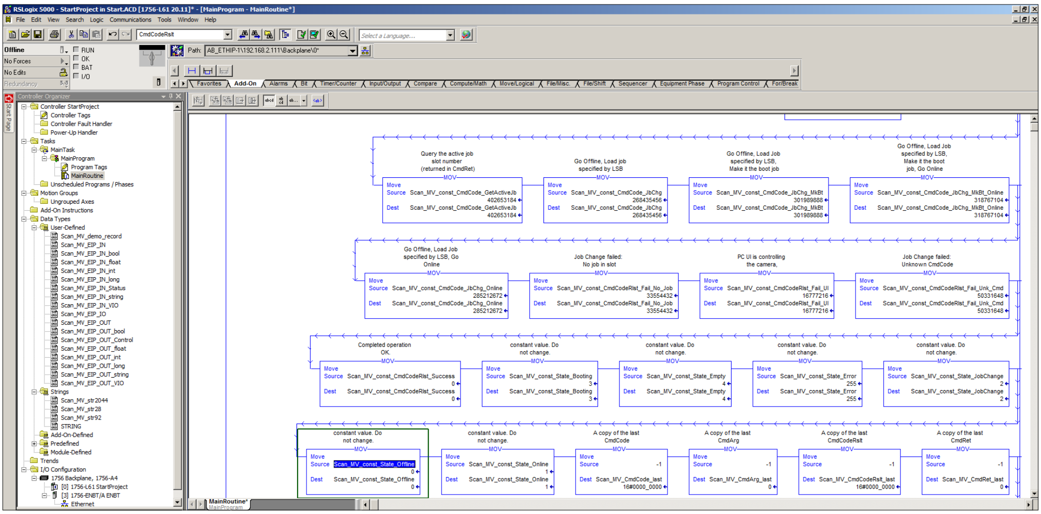


Double-click the camera name output tag assigned earlier.

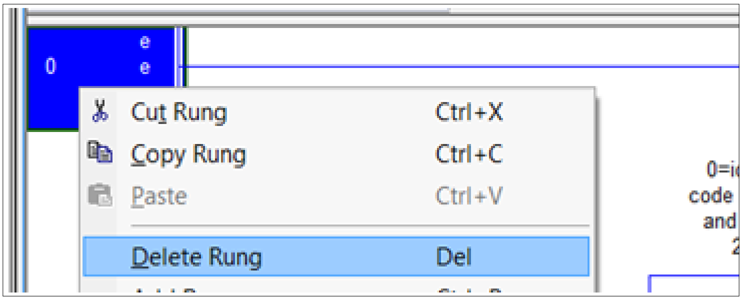
Click the **Other Components** icon in the tree view to select the **Component References**. Select the camera in the **Final Name** column. Click **OK** to complete the import.



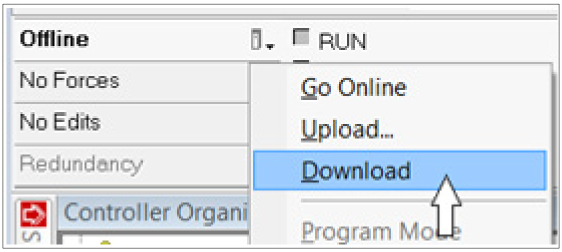

Click **OK** and the **Main Routine** and **User-Defined tags** will be populated.



Delete any empty rungs (check rung **0**).

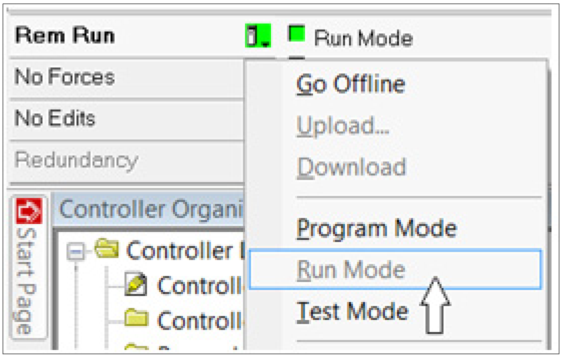



Download the project to the PLC.

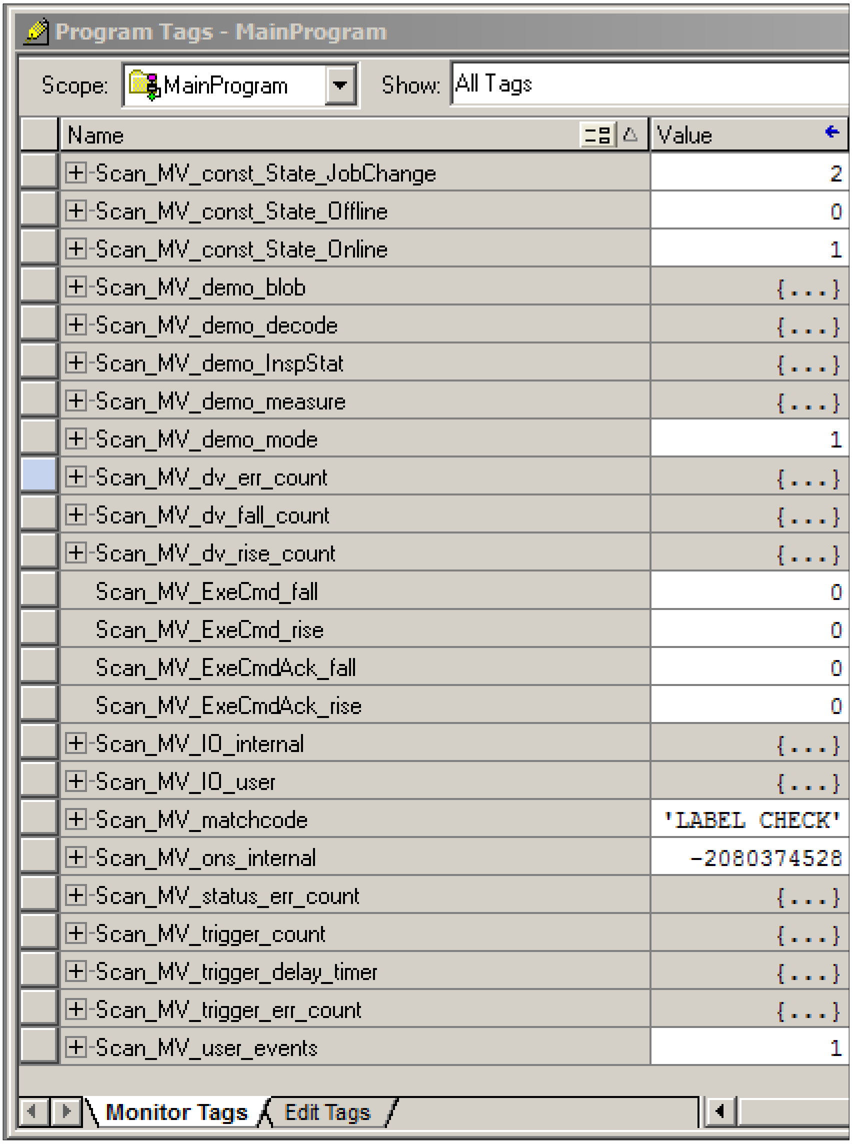



**Note:** Be sure the path to the PLC has been set in the project so that communications to the PLC can be established.


Put the PLC in **Run Mode**.



Open the **Program Tags** window and select **Monitor Tags**.



Expand **Scan_MV_IO_user** so that the **Echo** in the **.IN.Status** and **.OUT.Control** structures is visible.

| Name | Value |
|---|---|
| Scan_MV_IO_user.IN.Status.reserved15 | 0 |
| Scan_MV_IO_user.IN.Status.Echo | 0 |
| Scan_MV_IO_user.IN.Status.CmdCodeRslt | 16#0000_0000 |
| Scan_MV_IO_user.IN.Status.CmdRet | 0 |
| Scan_MV_IO_user.IN.Status.reserved96_103 | 0 |
| Scan_MV_IO_user.IN.Status.reserved104_111 | 0 |
| Scan_MV_IO_user.IN.Status.State | 1 |
| Scan_MV_IO_user.IN.Status.reserved120_127 | 0 |
| Scan_MV_IO_user.IN.VIO | {...} |
| Scan_MV_IO_user.IN.bool | {...} |
| Scan_MV_IO_user.IN.int | {...} |
| Scan_MV_IO_user.IN.long | {...} |
| Scan_MV_IO_user.IN.float | {...} |
| Scan_MV_IO_user.IN.string | {...} |
| Scan_MV_IO_user.OUT | {...} |
| Scan_MV_IO_user.OUT.Control | {...} |
| Scan_MV_IO_user.OUT.Control.GoOnline | 0 |
| Scan_MV_IO_user.OUT.Control.GoOffline | 0 |
| Scan_MV_IO_user.OUT.Control.reserved2 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved3 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetError | 0 |
| Scan_MV_IO_user.OUT.Control.ResetCount | 0 |
| Scan_MV_IO_user.OUT.Control.reserved6 | 0 |
| Scan_MV_IO_user.OUT.Control.ExeCmd | 0 |
| Scan_MV_IO_user.OUT.Control.Trigger | 0 |
| Scan_MV_IO_user.OUT.Control.reserved9 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved10 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetDataValid | 0 |
| Scan_MV_IO_user.OUT.Control.reserved12 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved13 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved14 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved15 | 0 |
| Scan_MV_IO_user.OUT.Control.Echo | 0 |
| Scan_MV_IO_user.OUT.Control.CmdCode | 16#0000_0000 |
| Scan_MV_IO_user.OUT.Control.CmdArg | 0 |

Change **.OUT.Control.Echo** to non-zero.

| | |
|---|---|
| Scan_MV_IO_user.OUT.Control.reserved15 | 0 |
| Scan_MV_IO_user.OUT.Control.Echo | 4321 |
| Scan_MV_IO_user.OUT.Control.CmdCode | 16#0000_0000 |

Verify that **Scan_MV_IO_user.IO.IN.Status.Echo** is the same value as the **.OUT.Control.Echo**.

| | |
|---|---|
| Scan_MV_IO_user.IN.Status.reserved15 | 0 |
| Scan_MV_IO_user.IN.Status.Echo | 4321 |
| Scan_MV_IO_user.IN.Status.CmdCodeRslt | 16#0000_0000 |

This confirms that the PLC and camera have successful two-way communication.

The demo code expects a demo vision job to be loaded on the camera, which populates the following input tags (camera to PLC) with vision tool results:

- .IN.bool.bool1, bool2, and bool3
- .IN.long.long1
- .IN.float.float1
- .IN.string.string1

The demo code will operate the **Control** and **Status** signals of the camera regardless of the vision job that is loaded. For a more detailed overview of the demo code and vision job, see **Allen-Bradley PLC Setup via Generic Ethernet Module for EtherNet/IP Operation**.

To send a trigger to the camera, scroll to **Scan_MV_IO_user.Control.Trigger**.

| | |
|---|---|
| Scan_MV_IO_user.OUT.Control.ExeCmd | 0 |
| Scan_MV_IO_user.OUT.Control.Trigger | 0 |
| Scan_MV_IO_user.OUT.Control.reserved9 | 0 |

Set the Trigger to **1**. This causes the demo code to trigger the camera, process the new inspection data, record the results in the **Scan_MV_demo_xxxx** tags, and clear the **DataValid** status signal.

The **Trigger** control changes to **0** when the camera is triggered. All processing is done when the counter **Scan_MV_dv_fall_count** increments, along with the pass/fail counters in the **Scan_MV_demo_xxxx** tags.

| Name | Value |
|---|---|
| Scan_MV_demo_blob | {...} |
| Scan_MV_demo_blob.pass_count | {...} |
| Scan_MV_demo_blob.fail_count | {...} |
| Scan_MV_demo_blob.fail_count.PRE | 0 |
| Scan_MV_demo_blob.fail_count.ACC | 30 |
| Scan_MV_demo_blob.fail_count.CU | 0 |
| Scan_MV_demo_blob.fail_count.CD | 0 |
| Scan_MV_demo_blob.fail_count.DN | 1 |
| Scan_MV_demo_blob.fail_count.OV | 0 |
| Scan_MV_demo_blob.fail_count.UN | 0 |
| Scan_MV_demo_blob.bool | 0 |
| Scan_MV_demo_blob.long | 6 |
| Scan_MV_demo_blob.long_max | 6 |
| Scan_MV_demo_blob.long_min | 4 |
| Scan_MV_demo_blob.float | 0.0 |
| Scan_MV_demo_blob.float_min | 0.0 |
| Scan_MV_demo_blob.float_max | 0.0 |
| Scan_MV_demo_blob.string | '' |
| Scan_MV_demo_decode | {...} |
| Scan_MV_demo_InspStat | {...} |
| Scan_MV_demo_measure | {...} |
| Scan_MV_demo_mode | 1 |
| Scan_MV_dv_err_count | {...} |
| Scan_MV_dv_fall_count | {...} |
| Scan_MV_dv_fall_count.PRE | 0 |
| Scan_MV_dv_fall_count.ACC | 59 |
| Scan_MV_dv_fall_count.CU | 0 |

CHAPTER 5

# Allen-Bradley PLC Setup via Generic Ethernet Module for EtherNet/IP Operation

This section describes how to use the Generic Ethernet Module to set up an Allen-Bradley PLC for EtherNet/IP operation.

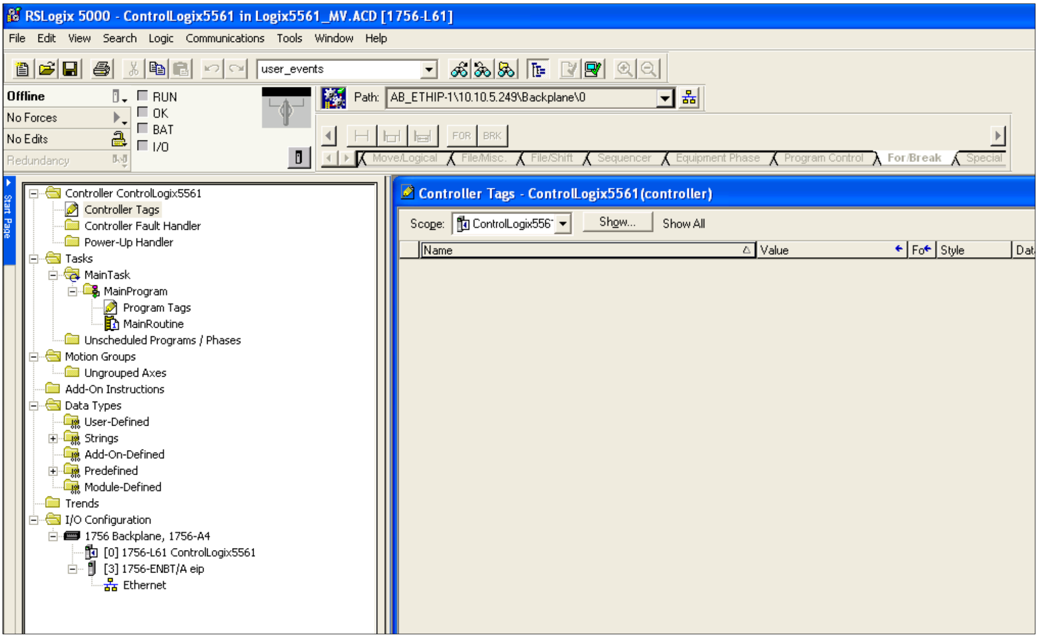
**Notes:**

- The camera communications protocol must be enabled for EtherNet/IP before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the camera, and information about switching camera communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate EtherNet/IP communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.

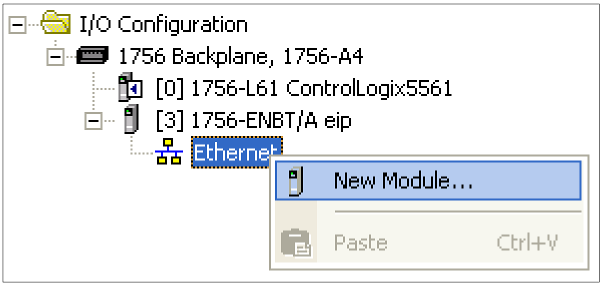
# Integrating the Camera into a PLC Environment

This section assumes you are using an Allen Bradley PLC with Rockwell RSLogix 5000 v16 or newer. RSLogix v19 and v20 may look slightly different than the screen shots shown, but the integration process is similar.

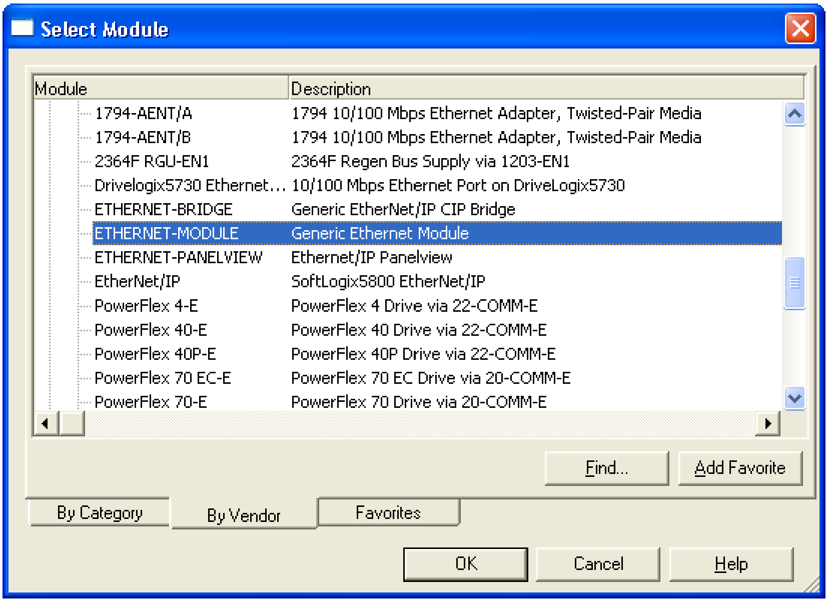
Create the **I/O Configuration** for the base system, including the system's Ethernet interface.




Add the camera by right-clicking the Ethernet interface and selecting **New Module**.



Select **ETHERNET-MODULE – Generic Ethernet Module**, and click **OK**.



Configure the following fields:

**Name:** A useful name to rememberfor the camera. The example here is **Scan_MV**.

**IP Address:** The IP Address of the camera

**Comm Format:** “Data – DINT”

**Input, Assembly Instance:** 102

**Input, Size:** 80

**Output, Assembly Instance:** 114

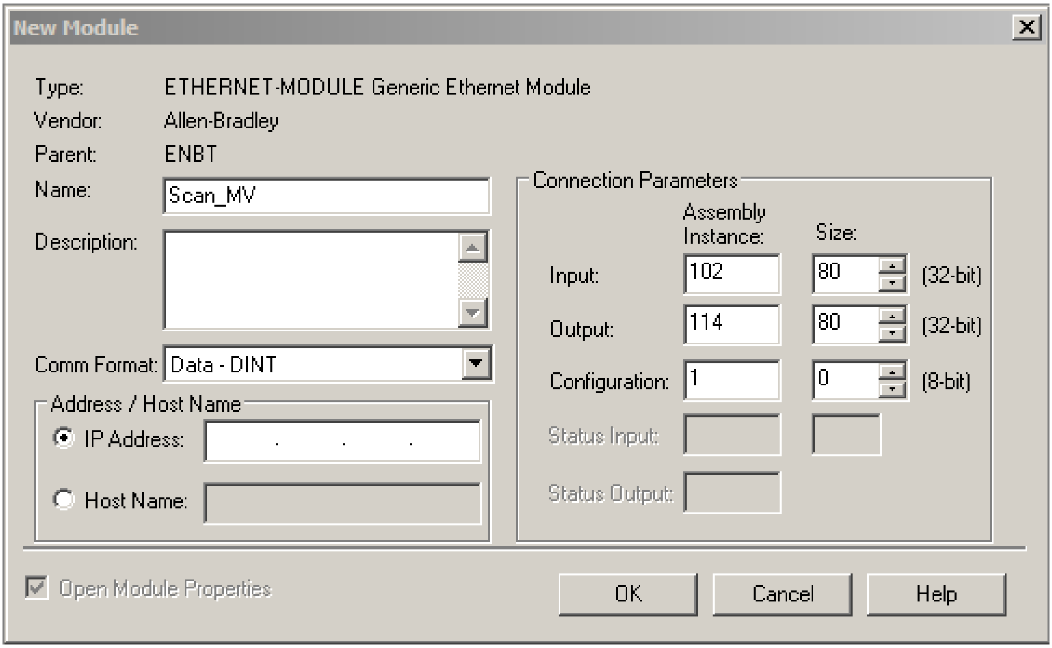
**Output, Size:** 80

**Configuration, Assembly Instance:** 1

**Configuration, Size:** 0 (none)

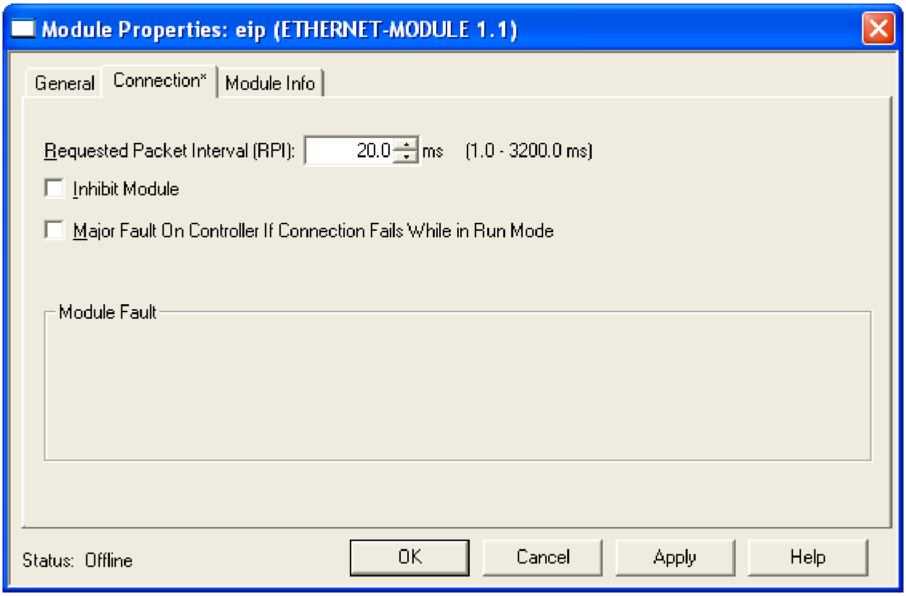
Click **OK** when done.

*Example:*

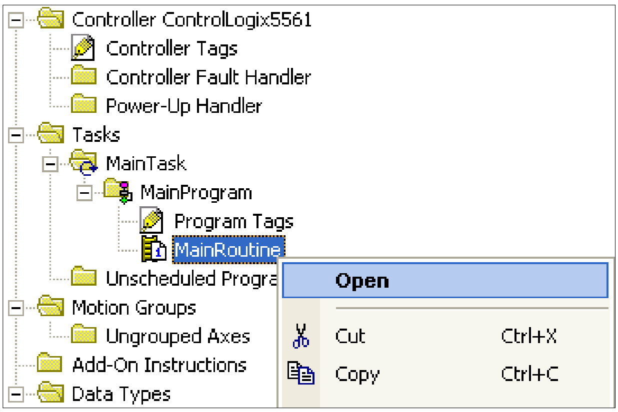


Configure the **Required Packet Interval (RPI)** and click **OK**.

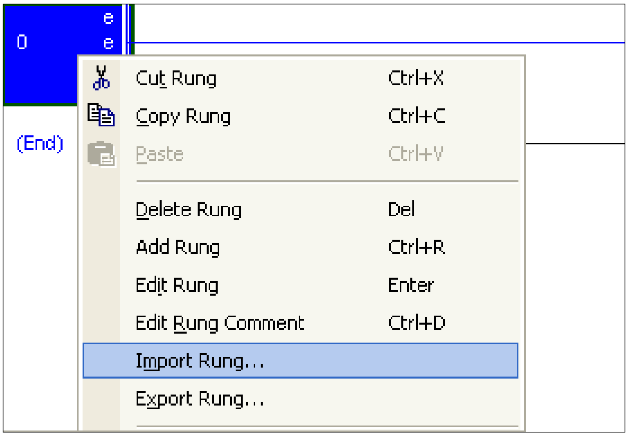
**10 ms** is the minimum allowed by the camera. **20 ms** or higher is recommended.



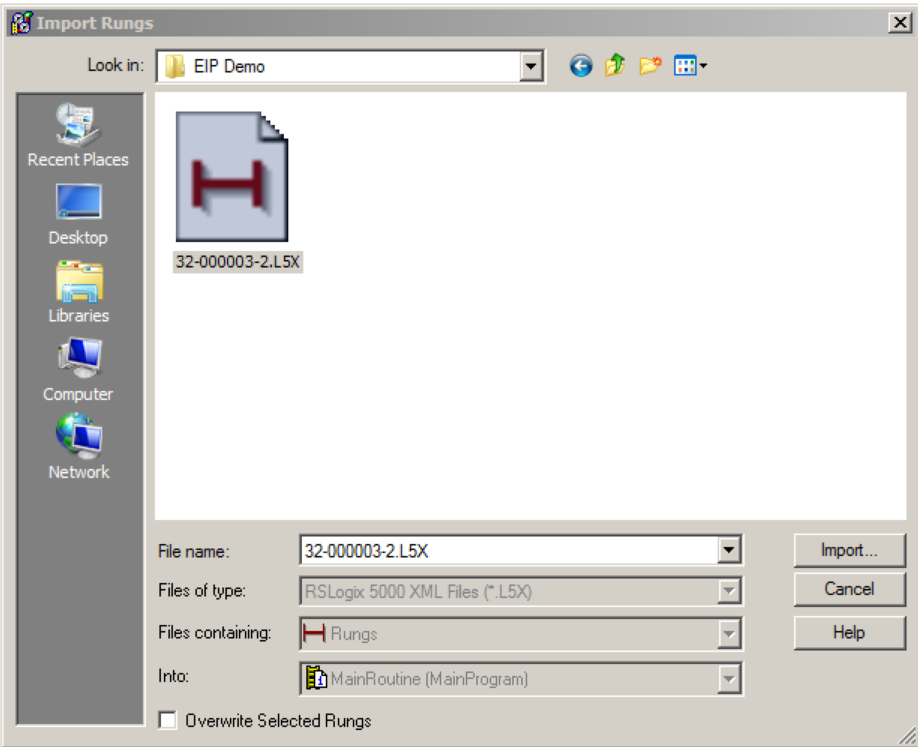

Open the **Main Routine**.



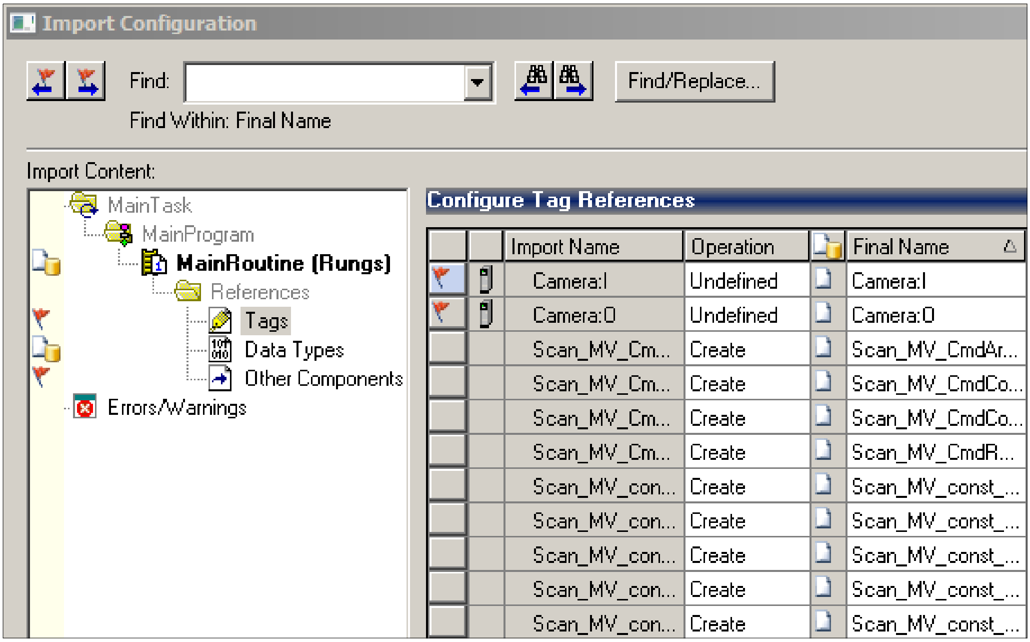
Right-click the top rung and select **Import Rung**.



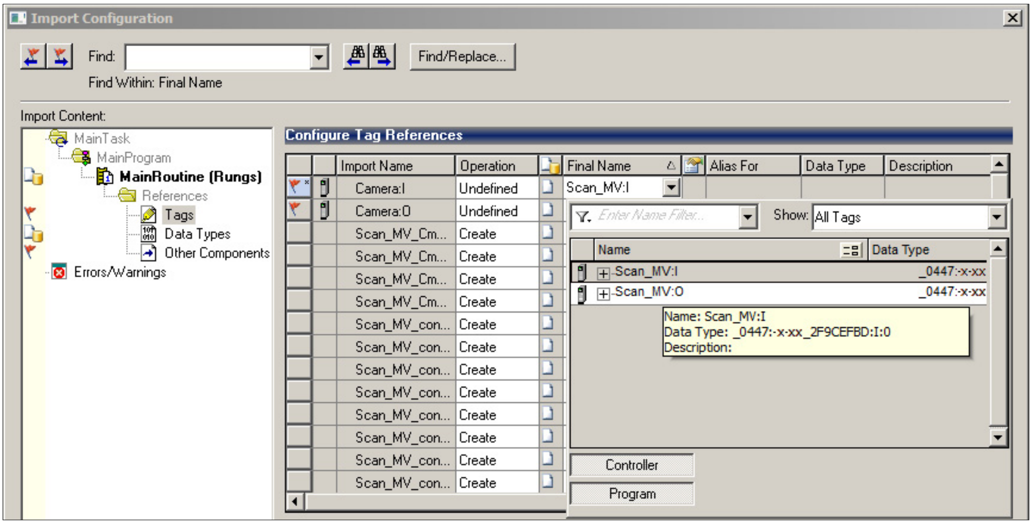

Navigate to the **32-000003-2.L5X** file and click **Import**.



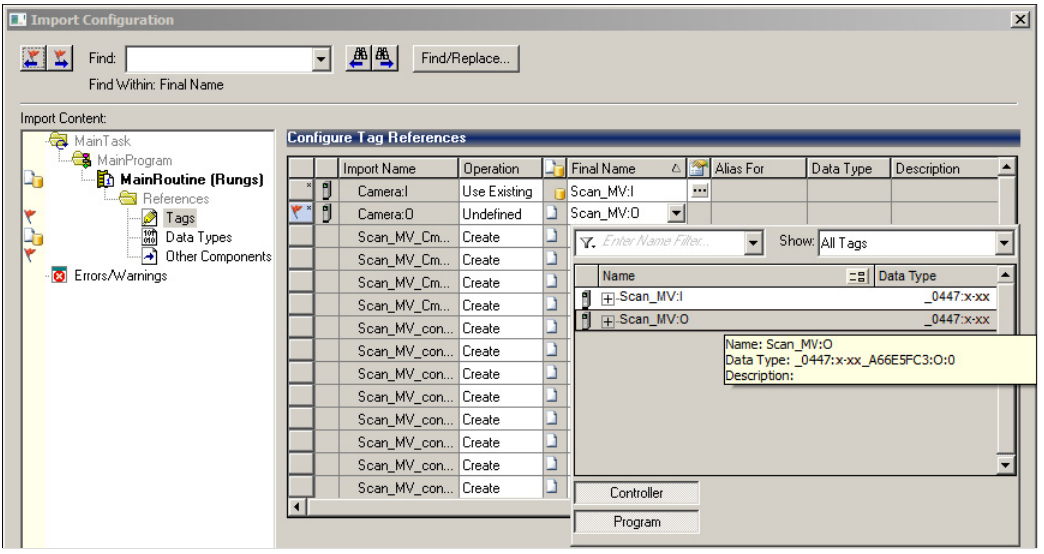
At the **Import Configuration** window, find the **Module Name** that was assigned to the **Generic Module**. Here the module name is **Camera**.




Click **Camera:I** and click the down arrow, then double-click the **Scan_MV:I** that appears below it.

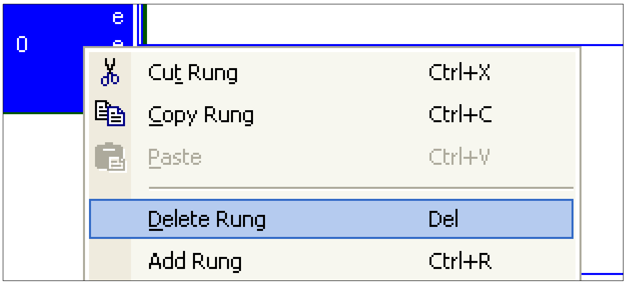


Click **Camera:O** and click the down-arrow, then double-click the **Scan_MV:O** that appears below it.



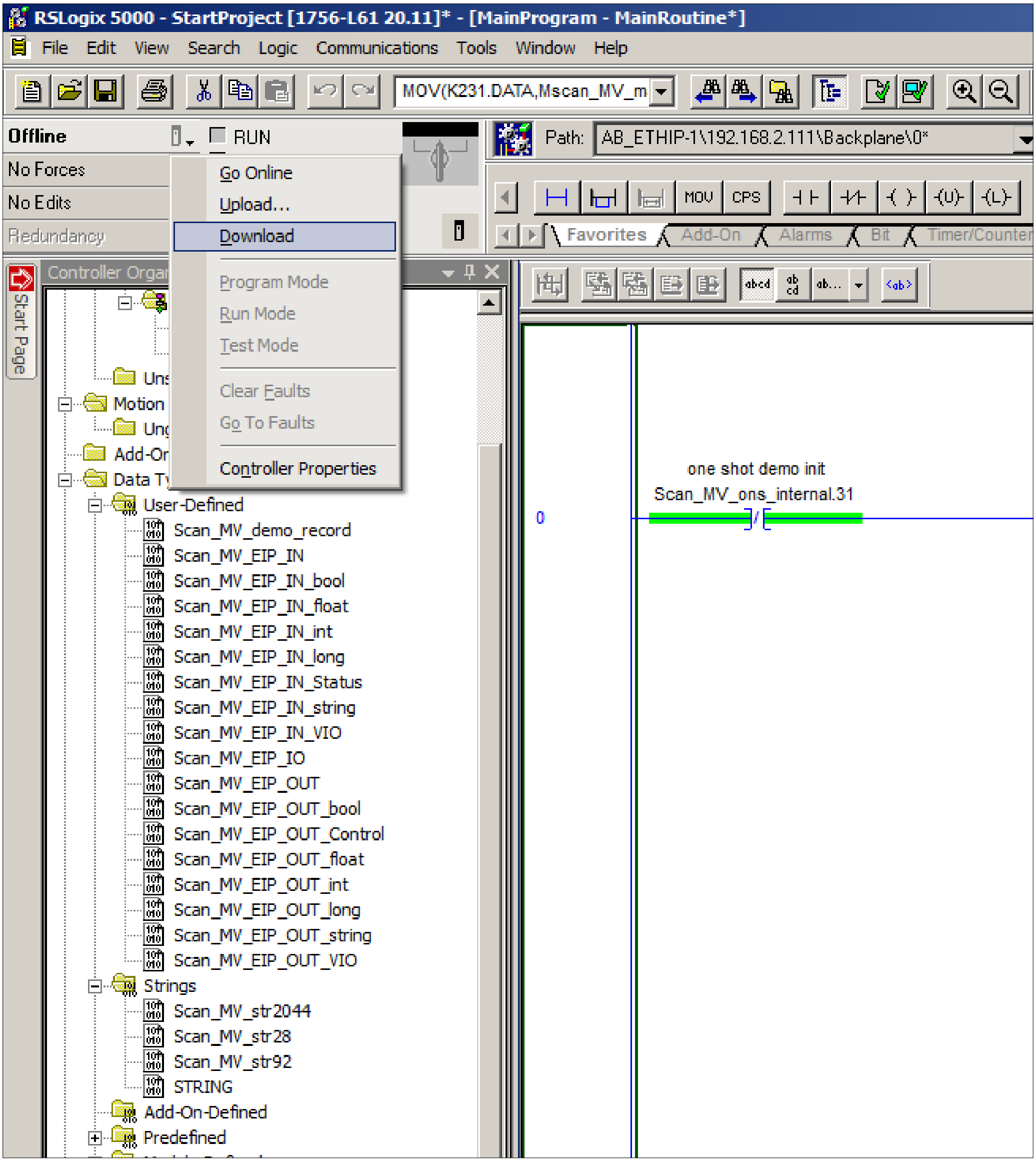

Click the **Other Components** icon in the tree view. Click **OK**.

Delete any empty rungs (rung **0** may be empty).



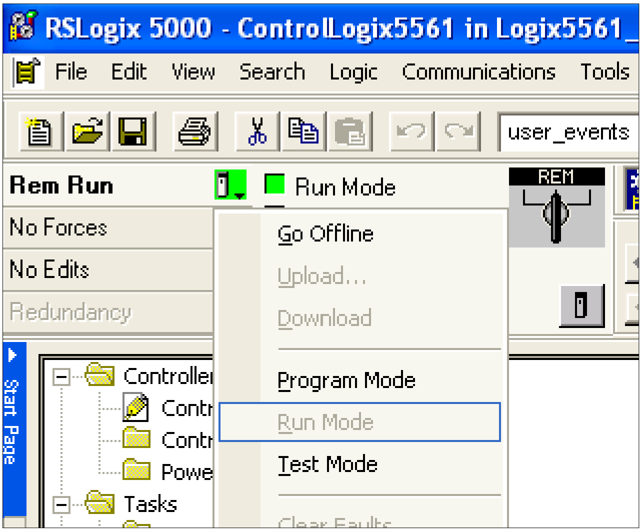

The tags and main program are now configured sufficiently to test communication with the camera.

Select the control button next to **Offline** and then select **Download**.

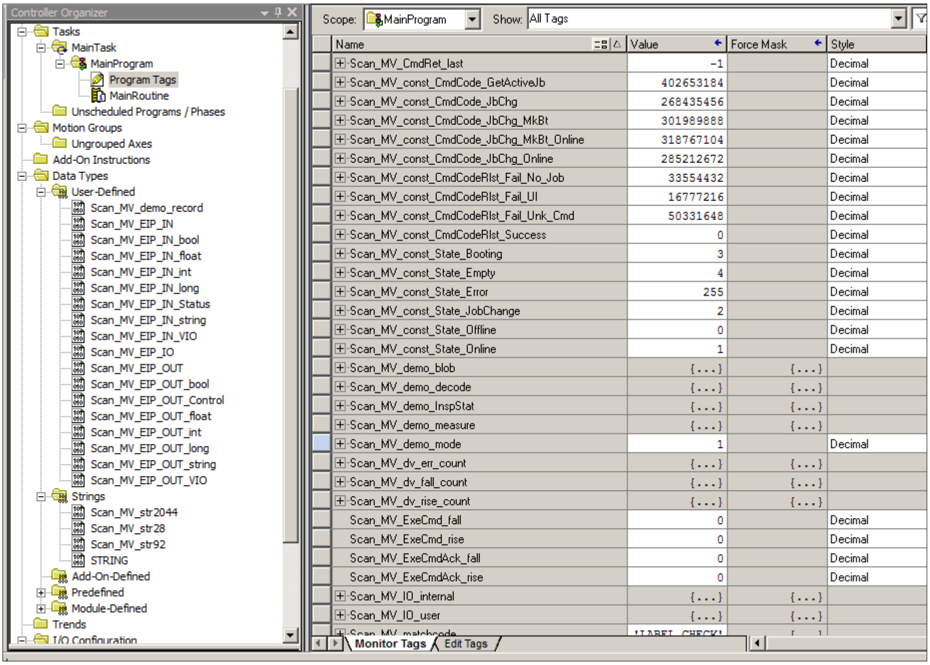



**Note:** Be sure the path to the PLC has been set in the project so that communications to the PLC can be established.

Once the program has downloaded, make sure the PLC is in **Run Mode**.



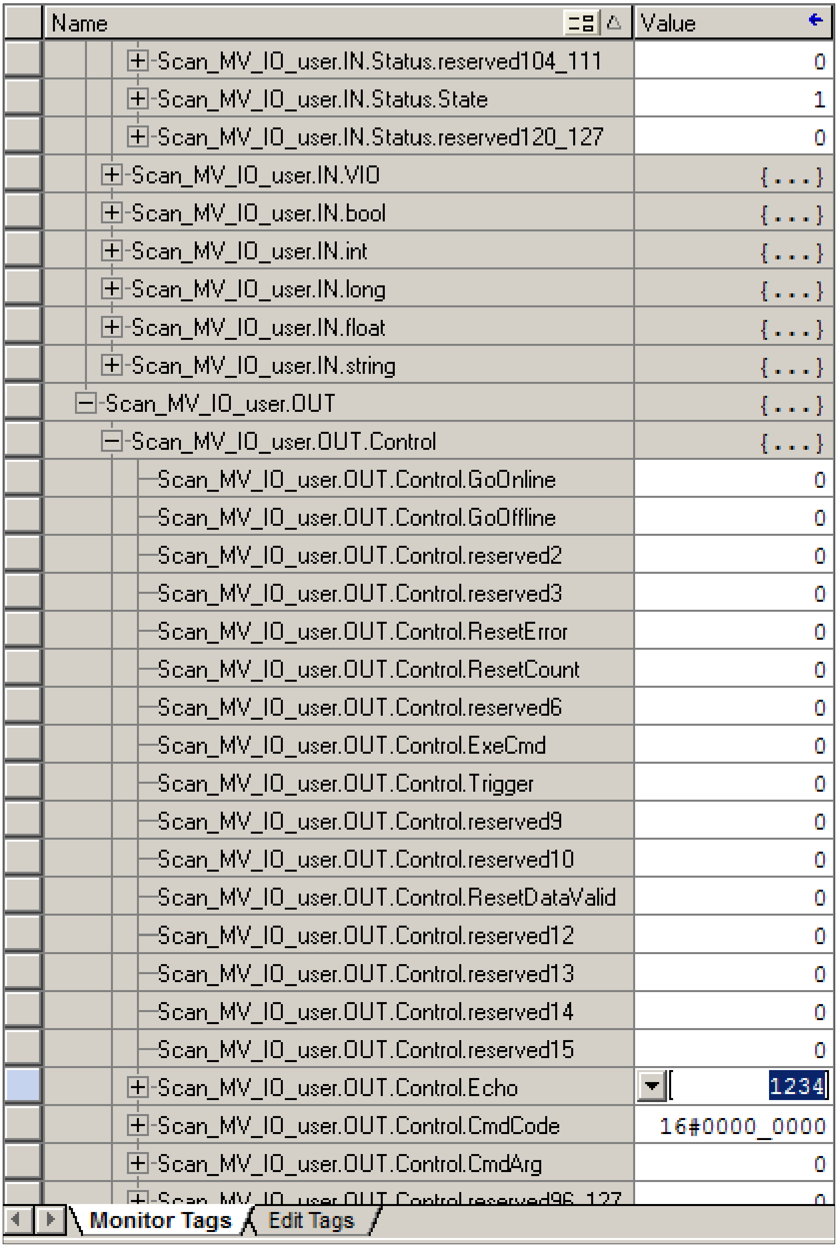

To open the **Program Tags**, double-click Program Tags, then select the **Monitor Tags** tab at the bottom of the tag window.



Expand **Scan_MV_IO_user** so that the **.IN.Status** and **.OUT.Control** structures are visible. Then scroll the window so **Scan_MV_IO_user.OUT.Control.Echo** is visible.

| Name | Value |
|---|---|
| Scan_MV_IO_user.IN.Status.reserved15 | 0 |
| + Scan_MV_IO_user.IN.Status.Echo | 0 |
| + Scan_MV_IO_user.IN.Status.CmdCodeRslt | 16#0000_0000 |
| + Scan_MV_IO_user.IN.Status.CmdRet | 0 |
| + Scan_MV_IO_user.IN.Status.reserved96_103 | 0 |
| + Scan_MV_IO_user.IN.Status.reserved104_111 | 0 |
| + Scan_MV_IO_user.IN.Status.State | 1 |
| + Scan_MV_IO_user.IN.Status.reserved120_127 | 0 |
| + Scan_MV_IO_user.IN.VIO | {...} |
| + Scan_MV_IO_user.IN.bool | {...} |
| + Scan_MV_IO_user.IN.int | {...} |
| + Scan_MV_IO_user.IN.long | {...} |
| + Scan_MV_IO_user.IN.float | {...} |
| + Scan_MV_IO_user.IN.string | {...} |
| − Scan_MV_IO_user.OUT | {...} |
| − Scan_MV_IO_user.OUT.Control | {...} |
| Scan_MV_IO_user.OUT.Control.GoOnline | 0 |
| Scan_MV_IO_user.OUT.Control.GoOffline | 0 |
| Scan_MV_IO_user.OUT.Control.reserved2 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved3 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetError | 0 |
| Scan_MV_IO_user.OUT.Control.ResetCount | 0 |
| Scan_MV_IO_user.OUT.Control.reserved6 | 0 |
| Scan_MV_IO_user.OUT.Control.ExeCmd | 0 |
| Scan_MV_IO_user.OUT.Control.Trigger | 0 |
| Scan_MV_IO_user.OUT.Control.reserved9 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved10 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetDataValid | 0 |
| Scan_MV_IO_user.OUT.Control.reserved12 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved13 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved14 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved15 | 0 |
| + Scan_MV_IO_user.OUT.Control.Echo | 0 |
| + Scan_MV_IO_user.OUT.Control.CmdCode | 16#0000_0000 |
| + Scan_MV_IO_user.OUT.Control.CmdArg | 0 |

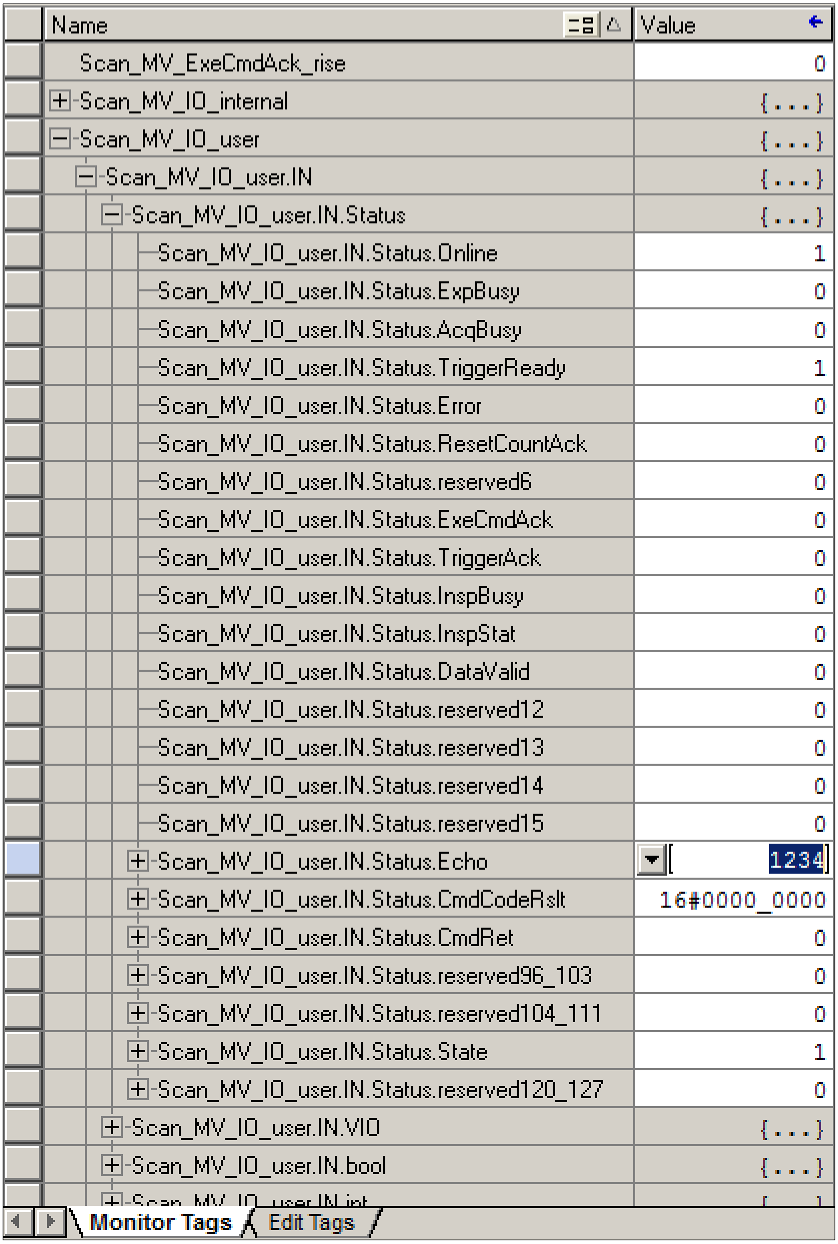
Change **.OUT.Control.Echo** to non-zero.



| Name | Value |
|---|---|
| + Scan_MV_IO_user.IN.Status.reserved104_111 | 0 |
| + Scan_MV_IO_user.IN.Status.State | 1 |
| + Scan_MV_IO_user.IN.Status.reserved120_127 | 0 |
| + Scan_MV_IO_user.IN.VIO | {...} |
| + Scan_MV_IO_user.IN.bool | {...} |
| + Scan_MV_IO_user.IN.int | {...} |
| + Scan_MV_IO_user.IN.long | {...} |
| + Scan_MV_IO_user.IN.float | {...} |
| + Scan_MV_IO_user.IN.string | {...} |
| − Scan_MV_IO_user.OUT | {...} |
| − Scan_MV_IO_user.OUT.Control | {...} |
| Scan_MV_IO_user.OUT.Control.GoOnline | 0 |
| Scan_MV_IO_user.OUT.Control.GoOffline | 0 |
| Scan_MV_IO_user.OUT.Control.reserved2 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved3 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetError | 0 |
| Scan_MV_IO_user.OUT.Control.ResetCount | 0 |
| Scan_MV_IO_user.OUT.Control.reserved6 | 0 |
| Scan_MV_IO_user.OUT.Control.ExeCmd | 0 |
| Scan_MV_IO_user.OUT.Control.Trigger | 0 |
| Scan_MV_IO_user.OUT.Control.reserved9 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved10 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetDataValid | 0 |
| Scan_MV_IO_user.OUT.Control.reserved12 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved13 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved14 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved15 | 0 |
| + Scan_MV_IO_user.OUT.Control.Echo | 1234 |
| + Scan_MV_IO_user.OUT.Control.CmdCode | 16#0000_0000 |
| + Scan_MV_IO_user.OUT.Control.CmdArg | 0 |
| + Scan_MV_IO_user.OUT.Control.reserved96_127 | 0 |

Monitor Tags / Edit Tags

Scroll the window so **Scan_MV_IO_user.IO.IN.Status.Echo** is visible, and verify that it is the same value as **.OUT.Control.Echo**.




This confirms that the PLC and camera have successful two-way communication.

To send a trigger to the camera, scroll to **Scan_MV_IO_user.Control.Trigger**.

| Name | Value |
|---|---|
| Scan_MV_IO_user.IN.int | {...} |
| Scan_MV_IO_user.IN.long | {...} |
| Scan_MV_IO_user.IN.float | {...} |
| Scan_MV_IO_user.IN.string | {...} |
| Scan_MV_IO_user.OUT | {...} |
| Scan_MV_IO_user.OUT.Control | {...} |
| Scan_MV_IO_user.OUT.Control.GoOnline | 0 |
| Scan_MV_IO_user.OUT.Control.GoOffline | 0 |
| Scan_MV_IO_user.OUT.Control.reserved2 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved3 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetError | 0 |
| Scan_MV_IO_user.OUT.Control.ResetCount | 0 |
| Scan_MV_IO_user.OUT.Control.reserved6 | 0 |
| Scan_MV_IO_user.OUT.Control.ExeCmd | 0 |
| Scan_MV_IO_user.OUT.Control.Trigger | 0 |
| Scan_MV_IO_user.OUT.Control.reserved9 | 0 |

Set the **Trigger** to **1**. This causes the demo code to trigger the camera, process the new inspection data, record the results in the **Scan_MV_demo_xxxx** tags, and clear the **DataValid** status signal. The **Trigger** control changes to **0** when the camera is triggered. The **Scan_MV_dv_fall_count** and **pass/fail** counters in the **Scan_MV_demo_xxxx** tags increment when all processing is done. For example:

| Name | Value |
|---|---|
| ⊟-Scan_MV_demo_blob | {...} |
| ⊟-Scan_MV_demo_blob.pass_count | {...} |
| ⊞-Scan_MV_demo_blob.pass_count.PRE | 0 |
| ⊞-Scan_MV_demo_blob.pass_count.ACC | 1 |
| Scan_MV_demo_blob.pass_count.CU | 0 |
| Scan_MV_demo_blob.pass_count.CD | 0 |
| Scan_MV_demo_blob.pass_count.DN | 1 |
| Scan_MV_demo_blob.pass_count.OV | 0 |
| Scan_MV_demo_blob.pass_count.UN | 0 |
| ⊞-Scan_MV_demo_blob.fail_count | {...} |
| Scan_MV_demo_blob.bool | 0 |
| ⊞-Scan_MV_demo_blob.long | 6 |
| ⊞-Scan_MV_demo_blob.long_max | 6 |
| ⊞-Scan_MV_demo_blob.long_min | 4 |
| Scan_MV_demo_blob.float | 0.0 |
| Scan_MV_demo_blob.float_min | 0.0 |
| Scan_MV_demo_blob.float_max | 0.0 |
| ⊞-Scan_MV_demo_blob.string | '' |
| ⊞-Scan_MV_demo_decode | {...} |
| ⊞-Scan_MV_demo_InspStat | {...} |
| ⊞-Scan_MV_demo_measure | {...} |
| ⊞-Scan_MV_demo_mode | 1 |
| ⊞-Scan_MV_dv_err_count | {...} |
| ⊟-Scan_MV_dv_fall_count | {...} |
| ⊞-Scan_MV_dv_fall_count.PRE | 0 |
| ⊞-Scan_MV_dv_fall_count.ACC | 1 |
| Scan_MV_dv_fall_count.CU | 0 |
| Scan_MV_dv_fall_count.CD | 0 |
| Scan_MV_dv_fall_count.DN | 1 |
| Scan_MV_dv_fall_count.OV | 0 |
| Scan_MV_dv_fall_count.UN | 0 |
| ⊞-Scan_MV_dv_rise_count | {...} |

Monitor Tags / Edit Tags

## Parameterize the Camera

Open the **Scan_MV_IO_user.OUT.long**, **float**, and **string** tags and verify that they are configured as shown below.

| Name | Value | | Style | Data Typ |
|---|---|---|---|---|
| ⊟-Scan_MV_IO_user | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.IN | {...} | [ | | Scan_... |
| ⊟-Scan_MV_IO_user.OUT | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.Control | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.VIO | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.bool | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.int | {...} | [ | | Scan_... |
| ⊟-Scan_MV_IO_user.OUT.long | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.long.long101 | 4 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long102 | 4 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long103 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long104 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long105 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long106 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long107 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long108 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long109 | 0 | | Decimal | DINT |
| ⊞-Scan_MV_IO_user.OUT.long.long110 | 0 | | Decimal | DINT |
| ⊟-Scan_MV_IO_user.OUT.float | {...} | [ | | Scan_... |
| Scan_MV_IO_user.OUT.float.float101 | 100.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float102 | 200.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float103 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float104 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float105 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float106 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float107 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float108 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float109 | 0.0 | | Float | REAL |
| Scan_MV_IO_user.OUT.float.float110 | 0.0 | | Float | REAL |
| ⊟-Scan_MV_IO_user.OUT.string | {...} | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.string.string101 | 'LABEL CHECK' | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.string.string102 | '' | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.string.string103 | '' | [ | | Scan_... |
| ⊞-Scan_MV_IO_user.OUT.string.string104 | '' | [ | | Scan_... |
| ⊞-Scan_MV_matchcode | 'LABEL CHECK' | [ | | Scan_... |
| ⊞-Scan_MV_ons_internal | -2080374528 | | Decimal | DINT |
| ⊞-Scan_MV_status_err_count | {...} | [ | | COUN... |

This configures the **Measure (float101 and float102)**, **Decode (string101)** and **Count Blob (long101 and long102)** tools in the same way they were configured in AutoVISION during Try Out.

Note the **Description** column. It offers a hint for what each linked tag does in the vision job.

## Trigger the Camera

To send a trigger to the camera, scroll to **Scan_MV_IO_user.Control.Trigger**.

| Name | Value |
|---|---|
| Scan_MV_IO_user | {...} |
| Scan_MV_IO_user.IN | {...} |
| Scan_MV_IO_user.OUT | {...} |
| Scan_MV_IO_user.OUT.Control | {...} |
| Scan_MV_IO_user.OUT.Control.GoOnline | 0 |
| Scan_MV_IO_user.OUT.Control.GoOffline | 0 |
| Scan_MV_IO_user.OUT.Control.reserved2 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved3 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetError | 0 |
| Scan_MV_IO_user.OUT.Control.ResetCount | 0 |
| Scan_MV_IO_user.OUT.Control.reserved6 | 0 |
| Scan_MV_IO_user.OUT.Control.ExeCmd | 0 |
| Scan_MV_IO_user.OUT.Control.Trigger | 0 |
| Scan_MV_IO_user.OUT.Control.reserved9 | 0 |
| Scan_MV_IO_user.OUT.Control.reserved10 | 0 |
| Scan_MV_IO_user.OUT.Control.ResetDataValid | 0 |
| Scan_MV_IO_user.OUT.Control.reserved12 | 0 |

Set the **Trigger** to **1**. When the Trigger returns to a value of **0**, the camera may be re-triggered.

If you connect to the camera with AutoVISION, it will display a new inspection result each time the camera is triggered. Recall that the vision job was created with predefined images to produce predictable **Passed** and **Failed** results. The camera's illumination lights will not flash when triggered.

The inspection results can be seen in the PLCs's **IN** tags, as well as in AutoVISION. Open the RSLogix tag window so **Scan_MV_IO_user.IN.Status** and **bool** are visible.

This example shows a **Passed** inspection, where the following tags are all **1**:

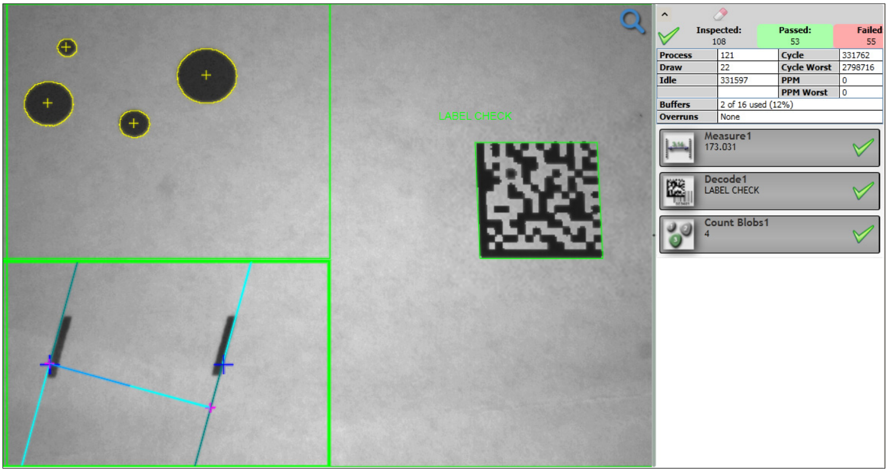
- IN.Status.InspStat
- IN.bool.bool1 (Measure status)
- IN.bool.bool2 (decode+matchcode status)
- IN.bool.bool3 (count blob status)

| Name | Value | | Style | Data Typ | Description |
|---|---|---|---|---|---|
| Scan_MV_IO_user | {...} | [ | | Scan_... | user's device tags when M |
| Scan_MV_IO_user.IN | {...} | [ | | Scan_... | user's device tags when M |
| Scan_MV_IO_user.IN.Status | {...} | [ | | Scan_... | user's device tags when M |
| Scan_MV_IO_user.IN.Status.Online | 1 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.ExpBusy | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.AcqBusy | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.TriggerReady | 1 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.Error | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.ResetCountAck | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved6 | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.ExeCmdAck | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.TriggerAck | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.InspBusy | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.InspStat | 1 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.DataValid | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved12 | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved13 | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved14 | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved15 | 0 | | Decimal | BOOL | user's device tags when M |
| Scan_MV_IO_user.IN.Status.Echo | 0 | | Decimal | INT | user's device tags when M |
| Scan_MV_IO_user.IN.Status.CmdCodeRslt | 16#0000_0000 | | Hex | DINT | user's device tags when M |
| Scan_MV_IO_user.IN.Status.CmdRet | 0 | | Decimal | DINT | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved96_103 | 0 | | Decimal | SINT | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved104_111 | 0 | | Decimal | SINT | user's device tags when M |
| Scan_MV_IO_user.IN.Status.State | 1 | | Decimal | SINT | user's device tags when M |
| Scan_MV_IO_user.IN.Status.reserved120_127 | 0 | | Decimal | SINT | user's device tags when M |
| Scan_MV_IO_user.IN.VIO | {...} | [ | | Scan_... | user's device tags when M |
| Scan_MV_IO_user.IN.bool | {...} | [ | | Scan_... | user's device tags when M |
| Scan_MV_IO_user.IN.bool.bool1 | 1 | | Decimal | BOOL | measure status |
| Scan_MV_IO_user.IN.bool.bool2 | 1 | | Decimal | BOOL | decode+matchcode status |
| Scan_MV_IO_user.IN.bool.bool3 | 1 | | Decimal | BOOL | blob count status |

If you scroll down to the **IN.long**, **float**, and **string** values, you will see the literal results of the vision tools.

| Name | Value | | Style | Data Typ | Description |
|---|---|---|---|---|---|
| +-Scan_MV_IO_user.IN.VIO | {...} | { | | Scan_... | user's device tag |
| +-Scan_MV_IO_user.IN.bool | {...} | { | | Scan_... | user's device tag |
| +-Scan_MV_IO_user.IN.int | {...} | { | | Scan_... | user's device tag |
| −-Scan_MV_IO_user.IN.long | {...} | { | | Scan_... | user's device tag |
| +-Scan_MV_IO_user.IN.long.long1 | 4 | | Decimal | DINT | Blob count |
| +-Scan_MV_IO_user.IN.long.long2 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long3 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long4 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long5 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long6 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long7 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long8 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long9 | 0 | | Decimal | DINT | user's device tag |
| +-Scan_MV_IO_user.IN.long.long10 | 0 | | Decimal | DINT | user's device tag |
| −-Scan_MV_IO_user.IN.float | {...} | { | | Scan_... | user's device tag |
| Scan_MV_IO_user.IN.float.float1 | 173.0306 | | Float | REAL | Measure value |
| Scan_MV_IO_user.IN.float.float2 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float3 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float4 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float5 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float6 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float7 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float8 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float9 | 0.0 | | Float | REAL | user's device tag |
| Scan_MV_IO_user.IN.float.float10 | 0.0 | | Float | REAL | user's device tag |
| −-Scan_MV_IO_user.IN.string | {...} | { | | Scan_... | user's device tag |
| +-Scan_MV_IO_user.IN.string.string1 | 'LABEL CHECK' | { | | Scan_... | Decode text |

This is equivalent to the AutoVISION inspection result.



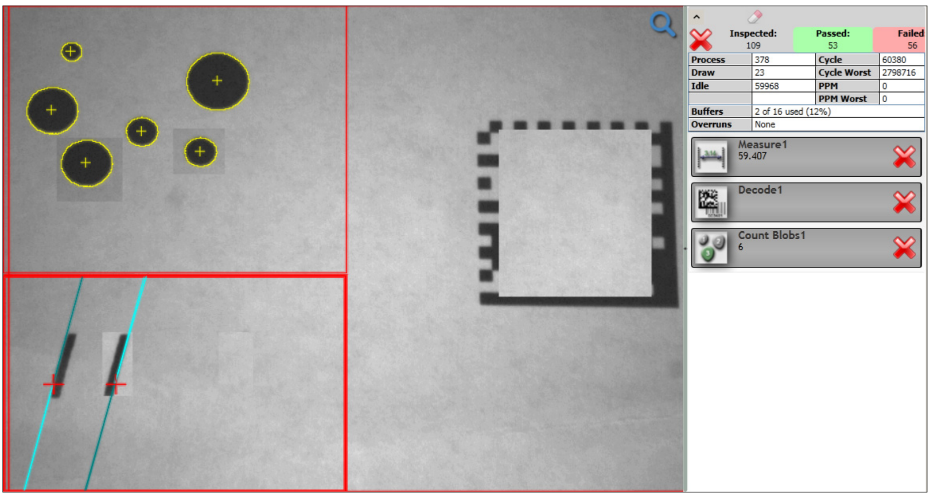

This example shows a **Failed** inspection, where every tool reports a fail.

| Name | Value | Style | Data Typ | Description |
|---|---|---|---|---|
| Scan_MV_IO_user.IN | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.Status | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.Status.Online | 1 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.ExpBusy | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.AcqBusy | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.TriggerReady | 1 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.Error | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.ResetCountAck | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved6 | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.ExeCmdAck | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.TriggerAck | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.InspBusy | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.InspStat | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.DataValid | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved12 | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved13 | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved14 | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved15 | 0 | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.Echo | 0 | Decimal | INT | user's device tags |
| Scan_MV_IO_user.IN.Status.CmdCodeRslt | 16#0000_0000 | Hex | DINT | user's device tags |
| Scan_MV_IO_user.IN.Status.CmdRet | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved96_103 | 0 | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved104_111 | 0 | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.Status.State | 1 | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved120_127 | 0 | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.VIO | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.bool | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.bool.bool1 | 0 | Decimal | BOOL | measure status |
| Scan_MV_IO_user.IN.bool.bool2 | 0 | Decimal | BOOL | decode+matchcoc |
| Scan_MV_IO_user.IN.bool.bool3 | 0 | Decimal | BOOL | blob count status |

This is the Failed inspection's literal data.

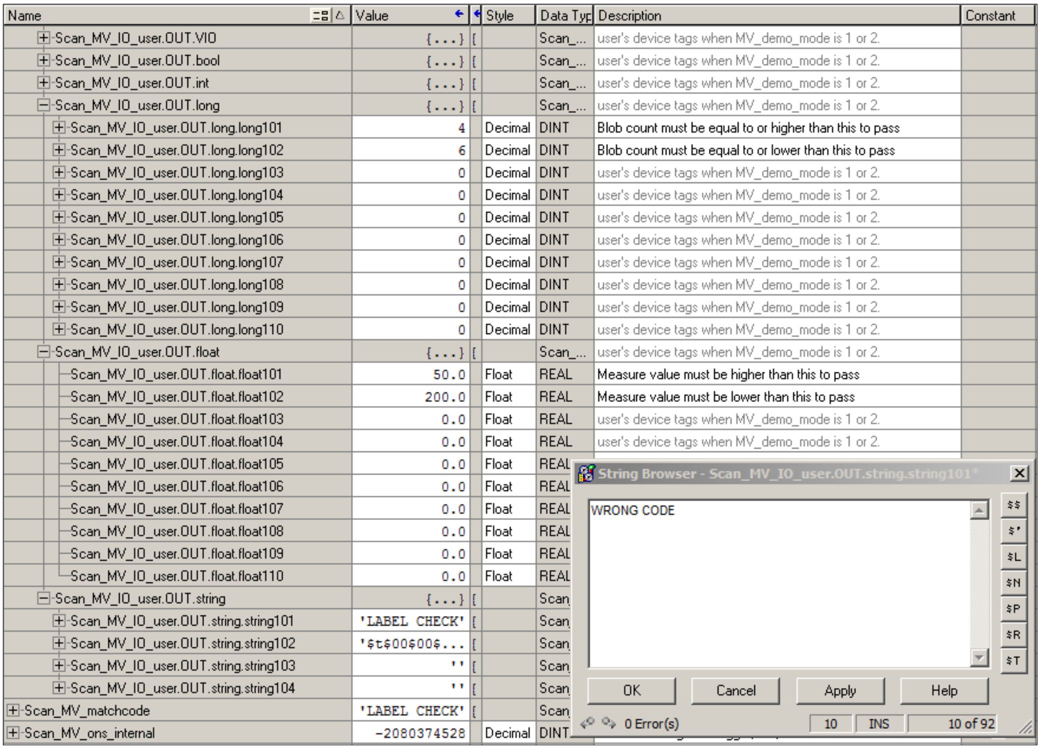
| Name | Value | Style | Data Typ | Description |
|---|---|---|---|---|
| Scan_MV_IO_user.IN.long | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.long.long1 | 6 | Decimal | DINT | Blob count |
| Scan_MV_IO_user.IN.long.long2 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long3 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long4 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long5 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long6 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long7 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long8 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long9 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.long.long10 | 0 | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.float | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.float.float1 | 59.406998 | Float | REAL | Measure value |
| Scan_MV_IO_user.IN.float.float2 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float3 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float4 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float5 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float6 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float7 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float8 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float9 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.float.float10 | 0.0 | Float | REAL | user's device tags |
| Scan_MV_IO_user.IN.string | {...} | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.string.string1 | '' | | Scan_... | Decode text |

This is equivalent to the AutoVISION inspection report.



## Parameterize the Camera Again

The **Measure** and **Count Blob** tools can be parameterized by the PLC so they always pass. The **Decode Tool** can be parameterized so it always fails, either due to no decode, or a **Match Strings** mismatch. Scroll the tag window so **OUT.long**, **float**, and **string** are visible, then change them as shown below.



## Trigger the Camera Again

Trigger the camera twice and you will see the Status results stay the same for all triggers:

**bool2 (decode+matchcode status) = 0**

Why: Decode + Matchcode status always fails because the matchcode has been changed to wrong code, or there is no decode.

**bool1 (Measure status) and bool3 (count blob status) = 1**

Why: The inspected values are now within tolerance.

**InspStat = 0**

Why: The Decode tool fails, so the overall Inspection result is a Fail.

**PLC tags:**

| Name | Value | | Style | Data Typ | Description |
|---|---|---|---|---|---|
| Scan_MV_IO_user.IN | {...} | [ | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.Status | {...} | [ | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.Status.Online | 1 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.ExpBusy | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.AcqBusy | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.TriggerReady | 1 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.Error | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.ResetCountAck | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved6 | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.ExeCmdAck | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.TriggerAck | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.InspBusy | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.InspStat | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.DataValid | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved12 | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved13 | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved14 | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved15 | 0 | | Decimal | BOOL | user's device tags |
| Scan_MV_IO_user.IN.Status.Echo | 0 | | Decimal | INT | user's device tags |
| Scan_MV_IO_user.IN.Status.CmdCodeRslt | 16#0000_0000 | | Hex | DINT | user's device tags |
| Scan_MV_IO_user.IN.Status.CmdRet | 0 | | Decimal | DINT | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved96_103 | 0 | | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved104_111 | 0 | | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.Status.State | 1 | | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.Status.reserved120_127 | 0 | | Decimal | SINT | user's device tags |
| Scan_MV_IO_user.IN.VIO | {...} | [ | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.bool | {...} | [ | | Scan_... | user's device tags |
| Scan_MV_IO_user.IN.bool.bool1 | 1 | | Decimal | BOOL | measure status |
| Scan_MV_IO_user.IN.bool.bool2 | 0 | | Decimal | BOOL | decode+matchcode |
| Scan_MV_IO_user.IN.bool.bool3 | 1 | | Decimal | BOOL | blob count status |

This concludes the EtherNet/IP demo.

CHAPTER 6

# Demo EtherNet/IP PLC Code

This section describes how to use Omron Microscan demo PLC code with a vision job and camera target.

The EtherNet/IP demo files can be found where AutoVISION is installed, in the folder **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\EIP demo**. Open the **EIP_demo.avp** with AutoVISION and download it to the camera.

During PLC integration, import the 32-000003-2.L5X file to create the camera's demo tags and ladder logic.

**Notes:**

- The camera communications protocol must be enabled for EtherNet/IP before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the camera, and information about switching camera communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate EtherNet/IP communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.

# Glossary of Terms

The following terms are used in the description of Microscan's demo PLC program.

## Camera

The Omron Microscan Smart Camera used in this application, which has an EtherNet/IP communication interface.

## User App

The PLC logic code written by the end user or system integrator.

## Demo Code

The PLC logic code distributed by Omron Microscan that can be imported into the PLC's ladder logic area. It encapsulates most of the device Control and Status management.

The demo code expects a demo vision job loaded on the camera. However, the demo code will operate whether or not the demo vision job is loaded on the camera.

## Activate / Set High

Writing a 1 value to a single Control bit, or any other bool bit.

## Active

A Control, Status, bool, or PLC logic "contact" in a 1 state.

## Clear

A Control, Status, bool, or PLC logic "contact" in a 0 state.

## One Shot

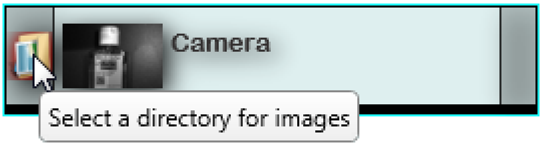
A PLC tag write operation that is performed once, typically in reaction to an event. After a one shot operation, the PLC logic does not write to the same tag again unless another event occurs.

# Demo Setup

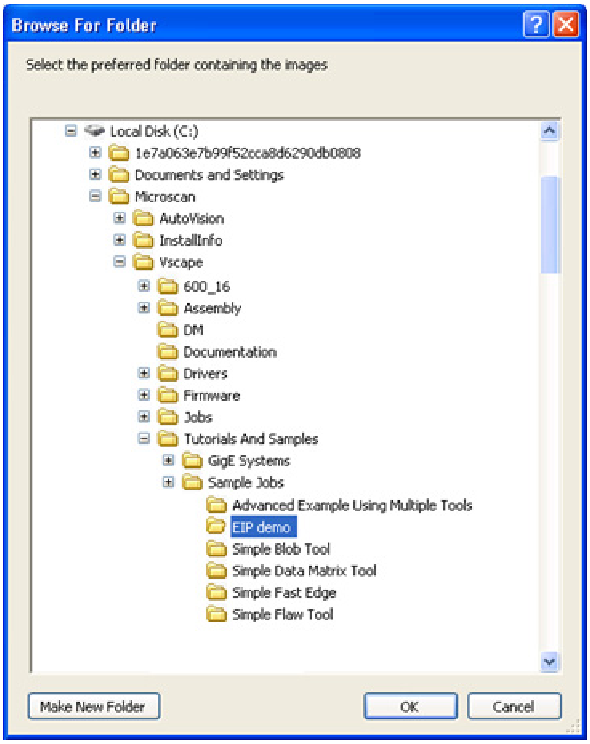
## Vision Job Setup

The EtherNet/IP demo files can be found where AutoVISION is installed, where the default folder is **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\EIP demo**.

1. Open **EIP_demo.avp** with AutoVISION.
2. To use pre-defined images, select the icon shown here.

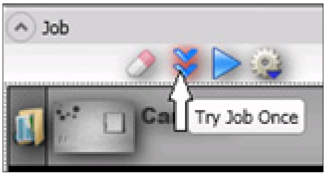



3. Browse to the **EIP demo** folder, select it, and click **OK**.



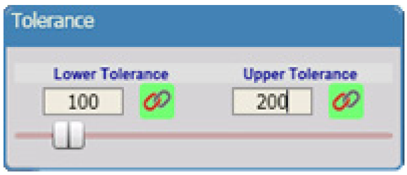
After the **EIP demo** folder has been enabled for image load, the camera icon will change to a folder.


4. While in **Edit** mode, **Try Out** can be used to get an understanding of what to expect after the job is sent to the camera.

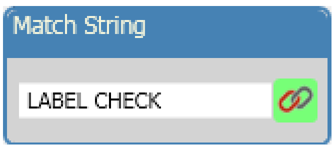



Before Try Out can be effective, the **Measure**, **Decode**, and **Count Tool** parameters must be specified. After job download, the tool parameters will be supplied by the PLC.

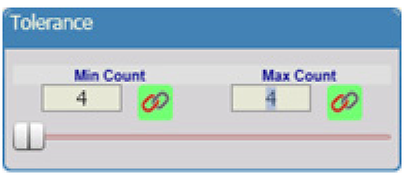
**Measure Tolerance:**




**Decode Matchstring:**

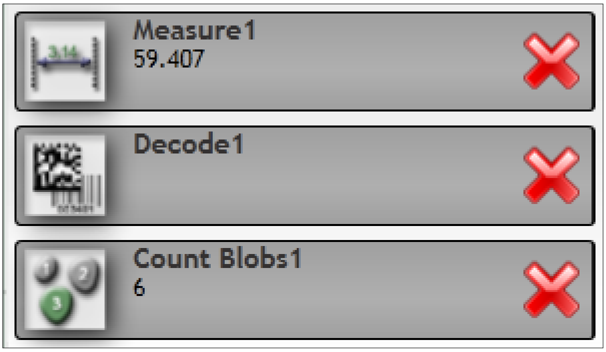



**Count Tolerance:**

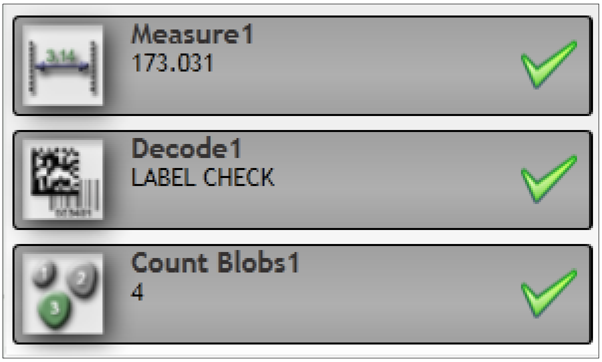


With these tool parameter configured as shown, Try Out will show the following Pass/Fail results:

**Fail:**




**Pass:**




5. Download the job to the camera.
6. Add the camera and demo code to the PLC environment (see the next section).

## PLC Demo Code Setup

During PLC integration, import the **32-000003-2.L5X** file found in the **EIP demo** folder to create the camera's demo tags and ladder logic (refer to the **Allen-Bradley PLC Setup** chapters).

# Description of PLC Tags

## Scan_MV_demo_mode

**Purpose**

This tag is intended for demonstration purposes only. It modifies the operation of the demo code and allows the first-time user to control the device directly with no assistance from the demo code. It also allows the demo code to manage the Control and Status signals fully.

The demo mode tag takes three different values, putting the demo code into one of three modes of operation:

- Exchange I/O data only
- Actively operate device controls, status, and demo data
- Automatically trigger the device after one second of idle time

**User App Method**

User app can set the demo mode with one of three values to define the demo code's mode of operation.

**0 = Exchange I/O data only**

In this mode, the user directly accesses the **Scan_MV_IO_internal** tag set. The demo code only exchanges data with the camera, doing nothing to control the device or respond to events from it.

**1 = Operate device controls and respond to device events**

This is the default mode of the demo code. In this mode, the user app accesses the **Scan_MV_IO_user** tag to control and monitor the camera. The user app must not access the Scan_MV_IO_internal tag set.

In this mode, the user activates the controls in **Scan_MV_IO_user.OUT.Control** (Trigger, ResetCount, GoOnline, GoOffline, ResetError, ExeCmd), and the demo code handles the rest.

**2 = Auto-trigger**

In this mode, the demo code fully manages the **Control** and **Status** signals, the same as when the mode is set to **1**. It also activates the camera's trigger after one second of idle time. The timer used to drive the trigger is **Scan_MV_trigger_delay_timer**.

### Demo Code Usage

Depending on the mode, the demo code will run the appropriate level of code.

- In **mode 0**, only the I/O exchange rungs are executed. All others are bypassed.
- In **mode 1**, the auto-trigger rungs are bypassed. This is the default mode of the demo code.
- In **mode 2**, all rungs are executed.

## Scan_MV_IO_user

### Purpose

User-accessible I/O data for the camera. The user app reads and writes these I/O tags, and the demo code handles the actual on-the-wire control of the camera.

### User App Method

Activate a Control by setting its value to **1**.

The user app can determine that the Control is done when the Control is clear (demo code changes the Control to a bit/bool value of **0**). Do not attempt to activate a Control unless it is clear.

The user app should activate the Controls using one shot writes. The use app should not continuously hold a Control in an active state. Holding a Control in an active state will prevent the demo code from notifying the user app that the Control operation is complete by clearing the Control.

Usually, when a Control is clear (**0**), the camera is ready for the Control to be activated again. Please see the **Specific Control Guidelines** and **Specific Status Guidelines** below for qualifications.

### Demo Code Usage

The demo code waits for the user app to activate a Control. When the user app activates a Control, the demo code handles all handshaking and confirmation that the Control operation is performed by the camera. When the operation is complete, the demo code clears the Control back to **0**.

## Specific Control Guidelines

### GoOnline and GoOffline

In order to take the camera **Online** and **Offline**, only one of these Controls can go active (change from **0** to **1**), and be active, at any given time.

### ResetCount

After the user app activates **ResetCount**, the demo code will clear ResetCount when the operation is complete. The next Inspection output will be **1** (as can be seen if AutoVISION is connected to the camera in run mode).

### Trigger

Do not trigger the camera unless the **TriggerReady Status** is active. If the Trigger goes active when TriggerReady is not active, the demo code increments the counter **Scan_MV_trigger_err_count**, and immediately clears the Trigger Control without attempting to trigger the camera.

After the user app activates the Trigger, the demo code will clear the Trigger when the camera indicates it has accepted the Trigger.

Do not re-trigger the camera until **DataValid** in the **Status** register goes active, all **Inspection** data has been processed, and DataValid is cleared using the **ResetDataValid Control**.

### ResetDataValid

When the user app sees DataValid go active, it should process the Inspection data, then clear DataValid by activating **ResetDataValid**.

See **Data Valid** for more details.

### ResetError

To clear the Error Status, activate **ResetError**.

### ExeCmd, CmdCode, CmdArg

These Controls can be used to perform a job change, and query the active job slot. The demo code includes tags with pre-defined **CmdCode** and **CmdCodeRslt** definitions:

| Tag | Meaning |
|---|---|
| Scan_MV_const_CmdCode_GetActiveJb | Query the active job slot number (returned in CmdRet) |
| Scan_MV_const_CmdCode_JbChg | Go Offline, Load job specified by LSB |
| Scan_MV_const_CmdCode_JbChg_MkBt | Go Offline, Load Job specified by LSB, Make it the boot job |
| Scan_MV_const_CmdCode_JbChg_MkBt_Online | Go Offline, Load Job specified by LSB, Make it the boot job, Go Online |
| Scan_MV_const_CmdCode_JbChg_Online | Go Offline, Load Job specified by LSB, Go Online |
| Scan_MV_const_CmdCodeRlst_Fail_No_Job | Job Change failed: No job in slot |
| Scan_MV_const_CmdCodeRlst_Fail_UI | PC UI is controlling the camera, |
| Scan_MV_const_CmdCodeRlst_Fail_Unk_Cmd | Job Change failed: Unknown CmdCode |
| Scan_MV_const_CmdCodeRlst_Success | Completed operation OK. |

The **ExeCmd**, **CmdCode**, and **CmdArg** controls are used in combination with these Status signals:

| Control signal | Status signal |
|---|---|
| ExeCmd | ExeCmdAck |
| CmdCode | CmdCodeRslt |
| CmdArg | CmdRet |

The demo code records the final result of the command operation by copying **CmdCode**, **CmdArg**, **CmdCodeRslt** and **CmdRet** to the following tags:

| Source Control/Status tag | Final result tag |
|---|---|
| Scan_MV_IO_user.OUT.Control.ExeCmd | Scan_MV_CmdCode_last |
| Scan_MV_IO_user.OUT.Control.CmdArg | Scan_MV_CmdArg_last |
| Scan_MV_IO_internal.IN.Status.CmdCodeRslt | Scan_MV_CmdCodeRslt_last |
| Scan_MV_IO_internal.IN.Status.CmdRet | Scan_MV_CmdRet_last |

The demo code will automate the command process when **Scan_MV_demo_mode** is **1**, which is the default value at program startup, similar to how it assists the Triggering and DataValid Controls. The PLC integrator can initiate command operation by accessing the demo code's **Scan_MV_IO_user** tag set for Control and Status signals. While a command operation is active, the demo code forces all Control signals to an inactive state, except for the Echo. No Controls can be activated until the command operation is completed. To verify the camera is still "alive" during command execution, **Control.Echo** can be incremented, and the **Status.Echo** will update accordingly.

When the demo code automates the command process, the PLC integrator is responsible for the following steps:

1. Deactivate all Controls and clear DataValid and Error status signals. This is a best-practice measure to ensure that the PLC has transitioned from a state of triggering and processing inspections to issuing a command.
2. If a job change command is to be issued, populate the output tags required to configure the new job (**bool**, **int**, **long**, **float**, **string**).
3. Write the required **CmdCode** (see **Scan_MV_const_CmdCode_xxxx** tags) and **CmdArg**, then activate **ExeCmd**.
4. Wait for ExeCmd to go inactive (per typical demo mode **1** operation). Note that job changes can take up to a minute. While a job change command is being executed, the **Status**.State tag will be **2**.
5. When ExeCmd goes inactive, verify the following:

   **Scan_MV_CmdCodeRslt_last** is **0** (Success)

   **Scan_MV_CmdRest_last** contains the returned data from the command (if any)

   **Status.State** has changed to **0** (Offline) or **1** (Online)

   **ExeCmdAck** is inactive (**0**)

   **Status.Error** is inactive (**0**)

6. Put the camera online (if necessary), and continue with normal runtime operation.

## Specific Status Guidelines

### Online

The camera cannot be Triggered or generate Inspection data unless **Online** is active. See the **GoOnline Control**.

### TriggerReady

Do not attempt to trigger the camera unless **TriggerReady** is active.

See the description of the **Trigger** for more details.

### TriggerAck and ResetCountAck

Used by the demo code to complete the respective operations.

### DataValid

When **DataValid** goes active, the user app should process the Inspection data, then clear DataValid using the **ResetDataValid** control. This is handled, by the demo code in mode **1** and **2**, as a demonstration for the user app.

If the camera's DataValid goes active, but the user app has not cleared a previous DataValid event, the demo code does not overwrite **Scan_MV_IO_user** with new Inspection data. Instead, the demo code increments the counter **Scan_MV_dv_err_count**. The new Inspection data remains stranded in the **Scan_MV_internal** tag set, and is effectively lost.

## Scan_MV_trigger_count

Incremented by the demo code when a new trigger is issued to the camera over the EtherNet/IP interface (Trigger Control activated).

## Scan_MV_trigger_err_count

Incremented by the demo code if the user app attempts to trigger the camera when TriggerReady is not active.

## Scan_MV_dv_err_count

Incremented by the demo code when new Inspection data is received from the camera, but the user app has not cleared the previous DataValid.

## Scan_MV_status_err_count

Incremented by the demo code whenever the Error Status goes active.

## Scan_MV_demo_blob, Scan_MV_demo_decode, Scan_MV_demo_InspStat, Scan_MV_demo_measure

**Purpose**

These tags record counts and min and max values of several EIP IN data members.

The demo code expects a demo vision job to be loaded on the camera, and a demo target to be in the camera's field of view. The demo PLC code will operate without the demo vision job being loaded on the camera. However, the data records will not be valid.

The demo vision job has the following data members linked to certain job tools:

**IN**

**Bool1** = Measure status (pass/fail)

**Bool2** = Decode+Matchcode status (pass/fail)

**Bool3** = Blob count status (pass fail)

**Long1** = Blob count

**Float1** = Measure value

**String1** = Decode text

**OUT**

**Long101** = Blob count minimum count tolerance

**Long102** = Blob count maximum count tolerance

**Float101** = Measure lower tolerance

**Float102** = Measure upper tolerance

**String101** = Matchcode

Each tag set records the following data for each vision job tool result received in the Inspection report:

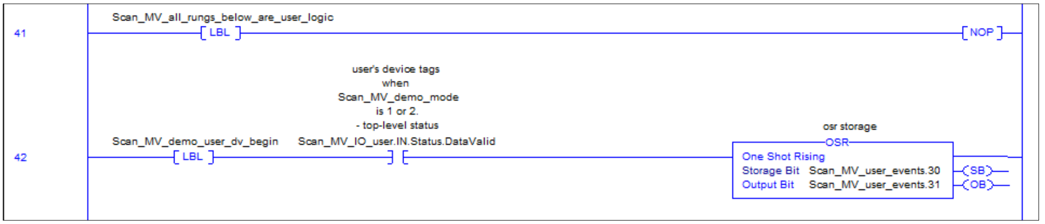
| **Tool Result** | **Record tag** | **EIP IN tag** |
|---|---|---|
| Measurement Status | Scan_MV_demo_measure.<br>bool = last status(pass/fail)<br>pass_count = count of passes<br>fail_count = count of fails | Scan_MV_user_IN.bool.bool1 |
| Measurement | Scan_MV_demo_measure.<br>float = last value<br>float_max = max value recorded<br>float_min = minimum value recorded | Scan_MV_user_IN.float.float1 |
| Decode Text Status (matchcode) | Scan_MV_demo_decode<br>bool = last status(pass/fail)<br>pass_count = count of passes<br>fail_count = count of fails | Scan_MV_user_IN.bool.bool2 |
| Decode Text | Scan_MV_demo_decode<br>string = text of the last barcode<br>decode attempt (null if noread) | Scan_MV_user_IO.string.string1 |
| Blob Status | Scan_MV_demo_blob<br>bool = last status(pass/fail)<br>pass_count = count of passes<br>fail_count = count of fails | Scan_MV_user_IN.bool.bool3 |
| Blob Count | Scan_MV_demo_blob<br>long = last value<br>long_max = max value recorded<br>long_min = minimum value recorded | Scan_MV_user_IN.long.long1 |

### User App Method

The user app can follow the demo code's usage of these tags for further application logic development.

During runtime, the user app can change the OUT data members, and observe the change in tool status after a new trigger.

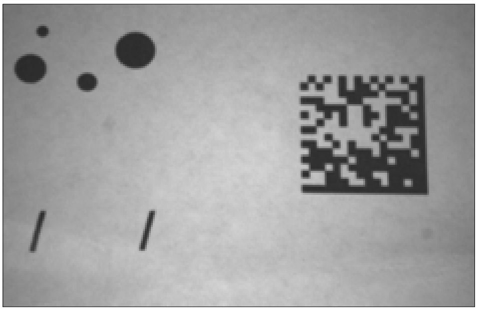
Specifically, the PLC integrator would typically modify the logic beginning at the following rungs:



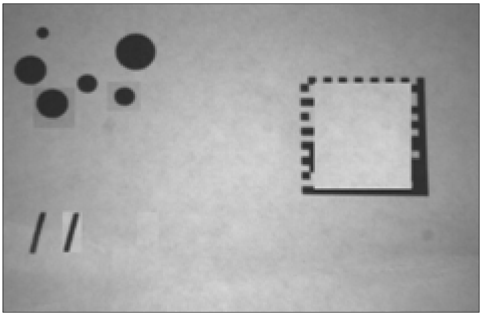

### Demo Target

The demo vision job uses predefined images. It is unnecessary to have the camera aimed at any specific target. If preferred, the job can be changed to enable the camera's image sensor. In this case, the demo targets should be printed approximately **2.5 inches** (**63 mm**) wide by **1.6 inches** (**40 mm**) tall, centered on white paper larger than the camera's field of view, and presented to the camera with the Data Matrix symbol on the right:

### "Pass" Image



**“Fail” Image**



**Demo Code Usage**

The user app example of the demo code watches for Data Valid. When it goes active, the user app example processes the user I/O data, updates each demo record with the results, then uses **ResetDataValid** to clear DataValid.

## Scan_MV_IO_internal, Scan_MV_ons_internal

**Purpose**

Used by the demo code to manage the camera.

**User App Method**

None. The user app must not attempt to read or write to this tag set.

**Demo Code Usage**

The demo code uses this tag set to abstract the on-the-wire control of the camera from the user app.

## Run the Camera: Runtime Operation of EtherNet/IP Demo

At this point in the evaluation, it is assumed that you have downloaded the demo vision job to the camera and that your PLC is running the EtherNet/IP demo code and exchanging data with the camera. The PLC can now parameterize, trigger and monitor the camera over EtherNet/IP.

7

CHAPTER 7

# Omron PLC Setup for EtherNet/IP Operation

This section describes how to set up an Omron PLC for EtherNet/IP operation using an Omron Microscan smart camera and CX-One software.

**Notes:**

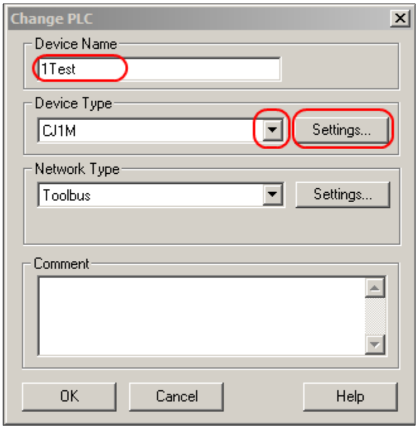
- The camera communications protocol must be enabled for EtherNet/IP before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the camera, and information about switching camera communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate EtherNet/IP communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.

# Setting Up an Omron PLC

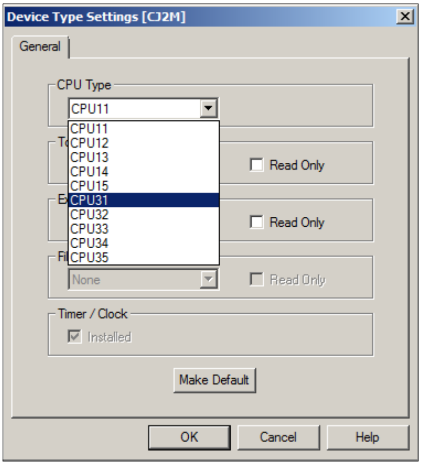
This section was created using the following Omron software and hardware:

- CX-Programmer version 9.43
- Network Configuration version 3.55
- PLC CJ2M CPU31

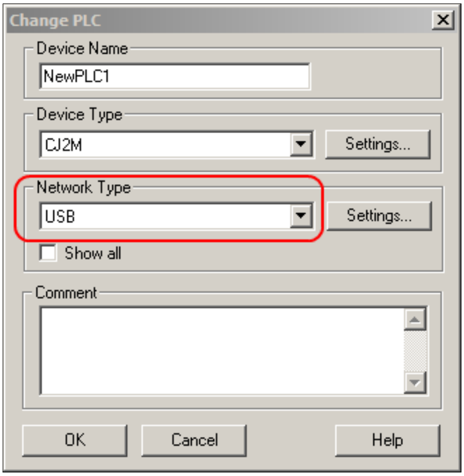
Start the CX-Programmer application and select menu item **File > New**. Enter the desired **Device Name**. Select the **Device Type** and then click **Settings** to the right of the Device Type menu.




In the **Device Type Settings** dialog, select the correct **CPU Type** and click **OK**.



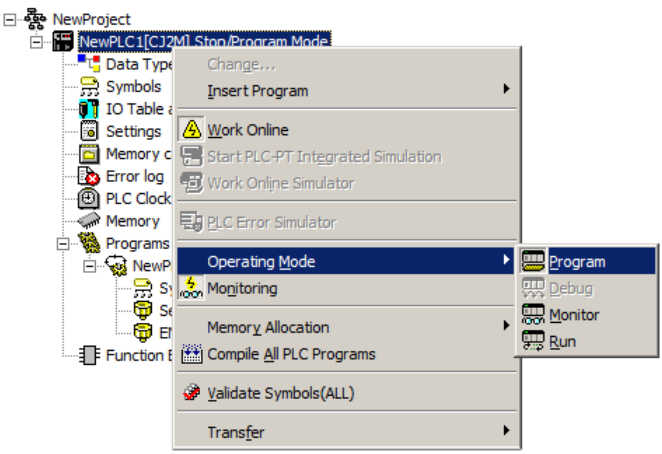
Select **USB** from the **Network Type** menu and click **OK**.



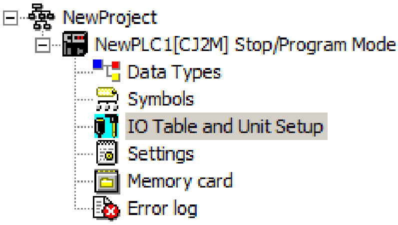

Connect to the PLC via USB connection. Select the menu item **PLC > Work Online** or click the online icon in the tool bar. When prompted, click **Yes** to complete the connection.


Once successfully connected, the background of the right pane will turn gray and the online icons in the ribbon will remain clicked.

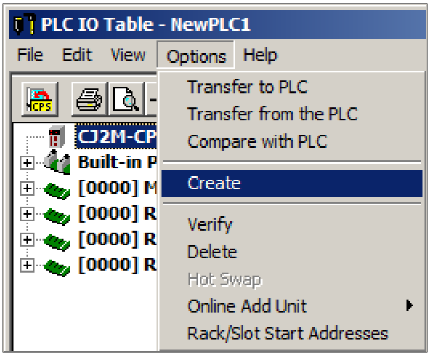
To complete the next steps the PLC must be in **Program Mode**. Right-click the PLC node in the tree view in the left pane and select **Operating Mode > Program**.



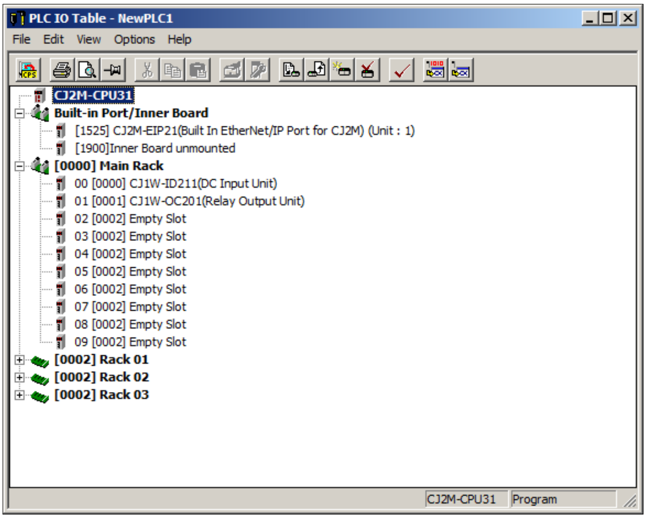
To register the I/O table, double-click **IO Table and Unit Setup**.



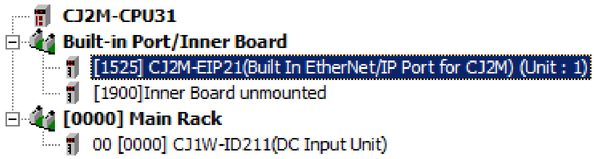

Select **Options > Create**.



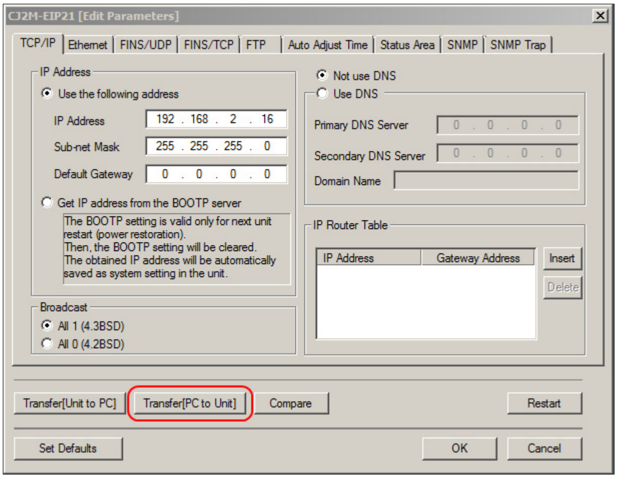

Click **Yes** at the I/O table creation prompt. Click **Yes** at the Initialize CPU bus settings prompt. Click **Transfer** at the transfer prompt. Click **OK** at the results prompt. The IO table will now be updated with the current PLC hardware settings.



To edit the EtherNet/IP items and mapping, double-click the EtherNet/IP node.




Input the desired IP settings and then click **Transfer [PC to Unit]**.

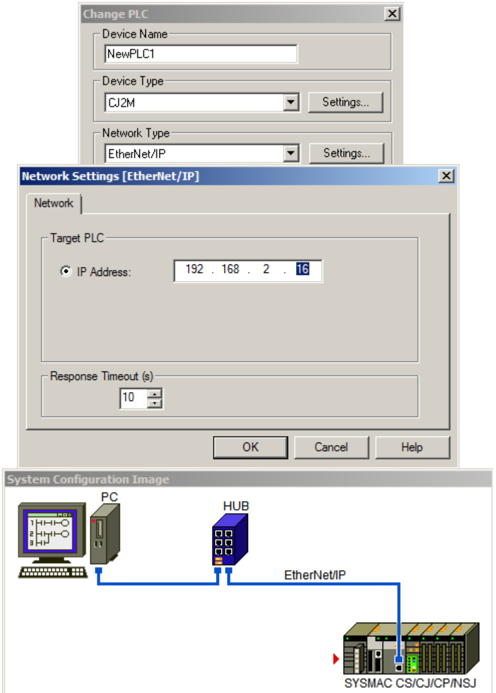



When prompted to transfer, click **Yes**. After the transfer prompt click **Close**. When prompted to restart the unit click **Yes**. Once the unit resets, click **OK** at the prompt. Close the IO edit dialog.

Physically power down the unit and adjust the rotary switches to match the last octet of the new IP address from above. Then power the unit back on.

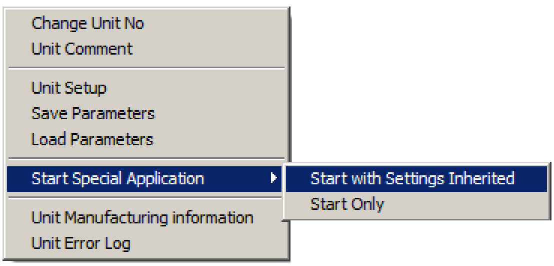
Set the CX-Programmer application to offline by selecting **PLC > Work Offline** or the online icon in the ribbon.

In the CX-Programmer application, double-click the PLC node. In the Network Type menu, select **EtherNet/IP**. Click **Settings** to the right of Network Type and enter the PLC's new IP address. Click the **OK** buttons to close the dialogs.



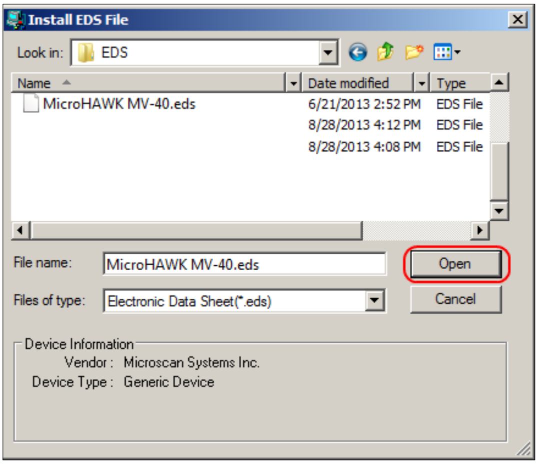

Set the CX-Programmer back to online. You will be prompted with a connection via EtherNet/IP. Click **Yes** to complete the connection.

Double-click the **IO Table and Unit Setup** node. Expand the **Built-In Port/Inner Board** node. Right-click and select **Start Special Application > Start with Settings Inherited**.

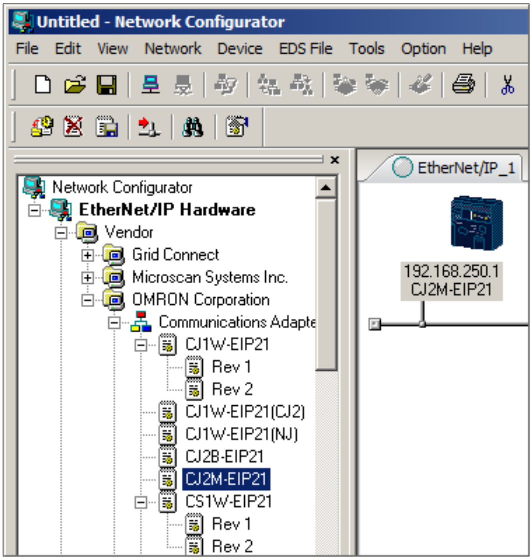


Select **Network Configurator** and click **OK**. Select port **TCP:2** and click **OK**. Then click **OK** for **EtherNet/IP_1** connection.

To install the EDS file, select **EDS File > Install**. Navigate to the EDS folder **C:\Microscan\Vscape\Firmware\eds\MicroHAWK** or **C:\Microscan\Vscape\Firmware\eds\HAWK**. Select the correct file and click **Open** to load the file. All other EDS files can be downloaded from **www.microscan.com**.

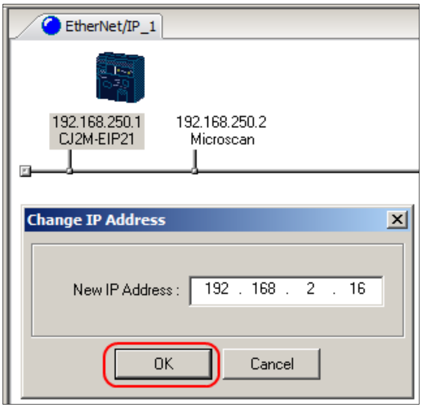



Expand the left tree view to open the **OMRON Corporation** files. Locate **CJ2M-EIP21** for this example and drag it to the line in the right pane.

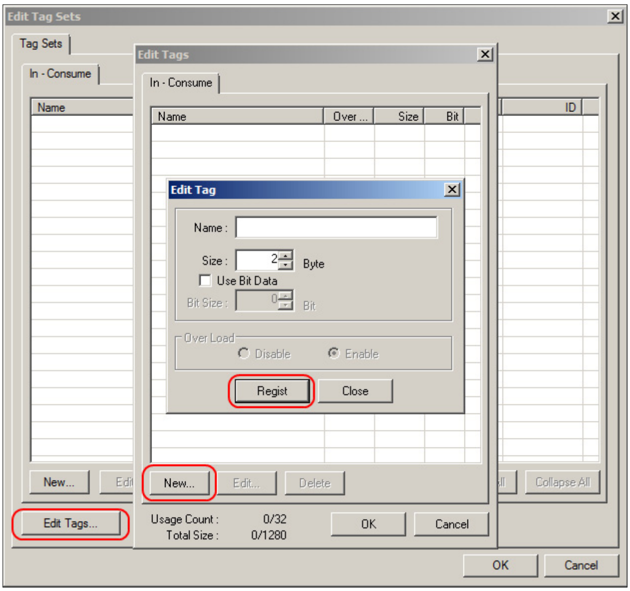


Expand the **Vendor Collection** node for the camera connected to the PLC and drag it to the line in the right pane.

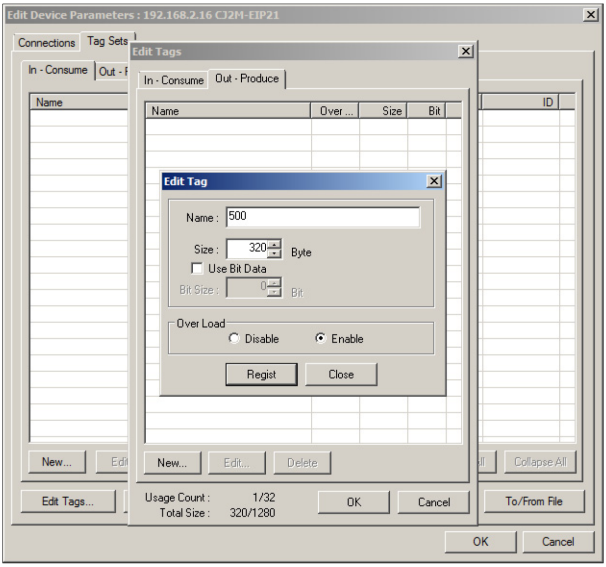
Right-click near the newly-added icons and select **Change IP Address**. Enter the IP address for the PLC and the camera or reader attached and click **OK**.



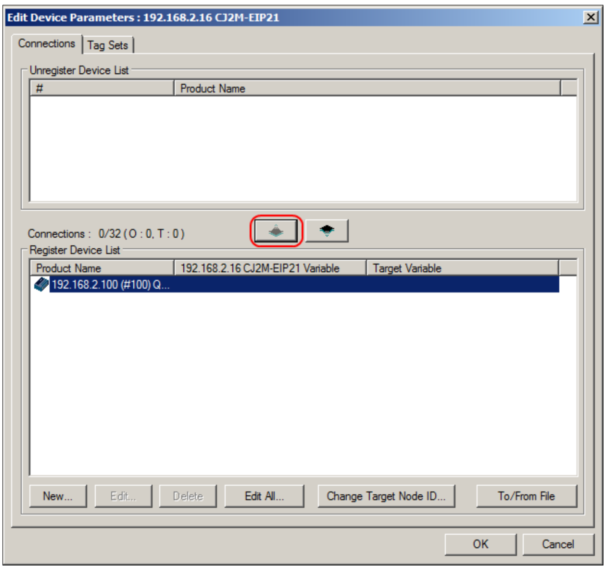

Double-click the PLC icon to edit the device parameters. This is where you will be linking and mapping the EtherNet/IP assembly data to the internal memory of the PLC. Select the **Tag Sets** tab. Select the **In - Consume** tab at the top. Click **Edit Tags** below. Then click **New** to edit/create a new tag. In this example we are naming this tag **300** for the peripheral memory linked to the input data. Select the size (**320 bytes**) for the entire input assembly. Click **Regist** then **Close** to continue.



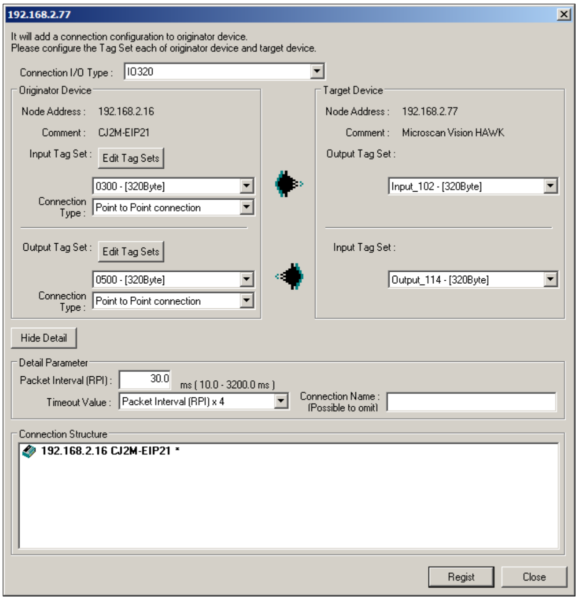
Select the **Out - Produce** tab and then click **New**. For the output assembly you are going to map to peripheral memory address **500** with **320 bytes**. Click **Regist** and then **Close**. Click **OK** on the **Edit Tags** dialog. When prompted to register new tags click **Yes**.



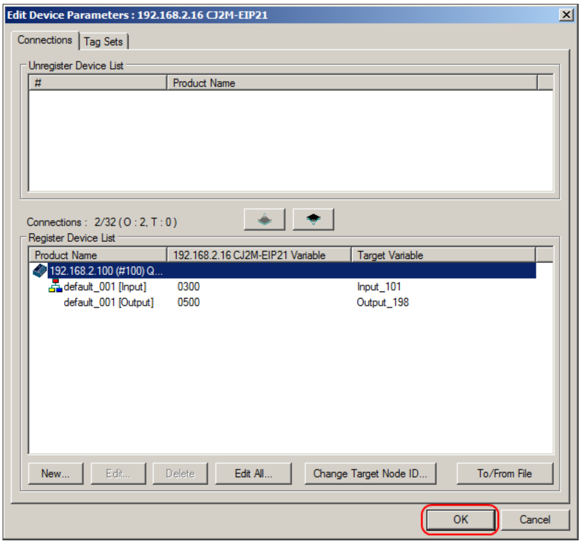

Click the **Connections** tab on the **Edit Device Parameters** dialog. Click the download button in the middle to register the device.



In the lower pane, double-click the PLC in the registered device list. This will open the linking dialog. If there are multiple connection types, they can be selected from the **Connection I/O Type** menu. In the **Originator Device** section, select the **Input Tag** then the **Output Tag**. Adjust the RPI if needed. When done, click **Regist** and then **Close**.

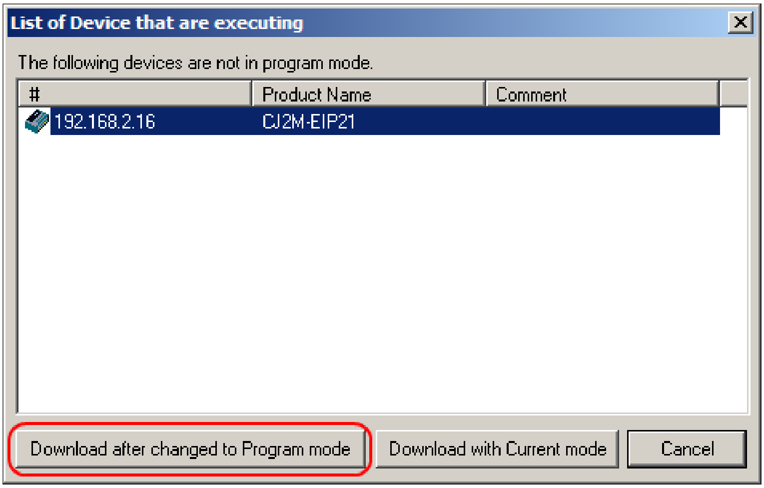



The registered device will now display the tags that are linked. Click **OK** to continue.



Download the new tags and links to the PLC by selecting **Device > Parameter > Download** or by clicking the download icon in the ribbon. Click **Yes** when prompted to download.


When the **List of Devices That Are Executing** dialog appears, select the PLC and click **Download after Changed to Program Mode**. When prompted to return the state, click **Yes** to continue.




Select **Network > Check Connection** in the Network Configurator to ensure that there are no connection problems.

CHAPTER 8

# Using PROFINET I/O

This section provides information necessary for using a Omron Microscan smart camera in a PROFINET I/O environment.

**Notes:**

- The camera communications protocol must be enabled for PROFINET I/O before it can be used in this environment. Refer to **Chapter 1**, **Enabling Industrial Protocols**, for information about enabling communications protocols for the camera, and information about switching camera communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate PROFINET I/O communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.
  **Important:** PROFINET I/O allows the IP address and subnet mask of the camera to be assigned automatically by the PLC. In order to allow this, the camera may not have an assigned IP address on reboot until the PLC is set to Run mode. During this time, the camera will not be visible on the network for AutoVISION or Visionscape FrontRunner.

# PROFINET I/O

## PROFINET I/O Identity

### Vendor ID

Microscan's Vendor ID is **0x0257**.

### Device ID

MicroHAWK: **0x7000**

## Connection Properties: RT Cyclic Messaging

Odd slot numbers are input to the PLC, even slot numbers are output from the PLC.

Maximum data size in either direction is 518 bytes. The data size can be reduced by removing slots that are not used.

Cycle update time for MicroHAWK: 8ms

### Slot/Subslot Layout Descriptions

| Slot | Dir | Bytes | Name | Description |
|---|---|---|---|---|
| 1 | In | 2 | STATUS | Status register of the camera, each bit of this register represents a different state item. See *Camera Status Register* for bit descriptions |
| 3 | In | 2 | ECHO | This 16 bit word value reflects back to the PLC the value that the PLC wrote to the output assembly ECHO register. The PLC can verify the output assembly has been written to the camera when this value matches the written value. |
| 5 | In | 4 | CmdCodeRslt | When Status.ExeCmdAck goes active in response to Control.ExeCmd, CmdCodeRslt reflects the result of the command invoked by Control.CmdCode. See *CmdCodeRslt* for definitions. |
| 7 | In | 4 | CmdRet | When Status.ExeCmdAck goes active in response to Control.ExeCmd, CmdRet contains the data returned from the command invoked by Control.CmdCode. See *CmdRet* for definitions. |
| 9 | In | 2 | State | Device State register. Depending on the current state of the camera, certain STATUS and CONTROL features may or may not be operational. See *State* for |

| | | | | |
|---|---|---|---|---|
| | | | | definitions. |
| 2 | Out | 2 | CONTROL | Control register of camera. Each bit of this register represents a different status item. See *Camera Control Register* for bit descriptions |
| 4 | Out | 2 | ECHO | This 16 bit value is reflected back to the PLC in the input assembly ECHO register. The PLC can verify the output assembly has been written to the camera when the input assembly matches this written value. |
| 6 | Out | 4 | CmdCode | Specifies the process invoked in the camera when Control.ExeCmd goes active. See *CmdCode* for definitions. |
| 8 | Out | 4 | CmdArg | Additional argument data for the CmdCode. See *CmdArg* for definition. |
| 11 | In | 2 | VIO | Each bit reflects the state of a virtual IO point. The least significant bit reflects vio point 145, the most significant bit vio point 160 |
| 10 | Out | 2 | VIO | Each bit reflects the state of a virtual IO point. The least significant bit reflects vio point 129, the most significant bit is vio point 144 |
| 13 | In | 8 | bool1-64 | Each bit represents a bool value. The least significant bit of byte 0 reads the value of bool1. The most significant bit of byte 7 reads bool64. |
| 12 | Out | 8 | Bool101-164 | Each bit represents a bool value. The least significant bit of byte 0 writes the value of bool101. The most significant bit of byte 7 writes bool164. |
| 15 | In | 20 | int1-10 | Each pair of sequential bytes represents a 16 bit signed integer value. The 20 bytes represent 10 integers. From bytes 0-1 for the value of int1 through bytes 18-19 for the value of int10. |

| | | | | |
|---|---|---|---|---|
| 14 | Out | 20 | int101-110 | Each pair of sequential bytes represents a 16 bit signed integer value. The 20 bytes represent 10 integers. From bytes 0-1 to write the value of int101 through bytes 18-19 for the value of int110. |
| 17 | In | 64 | long1-16 | Each group of 4 bytes represents a 32 bit signed integer value. The 64 bytes represent 16 long integers. From bytes 0-3 for the value of long1 through bytes 60-63 for the value of long16. |
| 16 | Out | 64 | long101-116 | Each group of 4 bytes represents a 32 bit signed integer value. The 64 bytes represent 16 long integers. From bytes 0-3 for the value of long101 through bytes 60-63 for the value of long116. |
| 19 | In | 96 | float1-24 | Each group of 4 bytes represents a 32 bit signed integer value. The 96 bytes represent 24 long integers. From byte offsets 0-1 for the value of float1 through byte offsets 92-95 for the value of float24. |
| 18 | Out | 96 | float101-124 | Each group of 4 bytes represents a 32 bit signed integer value. The 96 bytes represent 24 long integers. From bytes 0-3 for the value of float101 through bytes 92-95 for the value of float124. |
| 21 | In | 96 | string1 | These 96 bytes can store a string of up to 94, 8 bit characters, with the first 2 bytes containing the storage length and string length values. |
| 20 | Out | 96 | string101 | These 96 bytes can store a string of up to 94, 8 bit characters, with the first 2 bytes containing the storage length and string length values. |
| 23 | In | 96 | string2-string7 | 6 consecutive strings, each of 32 bytes can store a string of up to 30, 8 bit characters, with the first 2 bytes of each string group containing the storage length and string length values. |
| 22 | Out | 96 | string102-string107 | 6 consecutive strings, each of 32 bytes can store a string of up to 30, 8 bit characters, with the first 2 bytes of each string group containing the storage length and string length values. |

# Slot Data Layout Diagrams

### PLC Input

| Slot | Byte Offset | Data |
|---|---|---|
| 1 | 0 | STATUS |
| 3 | 0 | Echo In |
| 5 | 0 | CMD CODE RSLT |
| 7 | 0 | CMD RET |
| 9 | 0 | STATE |
| 11 | 0 | VIO 145.. 160 |
| 13 | 0 | bool 1.. 16 |
| | 2 | bool 17.. 32 |
| | 4 | bool 33.. 48 |
| | 6 | bool 49.. 64 |
| 15 | 0 | int 1 |
| | 2 | int 2 |
| | 4 | int 3 |
| | 6 | int 4 |
| | 8 | int 5 |
| | 10 | int 6 |
| | 12 | int 7 |
| | 14 | int 8 |
| | 16 | int 9 |
| | 18 | int 10 |
| 17 | 0 | long 1 |
| | 4 | long 2 |
| | 8 | long 3 |
| | 12 | long 4 |
| | 16 | long 5 |
| | 20 | long 6 |
| | 24 | long 7 |
| | 28 | long 8 |
| | 32 | long 9 |
| | 36 | long 10 |
| | 40 | long 11 |
| | 44 | long 12 |
| | 48 | long 13 |
| | 52 | long 14 |
| | 56 | long 15 |
| | 60 | long 16 |

### PLC Output

| Slot | Byte Offset | Data |
|---|---|---|
| 2 | 0 | CONTROL |
| 4 | 0 | Echo Out |
| 6 | 0 | CMD CODE |
| 8 | 0 | CMD ARG |
| | | |
| 10 | 0 | VIO 129.. 144 |
| 12 | 0 | bool 101.. 116 |
| | 2 | bool 117.. 132 |
| | 4 | bool 133.. 148 |
| | 6 | bool 149.. 164 |
| 14 | 0 | int 101 |
| | 2 | int 102 |
| | 4 | int 103 |
| | 6 | int 104 |
| | 8 | int 105 |
| | 10 | int 106 |
| | 12 | int 107 |
| | 14 | int 108 |
| | 16 | int 109 |
| | 18 | int 110 |
| 16 | 0 | long 101 |
| | 4 | long 102 |
| | 8 | long 103 |
| | 12 | long 104 |
| | 16 | long 105 |
| | 20 | long 106 |
| | 24 | long 107 |
| | 28 | long 108 |
| | 32 | long 109 |
| | 36 | long 110 |
| | 40 | long 111 |
| | 44 | long 112 |
| | 48 | long 113 |
| | 52 | long 114 |
| | 56 | long 115 |
| | 60 | long 116 |

**PLC Input**

| Slot | Byte Offset | Data |
|---|---|---|
| 19 | 0 | float 1 |
| | 4 | float 2 |
| | 8 | float 3 |
| | 12 | float 4 |
| | 16 | float 5 |
| | 20 | float 6 |
| | 24 | float 7 |
| | 28 | float 8 |
| | 32 | float 9 |
| | 36 | float 10 |
| | 40 | float 11 |
| | 44 | float 12 |
| | 48 | float 13 |
| | 52 | float 14 |
| | 56 | float 15 |
| | 60 | float 16 |
| | 64 | float 17 |
| | 68 | float 18 |
| | 72 | float 19 |
| | 76 | float 20 |
| | 80 | float 21 |
| | 84 | float 22 |
| | 88 | float 23 |
| | 92 | float 24 |

**PLC Output**

| Slot | Byte Offset | Data |
|---|---|---|
| 18 | 0 | float 101 |
| | 4 | float 102 |
| | 8 | float 103 |
| | 12 | float 104 |
| | 16 | float 105 |
| | 20 | float 106 |
| | 24 | float 107 |
| | 28 | float 108 |
| | 32 | float 109 |
| | 36 | float 110 |
| | 40 | float 111 |
| | 44 | float 112 |
| | 48 | float 113 |
| | 52 | float 114 |
| | 56 | float 115 |
| | 60 | float 116 |
| | 64 | float 117 |
| | 68 | float 118 |
| | 72 | float 119 |
| | 76 | float 120 |
| | 80 | float 121 |
| | 84 | float 122 |
| | 88 | float 123 |
| | 92 | float 124 |

**PLC Input**

| Slot | Byte Offset | Data |
|---|---|---|
| | 0 | 94 |
| | 1 | <str 1 len> |
| 21 | 2 | string 1 |
| | | |
| | 95 | |
| | 0 | 30 |
| | 1 | <str 2 len> |
| | 2 | string 2 |
| | | |
| | 31 | |
| | 32 | 30 |
| | 33 | <str 3 len> |
| | 34 | string 3 |
| | | |
| | 63 | |
| | 64 | 30 |
| | 65 | <str 4 len> |
| 23 | 66 | string 4 |
| | | |
| | 95 | |
| | 96 | 30 |
| | 97 | <str 5 len> |
| | 98 | string 5 |
| | | |
| | 127 | |
| | 160 | 30 |
| | 161 | <str 6 len> |
| | 162 | string 6 |
| | | |
| | 191 | |

**PLC Output**

| Slot | Byte Offset | Data |
|---|---|---|
| | 0 | 94 |
| | 1 | <str 101 len> |
| 20 | 2 | string 101 |
| | | |
| | 95 | |
| | 0 | 30 |
| | 1 | <str 102 len> |
| | 2 | string 102 |
| | | |
| | 31 | |
| | 32 | 30 |
| | 33 | <str 103 len> |
| | 34 | string 103 |
| | | |
| | 63 | |
| | 64 | 30 |
| | 65 | <str 104 len> |
| 22 | 66 | string 104 |
| | | |
| | 95 | |
| | 96 | 30 |
| | 97 | <str 105 len> |
| | 98 | string 105 |
| | | |
| | 127 | |
| | 160 | 30 |
| | 161 | <str 106 len> |
| | 162 | string 106 |
| | | |
| | 191 | |

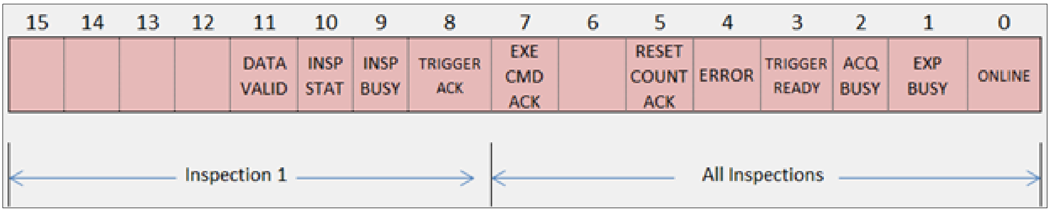
# STEP 7 PLC Slot Layout for MicroHAWK

(2) MicroHAWK

| Slot | Module | Order number | I address | Q address | Diagnostic address: |
|---|---|---|---|---|---|
| 0 | MicroHAWK | 7xxx-xxxx-xxxx- | | | 2040* |
| X1 | Interface | | | | 2039* |
| X1 P | Port 1 | | | | 2038* |
| 1 | Status | | 12...13 | | |
| 2 | Control | | | 12...13 | |
| 3 | Echo In | | 743...744 | | |
| 4 | Echo Out | | | 742...743 | |
| 5 | Cmd Code Rslt | | 735...738 | | |
| 6 | Cmd Code | | | 738...741 | |
| 7 | Cmd Ret | | 739...742 | | |
| 8 | Cmd Arg | | | 734...737 | |
| 9 | State | | 745 | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | | | | | |
| 20 | | | | | |
| 21 | | | | | |
| 22 | | | | | |
| 23 | | | | | |

## Status: Camera Status Register (16-bit)

Each bit of this register represents a signal that displays the camera's operational status. A high value of **1** indicates that the signal is **active** (true).




| Bit | Name | Description |
|---|---|---|
| 0 | ONLINE | Inspections are running |
| 1 | EXP BUSY | The camera is busy capturing an image. The camera should not be triggered or the part under inspection moved during this time if illuminated. |
| 2 | ACQ BUSY | The camera is busy acquiring an image. The camera cannot be triggered while busy. |
| 3 | TRIGGER READY | The camera is ready to be triggered. This is equivalent to ONLINE == 1 and ACQ BUSY == 0. |
| 4 | ERROR | An error has occurred. Set the RESET ERROR control bit high to clear. |
| 5 | RESET COUNT ACK | This bit mirrors the RESET COUNT control bit. The PLC can be certain the reset command was received by the camera when this goes high. The PLC can then bring the RESET COUNT control signal back low. |
| 7 | EXE CMD ACK | This bit mirrors the EXE CMD control bit. |
| 8 | TRIGGER ACK | This bit mirrors the TRIGGER control bit. |
| 9 | INSP BUSY | This bit is high when inspection 1 is busy processing an image. |
| 10 | INSP STAT | This bit represents the inspection 1 status result. It is 1 if the inspection passes. It is only valid when DataValid goes high. |
| 11 | DATA VALID | This bit goes high when inspection 1 is complete. The PLC should clear this signal by setting RESET DV high once it has read results. |

## CmdCodeRslt (32-bit)

The value of CmdCodeRslt is only valid when **ExeCmdAck** is active (**1**), in response to **ExeCmd** being active.

| CmdCodeRslt value (base 16 hex) | Meaning |
|---|---|
| 0x0000_0000 | Success |
| 0x0100_0000 | Fail.<br>Possible reasons:<br>Camera under PC control.<br>Job cannot be changed. |
| 0x0200_0000 | Fail: No Job in slot. |
| 0x0300_0000 | Fail: Unknown cmd. |

## CmdRet (32-bit)

The value of **CmdRet** is only valid when **ExeCmdAck** is active (**1**), in response to **ExeCmd** being active, and **CmdCodeRslt** is **0** (Success). The following chart shows which **CmdCodes** return data in the **CmdRet** register.

| CmdRet value (32 bit) | Associated CmdCode | Meaning |
|---|---|---|
| 0 | 0x1000_0000<br>to<br>0x1300_0000<br>(Job Change type) | Na |
| 1 – 255 | 0x1800_0000<br>(Query Active Job Slot) | Active Job Slot # |

## State (16-bit)

**State** reflects the following operational condition of the camera:

| State value (16 bit) | Meaning | Typical action required by the client (PLC), or system operator |
|---|---|---|
| 0 | Offline | Perform job change or put camera online. |
| 1 | Online | Normal runtime operation: Monitor TriggerReady and DataValid signals. Trigger the camera. |
| 2 | Changing Vision Job | If camera is under pc control:<br>Wait until State changes to Offline or Online.<br><br>If PLC is controlling the job change:<br>Use ExeCmd, CmdCode, ExeCmdAck, and CmdCodeRslt to complete the operation. |
| 3 | Booting* | Wait for camera to transition to Online or Offline. |
| 4 | Empty (no Vision Job) | Load a new job from AutoVISION or Front Runner. |

*Booting (3) State: This will rarely be seen by the PLC.

The value of State determines which Control and Status signals are available:

| Control/Status Signal | State 0 (Offline) | 1 (Online) | 2 (Job Change) | 3 (Booting) | 4 (Empty) |
|---|---|---|---|---|---|
| Control.GO ONLINE | Y | | | | |
| “.GO OFFLINE | | Y | | | |
| “.RESET ERROR | | | | | |
| “.RESET COUNT | Y | Y | | | |
| “.EXE CMD | Y | Y | Y | | Y |
| “.TRIGGER | | Y | | | |
| “.RESET DATA VALID | | Y | | | |
| Status.ONLINE | Y | Y | Y | Y | Y |
| “.ERROR | | | | | |
| “.RESET COUNT ACK | Y | Y | | | |
| “.EXE CMD ACK | Y | Y | Y | | Y |
| “.EXP BUSY | | Y | | | |
| “.ACQ BUSY | | Y | | | |
| “.TRIGGER READY | | Y | | | |
| “.TRIGGER ACK | | Y | | | |
| “.INSP BUSY | | Y | | | |
| “.INSP STAT | | Y | | | |
| “.DATA VALID | | y | | | |

Where:
Y = Signal is valid for this State
Empty cell = Signal is not valid for this State

## VIO Output Register Bits

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v144 | v143 | v142 | v141 | v140 | v139 | v138 | v137 | v136 | v135 | v134 | v133 | v132 | v131 | v130 | v129 |

## VIO Input Register Bits

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| v160 | v159 | v158 | v157 | v156 | v155 | v154 | v153 | v152 | v151 | v150 | v149 | v148 | v147 | v146 | v145 |

## Control: Camera Control Register (16-bit)

Each bit of this register controls a function on the camera. Transitions from a low state of **0** to a high state of **1** initiate the associated operation. The PLC should return the state of the control bit back to **0** after it has acknowledged the camera has processed the control. Unused bits should remain **0**.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | RESET DATA VALID | | | TRIGGER | EXE CMD | | RESET COUNT | RESET ERR | | | GO OFFLINE | GO ONLINE |
| Inspection 1 | | | | | | | | All Inspections | | | | | | | |

| Bit | Name | Description |
|---|---|---|
| 0 | GO ONLINE | Start all inspections running |
| 1 | GO OFFLINE | Stop all inspections |
| 4 | RESET ERROR | Reset ERROR in the Status register |
| 5 | RESET COUNT | Reset all inspection counts |
| 7 | EXE CMD | Execute the command specified by Control.CmdCode |
| 8 | TRIGGER | Trigger Inspection 1. The inspection must be configured for a triggered image acquisition. |
| 11 | RESET DATA VALID | Reset the Data Valid signal of the Status register |

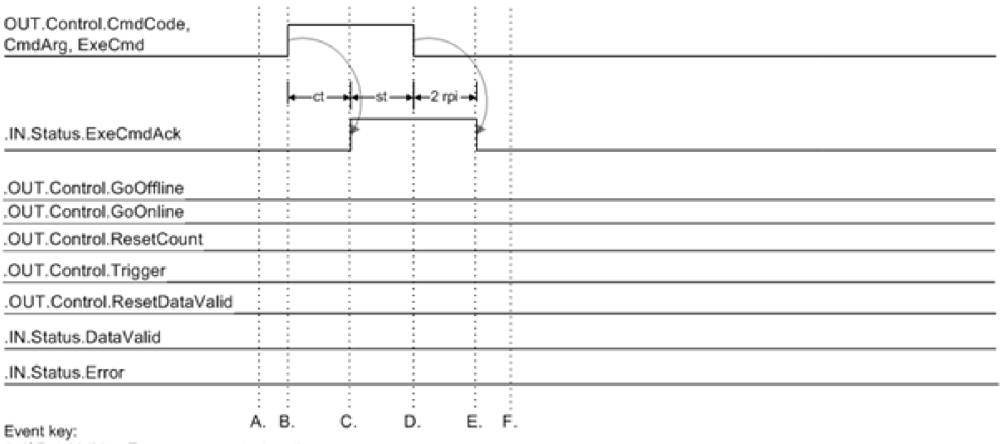
## CmdCode and CmdArg (32-bit)

Specifies the process invoked in the camera when **Control.ExeCmd** goes active.

### *List of Available CmdCodes and Associated CmdArg*

| CmdCode value | CmdArg | Operations performed |
|---|---|---|
| 0x1000_0000 | Job Slot (1-255) | Go Offline, Load job from specified slot |
| 0x1100_0000 | Job Slot (1-255) | Go Offline, Load job from specified slot, Go Online |
| 0x1200_0000 | Job Slot (1-255) | Go Offline, Load job from specified slot, Make it the boot job |
| 0x1300_0000 | Job Slot (1-255) | Go Offline, Load job from specified slot, Make it the boot job, and Go Online |
| 0x1800_0000 | na | Query active job slot. *CmdRet* will contain the active job slot number when the operation is done. |

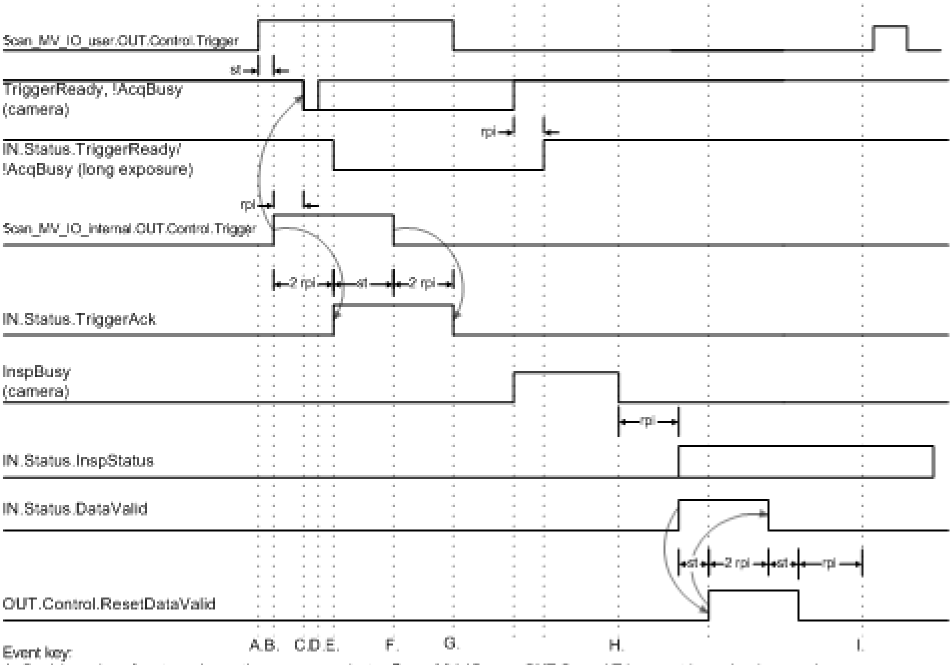
### *CmdCode and ExeCmd Operation*




Event key:
A. If DataValid or Error are present, clear them.
Set the following control signals idle, and keep them idle while the command is processed by the camera:
GoOffline, GoOnline, Trigger, ResetDataValid, ResetCount, ResetError.
If the command operation is a job change, populate the output tags required to configure the new job (bool, int, long, float, string).
B. Populate CmdCode and CmdArg, then activate ExeCmd.
C. Camera executes the command (may take up to a minute). While processing a Job Change command, State will be 2. Camera activates ExeCmdAck when it is done processing the command.
D. When the PLC sees an active ExeCmdAck, verify CmdCodeRslt is 0, and Error is 0. Process CmdRet if needed, then clear ExeCmd.
E. Camera clears ExeCmdAck when ExeCmd goes inactive. When ExeCmdAck goes inactive, CmdCodeRslt and CmdRet are no longer valid, and it may take a few seconds for the camera State and Online signals to settle to a final value (typically Online or Offline).
F. Camera can now be put online and triggered.

Notes:
st = PLC program scan time
ct = Command processing time in the camera. May take up to a minute for some commands.
rpi = Requested Packet Interval. Configured in the plc's EIP module connection properties. Allowed rpi is 10 ms to 3.2 s.
All signals represent the state of plc tags.

## *PROFINET I/O Control/Status Signal Operation*




Event key:
A. On rising edge of system trigger, the user app activates Scan_MV_IO_user.OUT.Control.Trigger to trigger the demo code.
B. Demo code detects rising edge of Scan_MV_IO_user.OUT.Control.Trigger, and if the camera is ready, sends a trigger to the camera.
C. Camera acquisition begins (may be delayed by one rpi).
D. If the camera's exposure time is shorter than the rpi, no change will be seen in TriggerReady and AcqBusy plc IN tags.
E. Camera firmware acks the trigger. The demo code may not see the ack until two rpi after the trigger was sent (event B).
F. Demo code detects TriggerAck and clears the Trigger.
G. Demo code detect falling edge of TriggerAck and clears the user Trigger.
H. Camera internal signal DataValid will go high when InspBusy goes low
I. Plc logic must delay one rpi time before re-asserting ResetDataValid

Notes:
1. The chart shows the workings of the Trigger and ResetDataValid Control signals, and the TriggerAck and DataValid Status signals.
2. st = plc program scan time
3. rpi = Requested Packet Interval. Configured in the plc's EIP module connection properties. Allowed rpi is 10 ms to 3.2 s.
4. All signals represent the state of plc tags, except where noted as "(camera)". The cam signals shown are visible in the EIP interface, but the state of the plc tags and internal firmware signals will be different for at least one or two requested packet intervals (rpi).
5. The plc is running the demo code distributed with the camera. The demo code and user app use the Scan_MV_IO_user tag set as the primary control, status, and data interface for the user app. All signal operations are still true even if the plc demo code is not used.
6. TriggerReady/!AcqBusy: Camera exposure times can range from less than 1 ms, up to 100 ms.

9

CHAPTER 9

# PROFINET I/O Blob Count Demo Using TIA Portal V13 and MicroHAWK

This section describes how to use Omron Microscan demo PLC code with a vision job and camera target. The PROFINET I/O demo files can be found where AutoVISION is installed, in the folder: **C:\Microscan\Tutorials and Samples\MicroHAWK\PROFINET demo\TIAPortal_Demos\BlobDemo**. Open **Blob_Demo.avp** with AutoVISION and download them to the camera.

**Notes:**

- The camera communications protocol must be enabled for PROFINET I/O before it can be used in this environment. Refer to **Chapter 1, Enabling Industrial Protocols**, for information about enabling and switching communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate PROFINET I/O communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.
**Important:** PROFINET I/O allows the IP address and subnet mask of the camera to be assigned automatically by the PLC. In order to allow this, the MicroHAWK may not have an assigned IP address on reboot until the PLC is set to Run mode. During this time, the MicroHAWK will not be visible on the network for AutoVISION or Visionscape FrontRunner.

# Overview

In this demonstration, you will learn how to load a saved job into the camera, establish connectivity via PROFINET I/O to a Siemens ET200SP CPU PLC, and run some example programs that interface with the camera. While evaluating PROFINET I/O capabilities, you will:

- **Begin with AutoVISION.**

Open a sample demo vision job in AutoVISION and use the Try Out feature to learn what to expect from the camera before it is connected to the PLC.

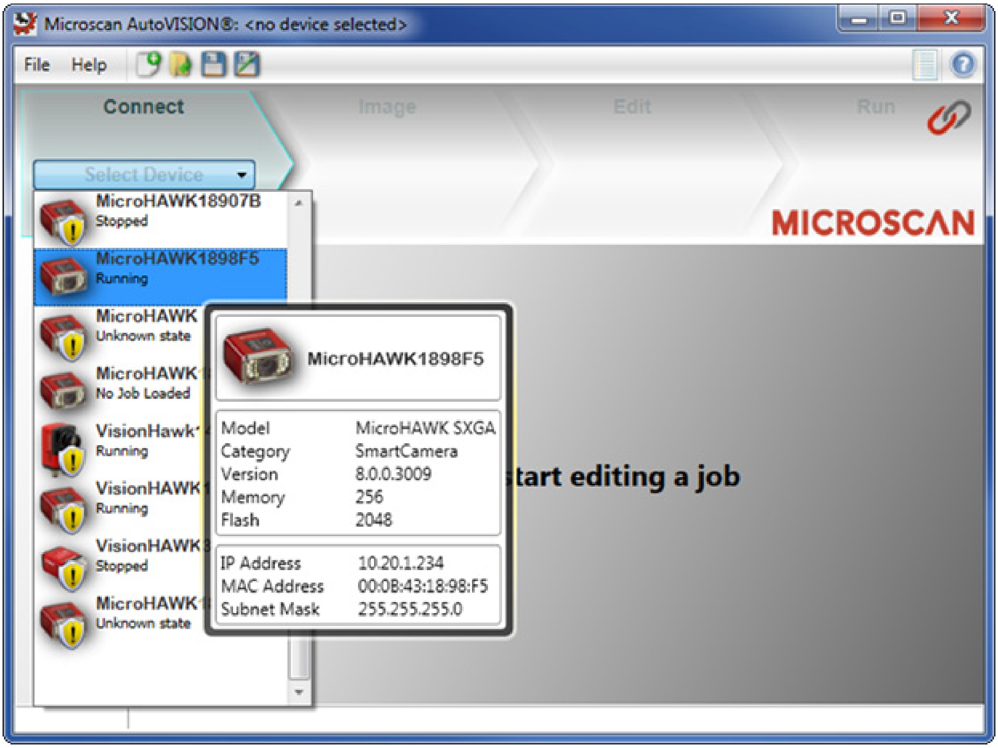
- **Prepare the PLC.**

Integrate the camera into the PLC environment with TIA Portal software and the GSD file.

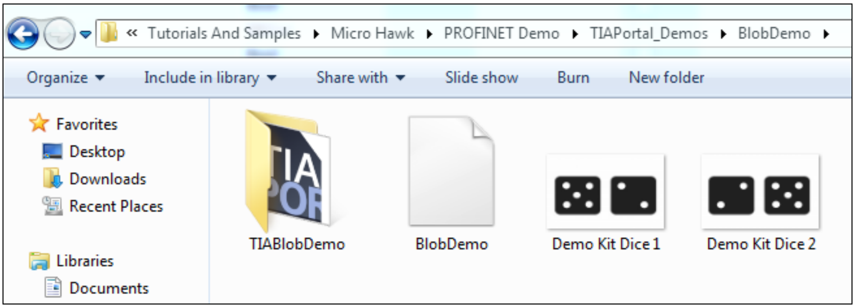
- **Run the camera.**

Trigger the camera while it is online with the PLC and observe changes in the Inspection status as the PLC reconfigures the vision job's parameters. Prior to starting AutoVISION, make sure the camera is either connected to the PLC or both PLC and camera are on the same physical network. Ensure that the PC, PLC, and camera have the same network class and corresponding subnet addresses.

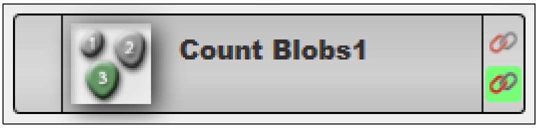
Open AutoVISION and select the camera.



From the Image view, click the **Load a Job** button. Then navigate to **C:\Microscan\Vscape\Tutorials And Samples\MicroHAWK\PROFINET demo\TIAPortal_Demos\BlobDemo**. Select **BlobDemo.avp**.




The demo job will include one tool: **Count Blobs**.

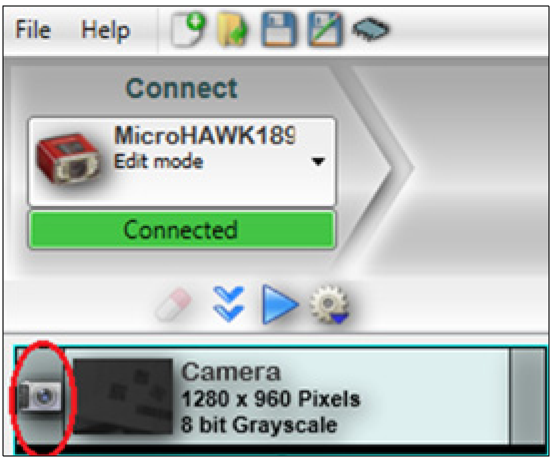



The Blob tool data item linked to the PROFINET I/O structure as shown here:

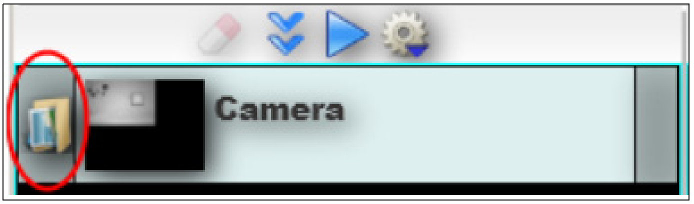
| Tool Result | PLC IN |
|---|---|
| Number of parts | "UserData".NumberOfParts (%DB2.DBW0) |

This data is transferred cyclically between the camera and PLC.

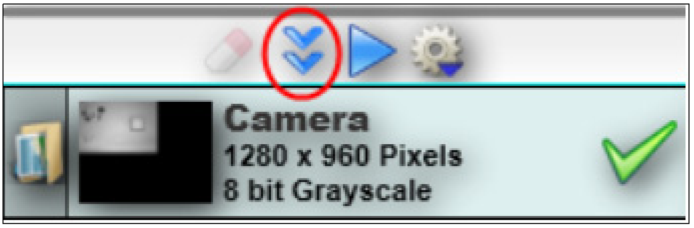
Once the job has loaded, the next step is to link the pre-saved images on the local PC. On the **Camera** button, click the far left icon to select and load an image.



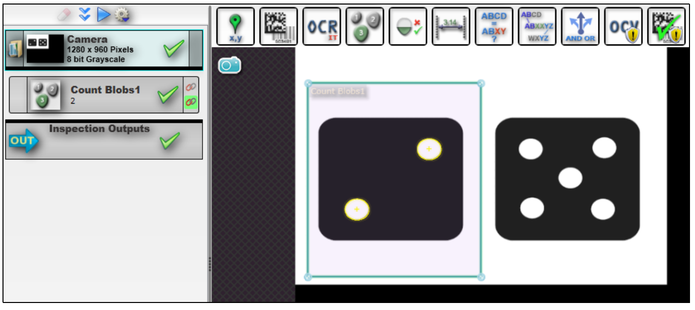
A file browser will open. Then navigate to the same folder where the demo job was loaded PROFINET I/O Demo. If the images are located, the icon will change from a camera to a folder.

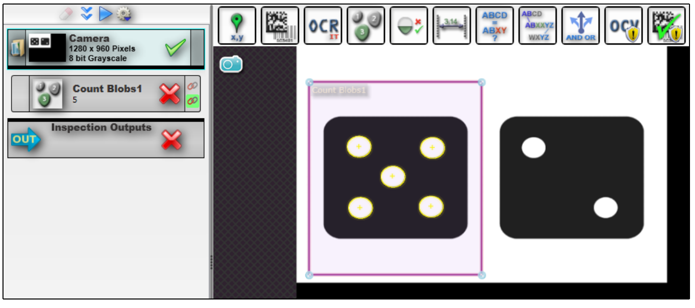



By clicking the **Try Job Once** icon, the application will cycle through the entire job with the loaded image.




There are two images on file. One will pass the inspection and one will fail.



Now click the **Run** button on the top ribbon. This will download the job to the camera. At this point the job is ready to run and can be tested. However, in order to run the job change demo this job needs to be loaded into **Slot 1**. Click back to the **Edit** view from the top ribbon bar.

Click the slot icon and select Slot 1 or **New Slot** if no slots are currently in use.

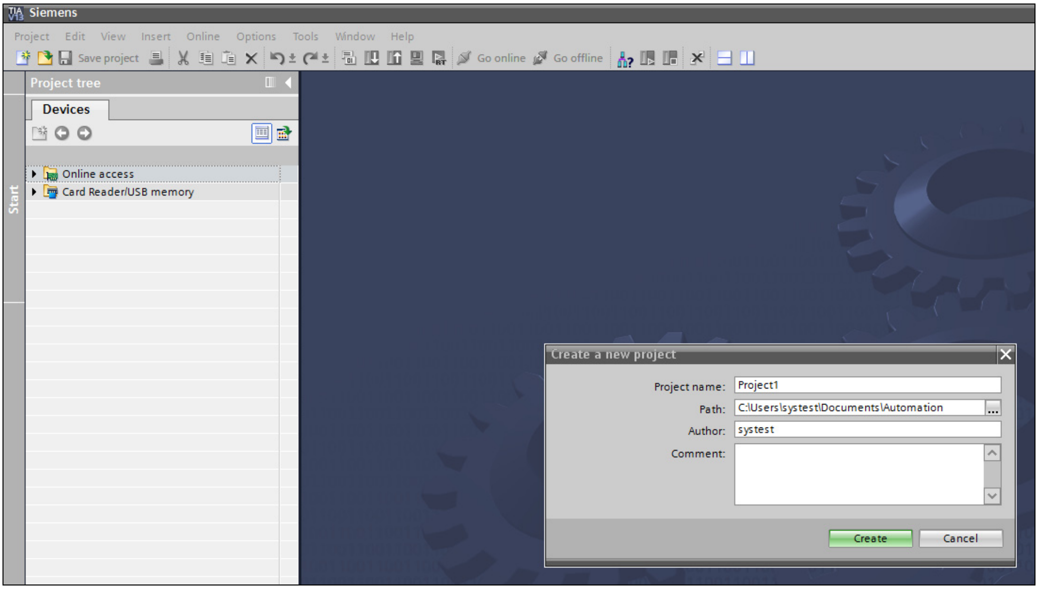

Now the job and images will be saved to the flash memory of the camera.

## TIA Portal V13 Setup

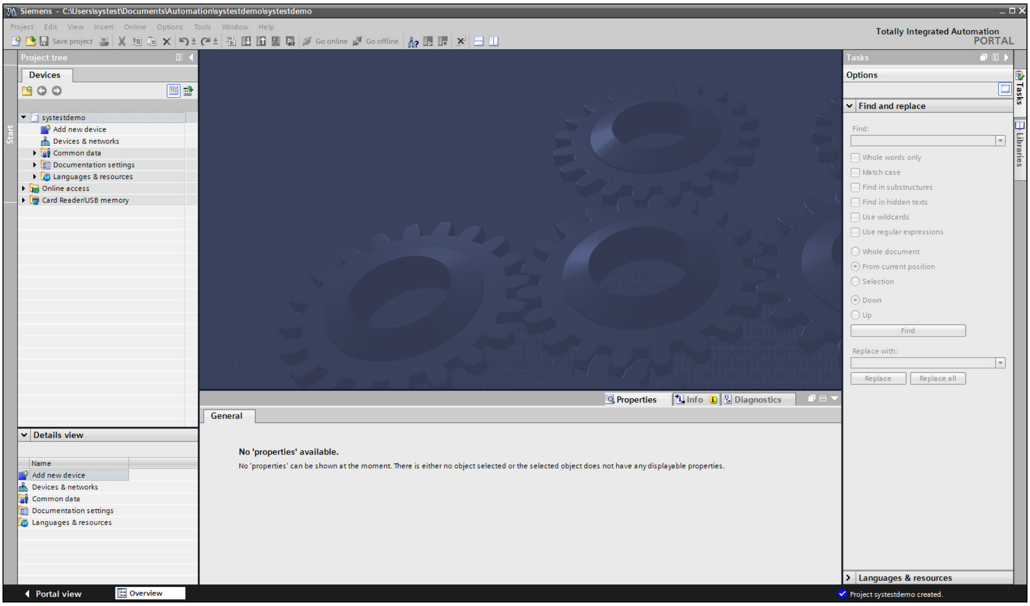
This section was created using Siemens TIA Portal software version 13 and an ET200SP CPU catalog number 6ES7 510-1DJ01-0AB0, HW version 2, FW version 1.8.2.

First launch the TIA Portal from the desktop.

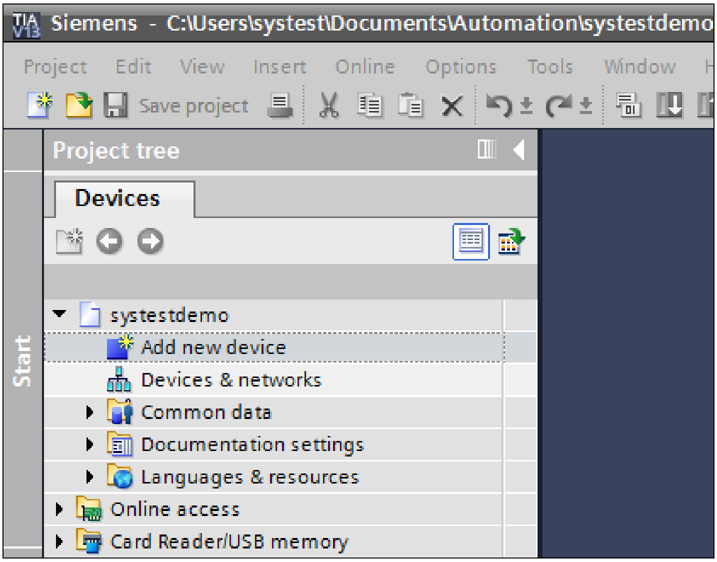

Create a new project from the menu item **Project->New**. The default name is Project1 as seen below. Fill out the required information, then click **Create**.



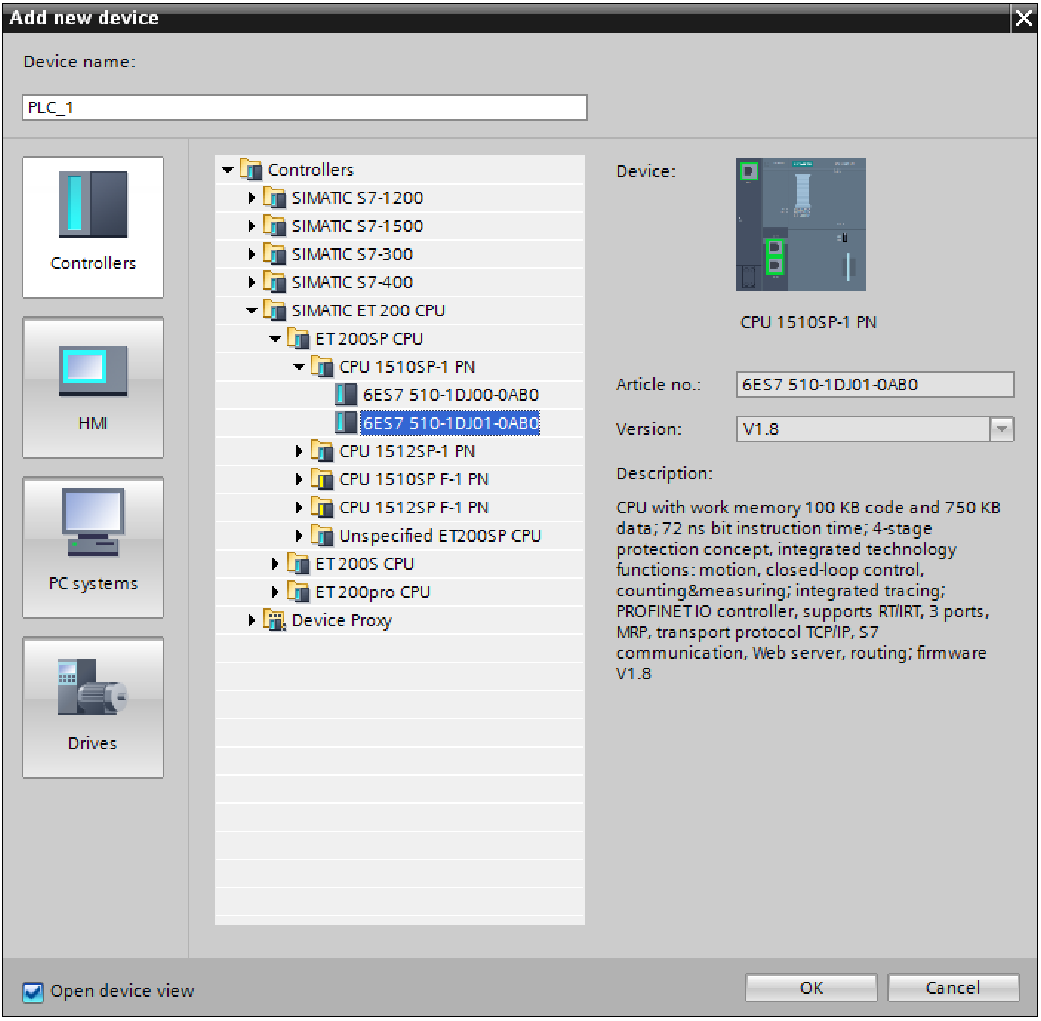
The screen below shows the main screen you will see when a project is opened.



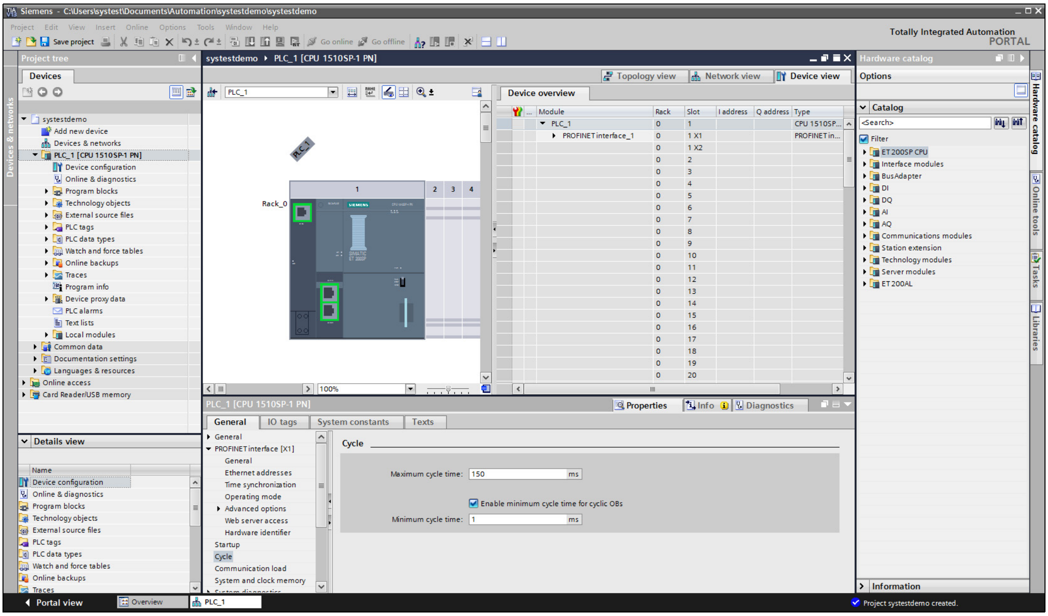
Double-click **Add new device**.



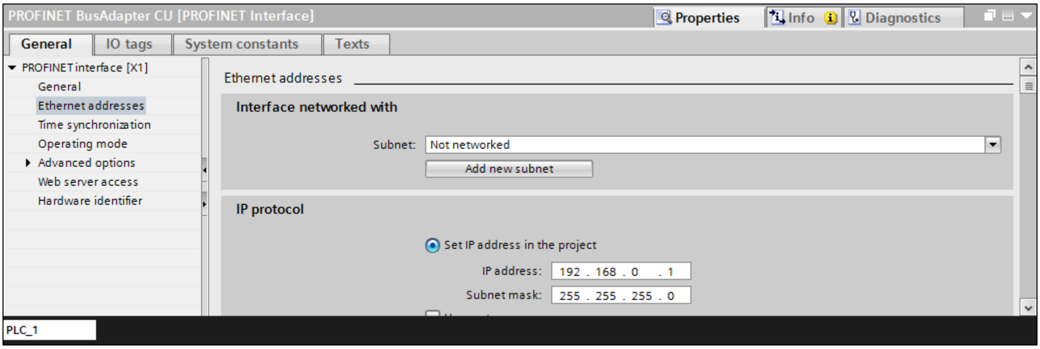
Open the **SIMATIC ET200 CPU** arrow, then open the **ET200SP CPU** arrow, then open the **CPU 1510SP-1 PN**, then click on **6ES7 510-1DJ01-0AB0**. You will see the screen below. Click **OK**.



You will see the following screen.

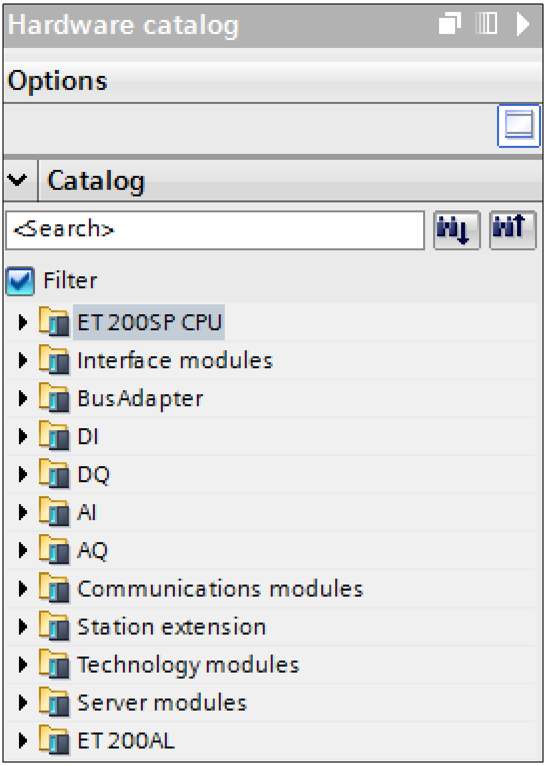



Click the lower green box on the PLC picture. The following will be displayed in the lower pane.




Click **Add new subnet**. The Subnet field will show PN/IE_1. Now go to the IP address field, and type the IP address you wish to use for this device. Click **Save Project** often to save your work.

To the right of the screen, you will see the following:

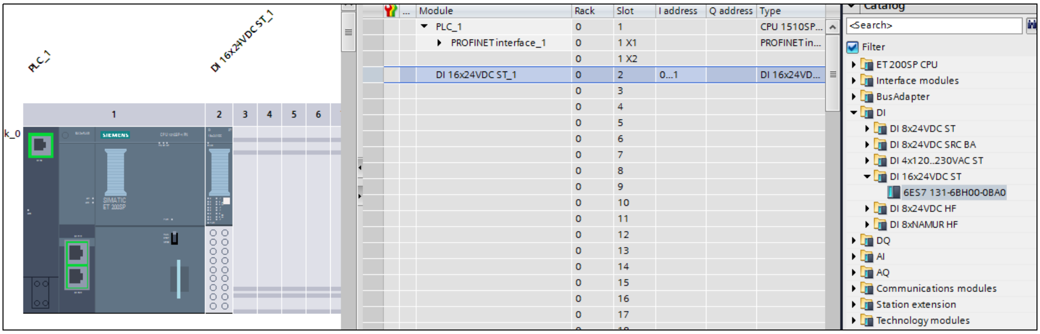



The modules in our test device are as follows:

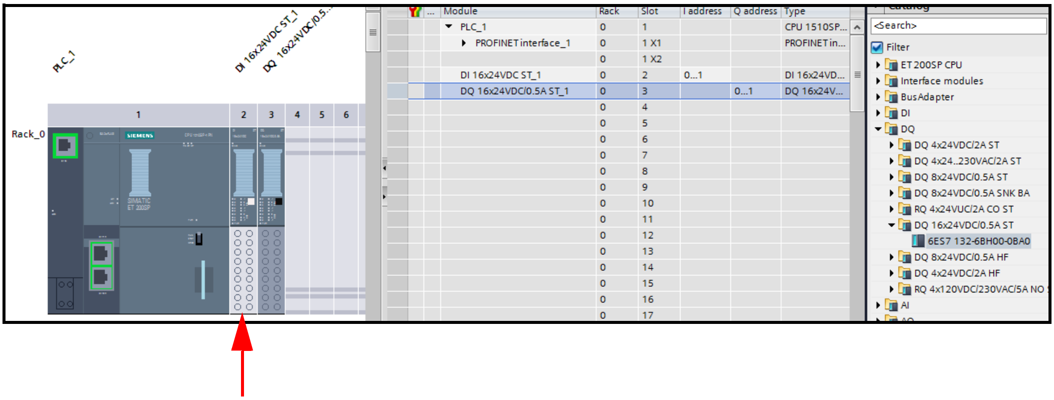
1. DI 16x24VDC ST order number 6ES7 131-6BH00-0BA0
2. DQ 16x24VDC ST order number 6ES7 132-6BH00-0BA0

To add these modules, first click the arrow next to **DI**, keep clicking on arrows until you find the order number for device 1. Click and hold down the left mouse key. Drag it to slot 2 just to the left.

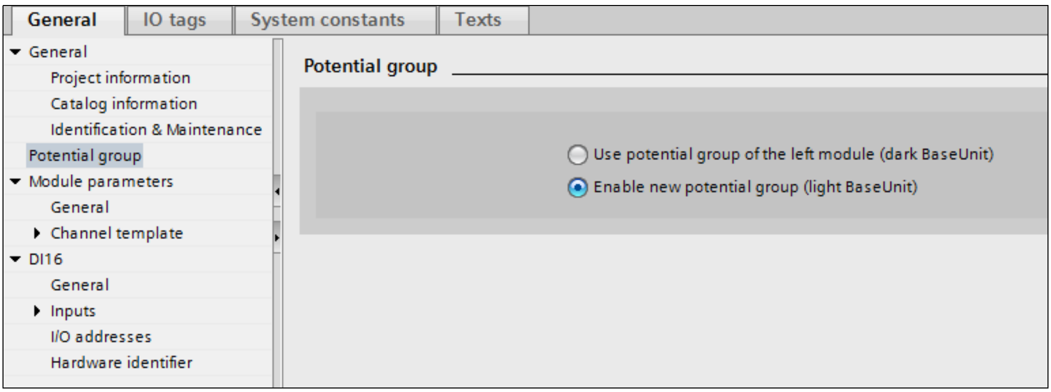
Your display should look like this:




Do the same thing for device 2, looking under the **DQ** arrow. Your display should now look like this:

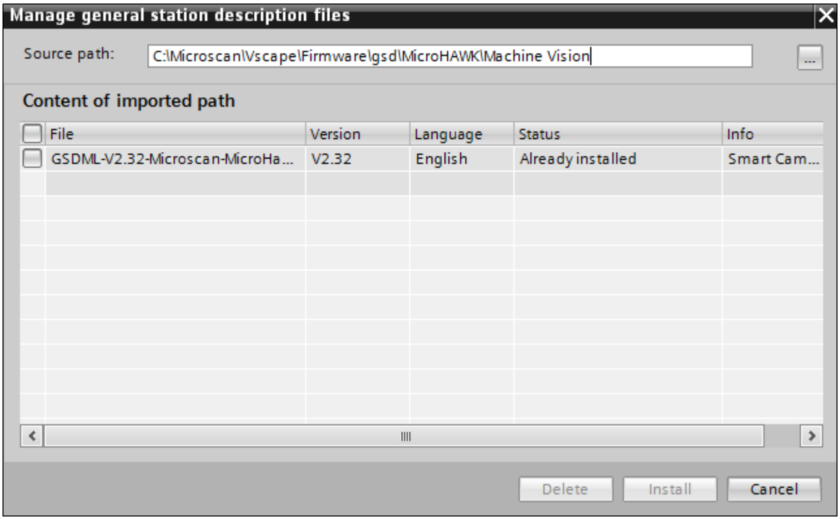



Now click on the PLC picture on slot 2. You will see the following screen:



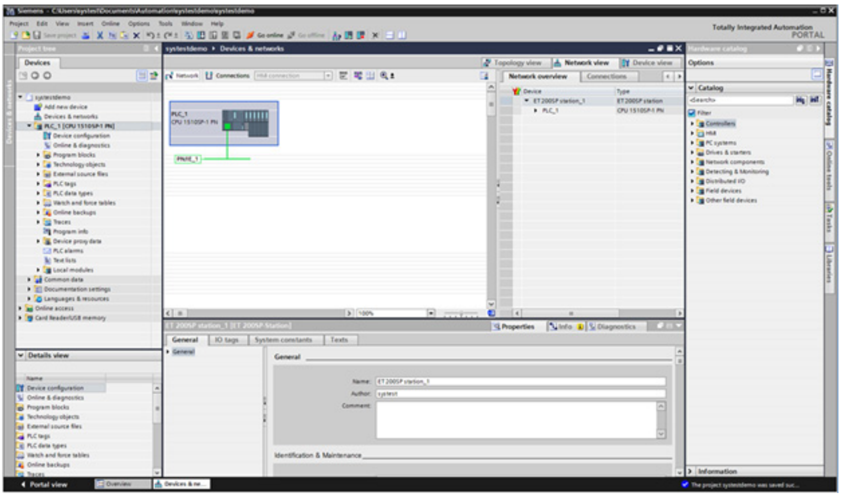
Be sure the **Enable new potential group (light Base Unit)** is selected. Do the same for Slot 3. Save the project.

Now click on **Options->Manage general station description files (GSD)**.

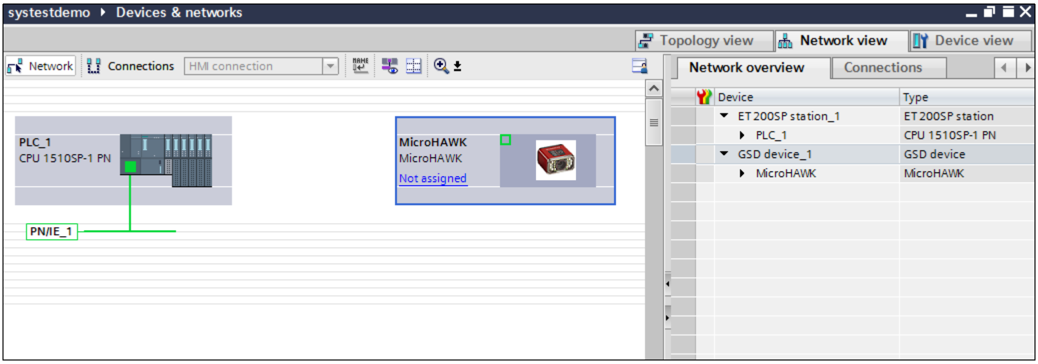



Browse to the following path: **C:\Microscan\Vscape\Firmware\gsd\MicroHAWK**. Select the appropriate GSD file by clicking the box next to the name, and click **Install.**

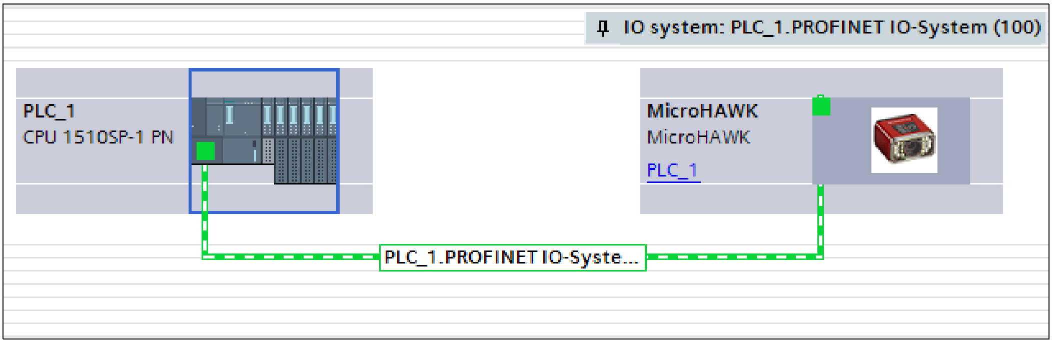
Click the tab that says **Network View**:



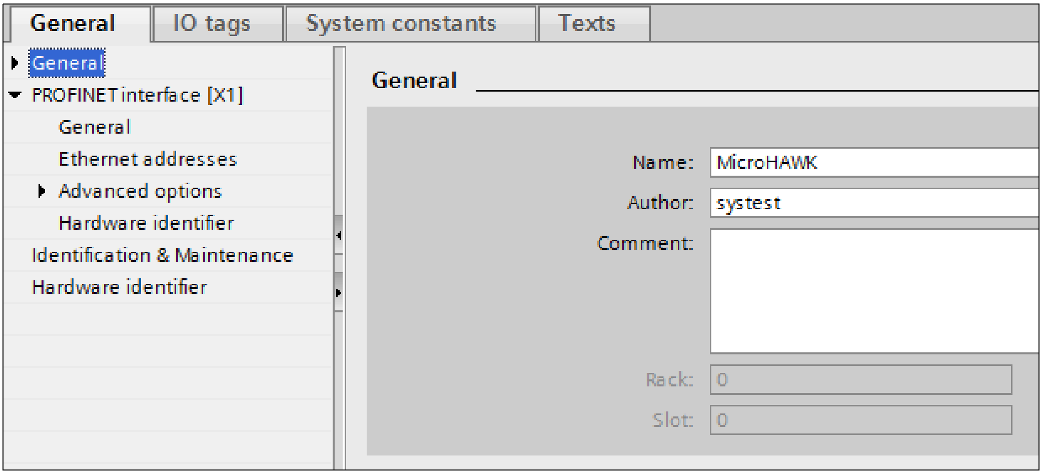
Look at the right panel and click **Hardware Catalog**. Use the arrows to open up the following area. **Other field devices->PROFINET IO ->General->MICROSCAN->Smart Camera->MicroHAWK MV-40**. Left-click on MicroHAWK MV-40 and hold it. Drag it to the left into the Devices and networks panel. Your screen will look like this:




On the MicroHAWK picture, click the **Not assigned** link, then click on **PLC_1.PROFINET IO-System**. This has just connected the MicroHAWK to the PROFINET network.



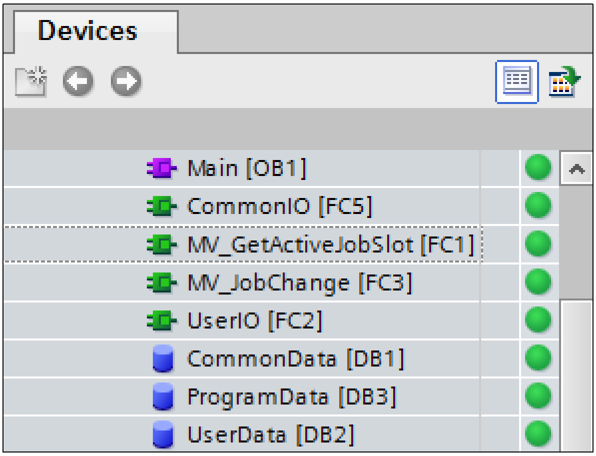
Double-click the picture of the MicroHAWK. In the bottom pane you will see the following:




Enter the name of the device in the Name: field. Then click Ethernet addresses on the left, and type in the IP address of the camera. This will allow the PLC to set the camera IP.

## Importing an Example Program

From TIA Portal, open the example program from the menu **Project > Open**. Click the **Browse** button to locate the (**BlobDemo.ap13**) program. The BlobDemo is located in **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\TIAPortal_Demos\BlobDemo**. In the left pane in the **Devices** tab, open the PLC_1[CPU 1510SP-1 PN] arrow. Open the Program blocks arrow. Select all the blocks as shown below.

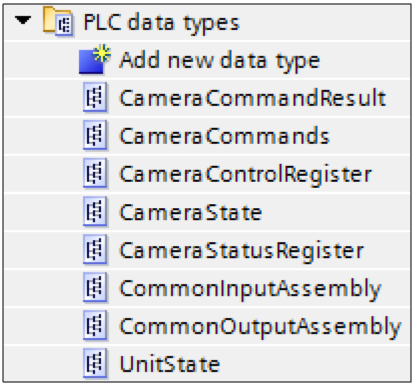


Right-click and copy. Go to your new project and open the same arrows, right-click on Program blocks, then click paste. Be sure to save your work. You may need to delete the Main [OB1] in the new project first.

Now back in the left pane of the BlobDemo project, click on PLC data types, right-click, copy, and paste them into the new project PLC data types.

In the left pane of the BlobDemo project, click on Watch and force tables. Right-click on Watch table_1, then copy. Paste into new project Watch and force tables. Be sure to save the project.

Your data in the new project should look like this:

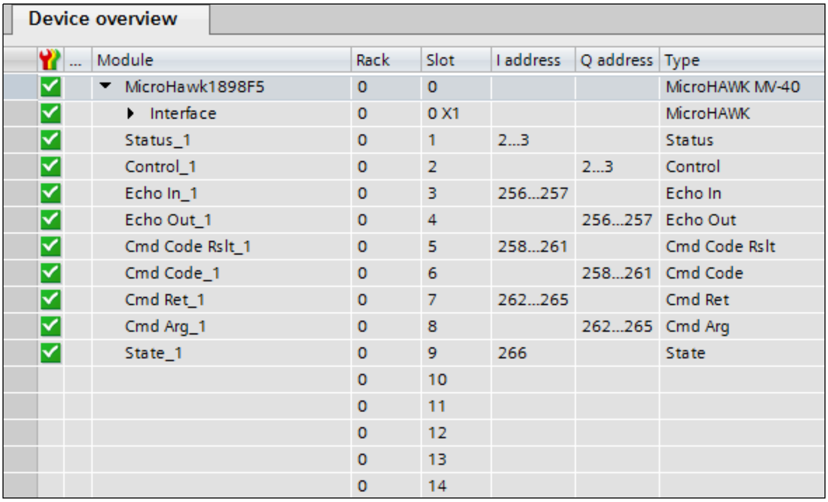


## Checking the Mapping of IO

Depending on the order the PLC can load things, we may need to change the IO addresses in our block to match the Device Overview in the Device view screen.

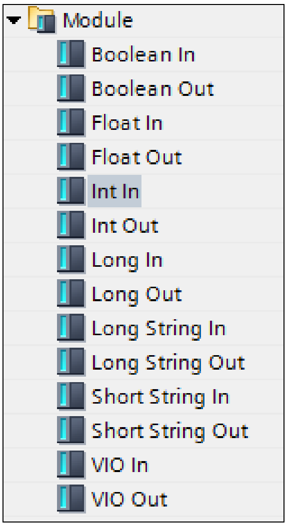
Double-click on the MicroHAWK picture, and the Device overview will be displayed to the right.

The image below is intended as an example only and may not exactly resemble your Device Overview.


Device overview

| Module | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|
| MicroHawk1898F5 | 0 | 0 | | | MicroHAWK MV-40 |
| Interface | 0 | 0 X1 | | | MicroHAWK |
| Status_1 | 0 | 1 | 2...3 | | Status |
| Control_1 | 0 | 2 | | 2...3 | Control |
| Echo In_1 | 0 | 3 | 256...257 | | Echo In |
| Echo Out_1 | 0 | 4 | | 256...257 | Echo Out |
| Cmd Code Rslt_1 | 0 | 5 | 258...261 | | Cmd Code Rslt |
| Cmd Code_1 | 0 | 6 | | 258...261 | Cmd Code |
| Cmd Ret_1 | 0 | 7 | 262...265 | | Cmd Ret |
| Cmd Arg_1 | 0 | 8 | | 262...265 | Cmd Arg |
| State_1 | 0 | 9 | 266 | | State |
| | 0 | 10 | | | |
| | 0 | 11 | | | |
| | 0 | 12 | | | |
| | 0 | 13 | | | |
| | 0 | 14 | | | |

From the Hardware catalog display to the far right, open the **Module** selection. You will see the following:



You must manually drag and drop “Int In” into the Device Overview pane. Click “Int In” and TIA will highlight the slot to drag it to. TIA will only allow you to insert them into their proper slot position.

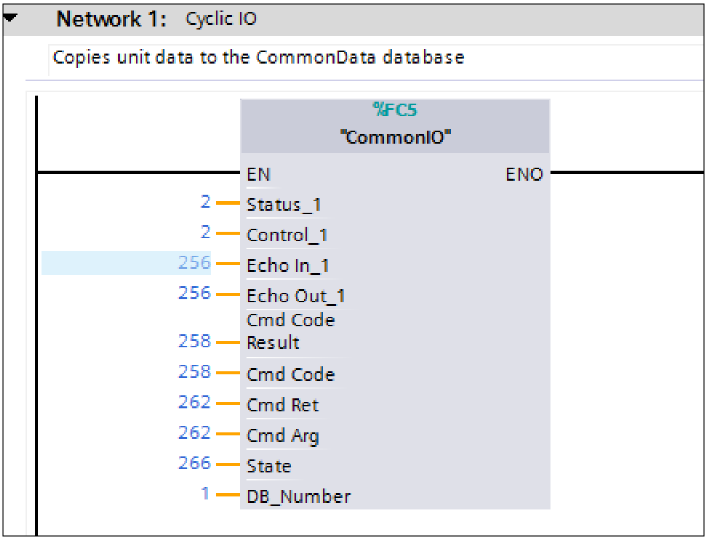
Your device view should now look as follows. The example addresses shown will not be the same for your device.

Device overview

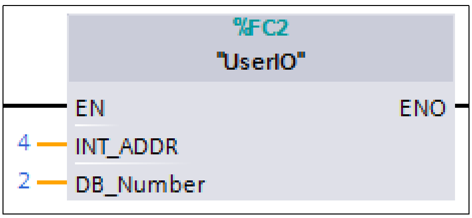
| | ... | Module | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|---|---|
| ✓ | | MicroHawk1898F5 | 0 | 0 | | | MicroHAWK MV-40 |
| ✓ | | Interface | 0 | 0 X1 | | | MicroHAWK |
| ✓ | | Status_1 | 0 | 1 | 2...3 | | Status |
| ✓ | | Control_1 | 0 | 2 | | 2...3 | Control |
| ✓ | | Echo In_1 | 0 | 3 | 256...257 | | Echo In |
| ✓ | | Echo Out_1 | 0 | 4 | | 256...257 | Echo Out |
| ✓ | | Cmd Code Rslt_1 | 0 | 5 | 258...261 | | Cmd Code Rslt |
| ✓ | | Cmd Code_1 | 0 | 6 | | 258...261 | Cmd Code |
| ✓ | | Cmd Ret_1 | 0 | 7 | 262...265 | | Cmd Ret |
| ✓ | | Cmd Arg_1 | 0 | 8 | | 262...265 | Cmd Arg |
| ✓ | | State_1 | 0 | 9 | 266 | | State |
| | | | 0 | 10 | | | |
| | | | 0 | 11 | | | |
| | | | 0 | 12 | | | |
| | | | 0 | 13 | | | |
| | | | 0 | 14 | | | |
| ✓ | | Int In_1 | 0 | 15 | 4...23 | | Int In |
| | | | 0 | 16 | | | |

You need to modify the PLC program blocks to agree with this display. On the left under the Devices pane, open the Program blocks arrow, and double-click on **OB1 [OB1]**. This will open a screen in the middle pane. Scroll this middle pane down until you see Network 1.

Change the numbers here to match with the Device overview display, as shown in the example below. Leave the DB_Number as is.

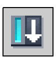



Repeat this for Network 4, using the Int In_1 address from the Device Overview and put it in the INT_ADDR below. Leave the DB_Number as is.

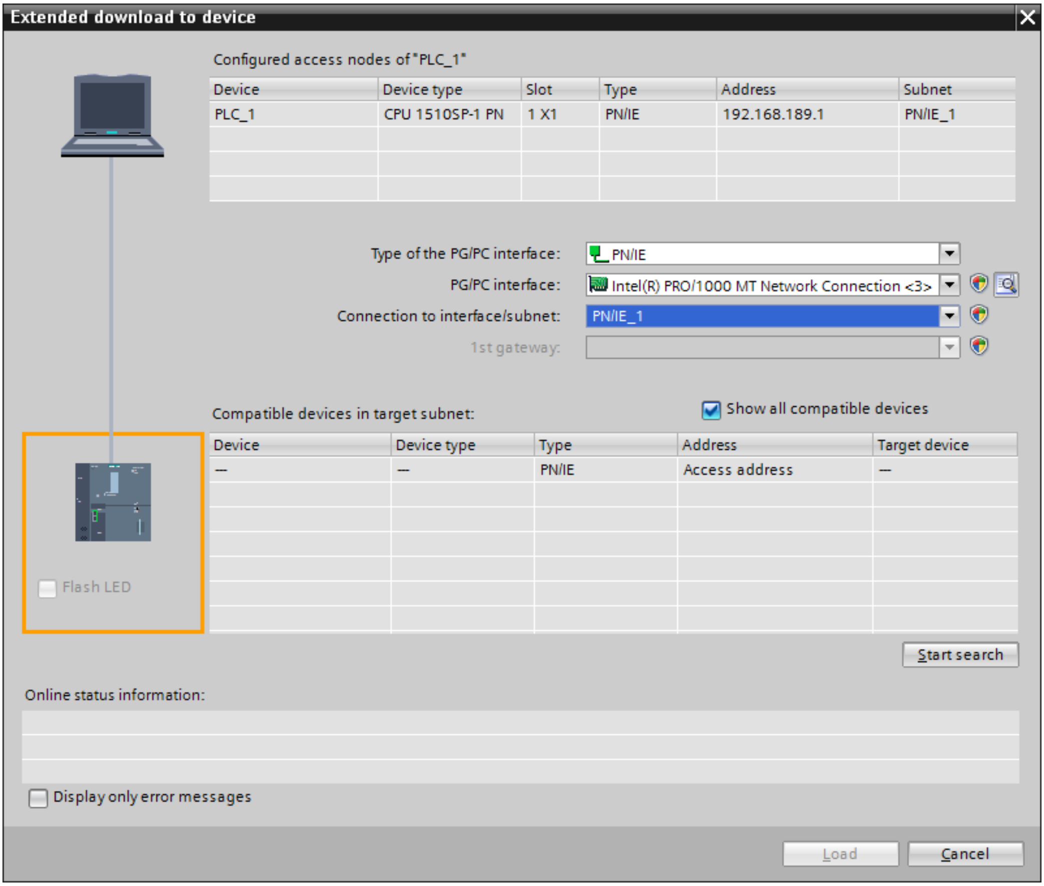



Be sure to save the project. Click the **PLC_1 {CPU 1510SP-1 PN]** top level, then click the compile icon.

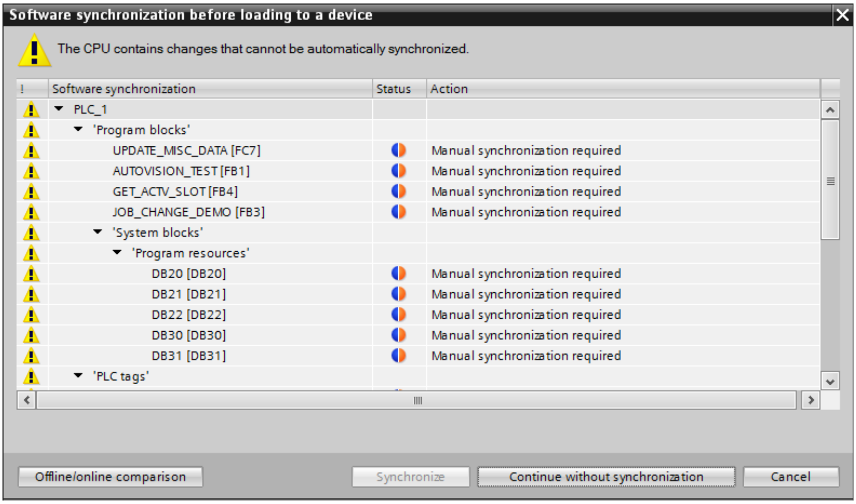
There should be NO compile errors. If there are no errors, click the download icon.



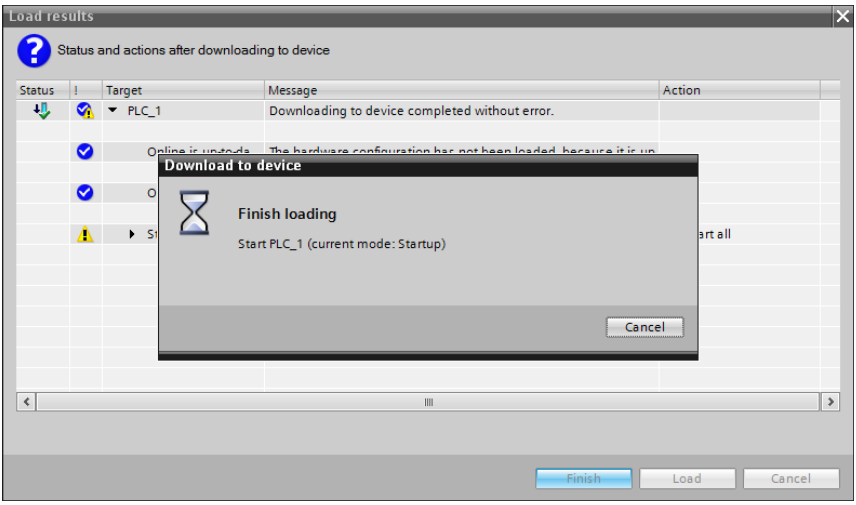
You will see the following screen:



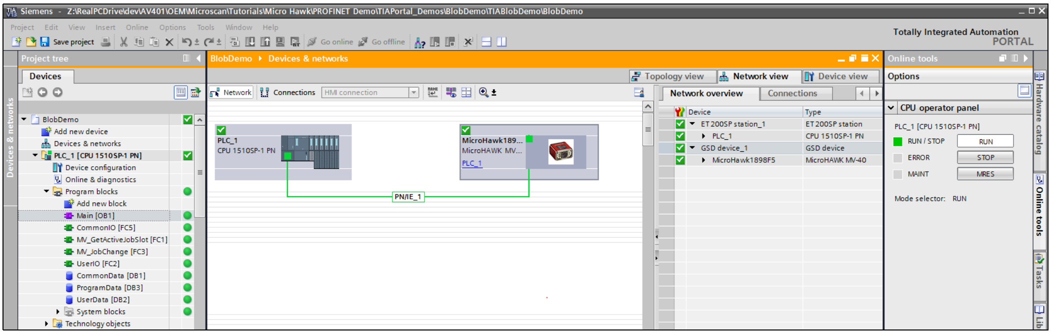
Select your PC network interface, and change the Connection to interface/subnet to **PN/IE_1**. Click **Start Search**. PLC_1 should be displayed with the address you gave it. Click **Load**. If you see a screen similar to below, do a memory reset (MRES) on the PLC, and try to load again. If this continues to be an issue, click **Continue without synchronization** and continue.




When downloading is working correctly you will see the following:



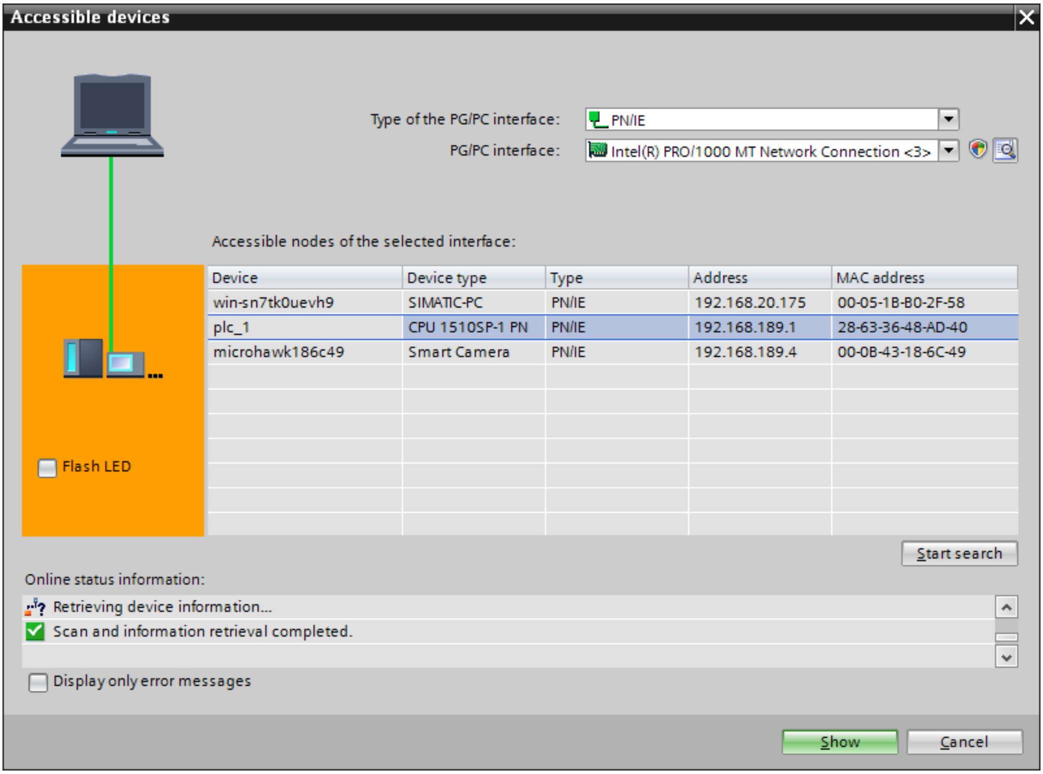
All LEDS on the PLC should be **GREEN**. Click **Go online**. You should see the following:



# Troubleshooting

If all LEDs are not green, go offline and click the Accessible devices icon.

Click **Start search**. The PLC and the MicroHAWK should be displayed here.




If the name of the MicroHAWK is not correct, you will not connect. The default name is microhawk with the last 3 of the MAC address. Ex: microhawk186c49. Go back to the general settings of the MicroHAWK in the device view, and change the name if required. Re-compile, and re-load. Try running again. This is the most common reason for not connecting.

## Running the Demo

In the left pane click the Watch and force tables arrow. Double-click Watch table_1. The watch table will open in the center pane. Click on the monitor variables icon.

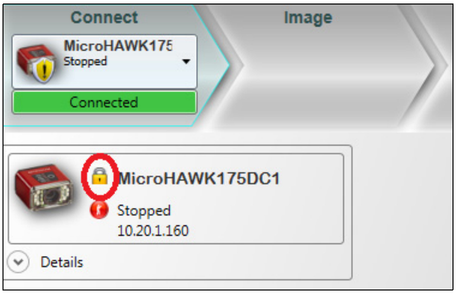
| | i | Name | Address | Display format | Monitor v... | Modify value | |
|---|---|---|---|---|---|---|---|
| 1 | | "ProgramData".ChangeJob | | Bool | FALSE | | |
| 2 | | "ProgramData".New_JobSlot | | DEC+/- | 2 | | |
| 3 | | "ProgramData".Get_CurrentJobSlot | | Bool | FALSE | | |
| 4 | | "ProgramData".Current_JobSlot | | DEC+/- | 1 | | |
| 5 | | "ProgramData".Trigger | | Bool | FALSE | | |
| 6 | | "ProgramData".TotalTriggerCount | | DEC+/- | 1 | | |
| 7 | | "ProgramData".Pass | | Bool | FALSE | | |
| 8 | | "ProgramData".TotalPassedCount | | DEC+/- | 0 | | |
| 9 | | "ProgramData".Fail | | Bool | TRUE | | |
| 10 | | "ProgramData".TotalFailedCount | | DEC+/- | 1 | | |
| 11 | | "ProgramData".TotalInspCount | | DEC+/- | 1 | | |
| 12 | | "ProgramData".ResetCounters | | Bool | FALSE | | |
| 13 | | "UserData".NumberOfParts | %DB2.DBW0 | Hex | 16#0000 | | |
| 14 | | "ProgramData".ResetError | | Bool | FALSE | | |
| 15 | | "ProgramData".CommandResult.Fail | | Bool | FALSE | | |
| 16 | | "ProgramData".CommandResult."No Job In Slot" | | Bool | FALSE | | |
| 17 | | "ProgramData".CommandResult.UnknownCmd | | Bool | FALSE | | |
| 18 | | "ProgramData".CommandResult.Success | | Bool | FALSE | | |

The following operations can be tried here:

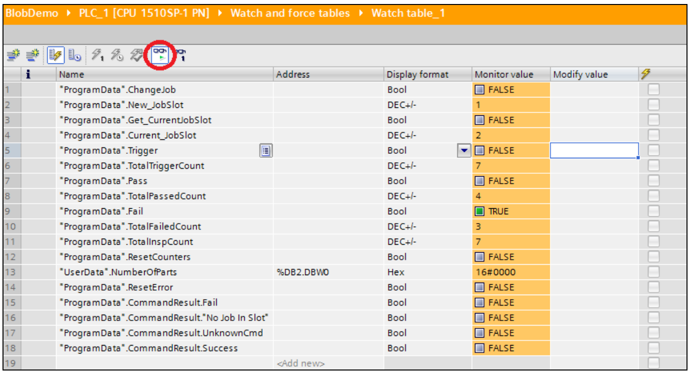
| | |
|---|---|
| Change Job | Switch jobs from one slot to another |
| Get Current Job Slot | Displays the slot of the currently running job |
| Trigger an Inspection | Causes the camera to cycle through its inspection. |
| Reset Counters | Resets the TotalTriggerCount, TotalPassedCount, TotalFailedCount, and TotalInspCount to 0, and also sends a command to the camera to reset its internal counters. |
| Reset an Error condition | If an error occurs during run time, this will reset it, so demo may be continued. |

### Change Jobs + Get Current Job Slot

Be sure to store a job in a slot in the camera memory as described earlier. To execute this test properly, load another job into **Slot 2** of the camera. For example, the **Circle_LocatorDemo** job. You must release control of the device by clicking on the **Connect** tab and change the lock icon to **closed** as in the following image, then click the **Run** tab again for job change to work properly.

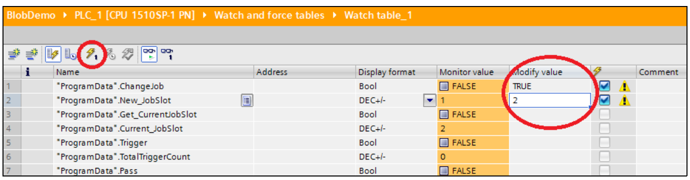



Go to **Watch table_1**, and click the button with the glasses and "play" symbol. This begins the monitoring of data.



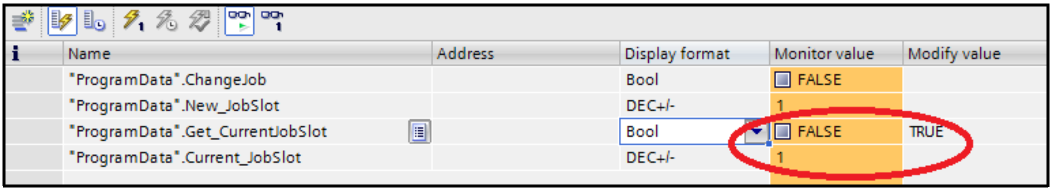

| i | Name | Address | Display format | Monitor value | Modify value |
|---|---|---|---|---|---|
| 1 | "ProgramData".ChangeJob | | Bool | FALSE | |
| 2 | "ProgramData".New_JobSlot | | DEC+/- | 1 | |
| 3 | "ProgramData".Get_CurrentJobSlot | | Bool | FALSE | |
| 4 | "ProgramData".Current_JobSlot | | DEC+/- | 2 | |
| 5 | "ProgramData".Trigger | | Bool | FALSE | |
| 6 | "ProgramData".TotalTriggerCount | | DEC+/- | 7 | |
| 7 | "ProgramData".Pass | | Bool | FALSE | |
| 8 | "ProgramData".TotalPassedCount | | DEC+/- | 4 | |
| 9 | "ProgramData".Fail | | Bool | TRUE | |
| 10 | "ProgramData".TotalFailedCount | | DEC+/- | 3 | |
| 11 | "ProgramData".TotalInspCount | | DEC+/- | 7 | |
| 12 | "ProgramData".ResetCounters | | Bool | FALSE | |
| 13 | "UserData".NumberOfParts | %DB2.DBW0 | Hex | 16#0000 | |
| 14 | "ProgramData".ResetError | | Bool | FALSE | |
| 15 | "ProgramData".CommandResult.Fail | | Bool | FALSE | |
| 16 | "ProgramData".CommandResult."No Job In Slot" | | Bool | FALSE | |
| 17 | "ProgramData".CommandResult.UnknownCmd | | Bool | FALSE | |
| 18 | "ProgramData".CommandResult.Success | | Bool | FALSE | |
| 19 | | <Add new> | | | |

Using the Watch table_1 elements **ChangeJob** and **New_JobSlot**, click on the **Modify Value** column next to **New_JobSlot**, type 2, type **Enter**. Click on the Modify value to the right of ChangeJob and enter **TRUE**. Click on the lightning bolt with a **1** at the top. This will set the job slot to 2. Then click the lightning bolt button again.




| i | Name | Address | Display format | Monitor value | Modify value | Comment |
|---|---|---|---|---|---|---|
| 1 | "ProgramData".ChangeJob | | Bool | FALSE | TRUE | |
| 2 | "ProgramData".New_JobSlot | | DEC+/- | 1 | 2 | |
| 3 | "ProgramData".Get_CurrentJobSlot | | Bool | FALSE | | |
| 4 | "ProgramData".Current_JobSlot | | DEC+/- | 2 | | |
| 5 | "ProgramData".Trigger | | Bool | FALSE | | |
| 6 | "ProgramData".TotalTriggerCount | | DEC+/- | 0 | | |
| 7 | "ProgramData".Pass | | Bool | FALSE | | |

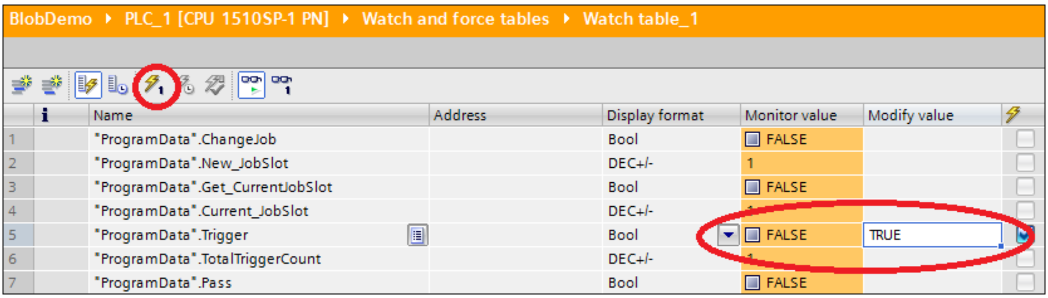
The current job will be changed to the job in slot 2. To verify that this clears all changes made to the Modify value column, click on Modify value to the right of **Get_CurrentJobSlot**. Enter **TRUE**, and click on the lightning bolt button again.




You will see the current job listed to the right of **Current_JobSlot**. You can repeat this procedure changing the slot back to 1 and verify that the current job slot is now **1**.

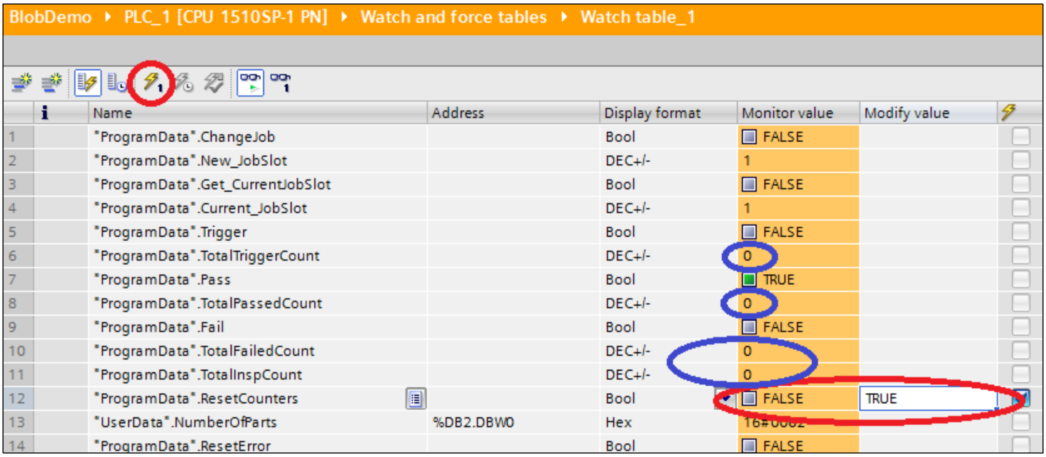
### Trigger an Inspection and Reset Counters

Clear any changes made to the Modify value column. If you want to watch the inspection, be sure AutoVISION is running and connected to the device. Be sure the job is running. If you choose not to run AutoVISION, you can look at the LEDs on the device to confirm your triggers. Change the value to the right of **Trigger** to be **TRUE**, and click the lightning bolt button again.




You will see the **NumberOfParts** change to either **2** or **5**. The pass and fail bools will indicate whether the inspection has passed or failed. Also, the appropriate counters will have incremented. Each click of the lightning bolt button will cause an inspection, which will bounce from one image to the next. One will pass and the next will fail.

To reset the counters, clear all settings in the Modify value column, and change the one next to **ResetCounters** to **TRUE**, and click the lightning bolt button again. You will see all the counters in blue reset to **0**.



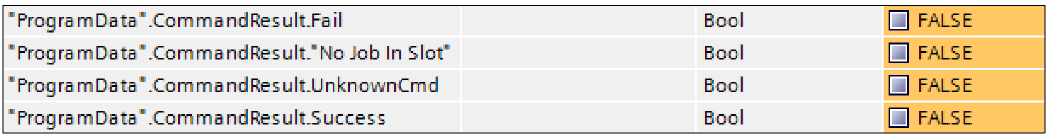
BlobDemo ▸ PLC_1 [CPU 1510SP-1 PN] ▸ Watch and force tables ▸ Watch table_1

| | i | Name | Address | Display format | Monitor value | Modify value |
|---|---|---|---|---|---|---|
| 1 | | "ProgramData".ChangeJob | | Bool | FALSE | |
| 2 | | "ProgramData".New_JobSlot | | DEC+/- | 1 | |
| 3 | | "ProgramData".Get_CurrentJobSlot | | Bool | FALSE | |
| 4 | | "ProgramData".Current_JobSlot | | DEC+/- | 1 | |
| 5 | | "ProgramData".Trigger | | Bool | FALSE | |
| 6 | | "ProgramData".TotalTriggerCount | | DEC+/- | 0 | |
| 7 | | "ProgramData".Pass | | Bool | TRUE | |
| 8 | | "ProgramData".TotalPassedCount | | DEC+/- | 0 | |
| 9 | | "ProgramData".Fail | | Bool | FALSE | |
| 10 | | "ProgramData".TotalFailedCount | | DEC+/- | 0 | |
| 11 | | "ProgramData".TotalInspCount | | DEC+/- | 0 | |
| 12 | | "ProgramData".ResetCounters | | Bool | FALSE | TRUE |
| 13 | | "UserData".NumberOfParts | %DB2.DBW0 | Hex | 16#0002 | |
| 14 | | "ProgramData".ResetError | | Bool | FALSE | |

If you clear this again, set the Trigger to TRUE, and click the lightning bolt button again, you will notice that the counts start at **1** on both the PLC counters and AutoVISION counters.

### Reset Error

If at any time there is an error indicated by a TRUE in any of the values below, you can clear the error by entering a TRUE to the right of **ResetError** and clicking the lightning button. Reset it to **FALSE** and click the button again. Any errors in the values below should be cleared.



| | | | |
|---|---|---|---|
| "ProgramData".CommandResult.Fail | | Bool | FALSE |
| "ProgramData".CommandResult."No Job In Slot" | | Bool | FALSE |
| "ProgramData".CommandResult.UnknownCmd | | Bool | FALSE |
| "ProgramData".CommandResult.Success | | Bool | FALSE |

CHAPTER 10

# PROFINET I/O Circle Locate Demo Using TIA Portal V13 and MicroHAWK

This section describes how to use Omron Microscan demo PLC code with a vision job and camera target. The PROFINET I/O demo files can be found where AutoVISION is installed, in the folder: **C:\Microscan\Tutorials and Samples\MicroHAWK\PROFINET demo\ TIAPortal_Demos\ Circle_LocateDemo**. Open **Circle_LocatorDemo.avp** with AutoVISION and download it to the camera.

**Notes:**

- The camera communications protocol must be enabled for PROFINET I/O before it can be used in this environment. Refer to **Chapter 1, Enabling Industrial Protocols**, for information about enabling and switching communications protocols.

- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate PROFINET I/O communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.
**Important:** PROFINET I/O allows the IP address and subnet mask of the camera to be assigned automatically by the PLC. In order to allow this, the MicroHAWK may not have an assigned IP address on reboot until the PLC is set to Run mode. During this time, the MicroHAWK will not be visible on the network for AutoVISION or Visionscape FrontRunner.

## Overview

In this demonstration, you will learn how to load a saved job into the camera, establish connectivity via PROFINET I/O to a Siemens ET200SP CPU PLC, and run some example programs that interface with the camera. While evaluating PROFINET I/O capabilities, you will:

- **Begin with AutoVISION.**

Open a sample demo vision job in AutoVISION and use the Try Out feature to learn what to expect from the camera before it is connected to the PLC.

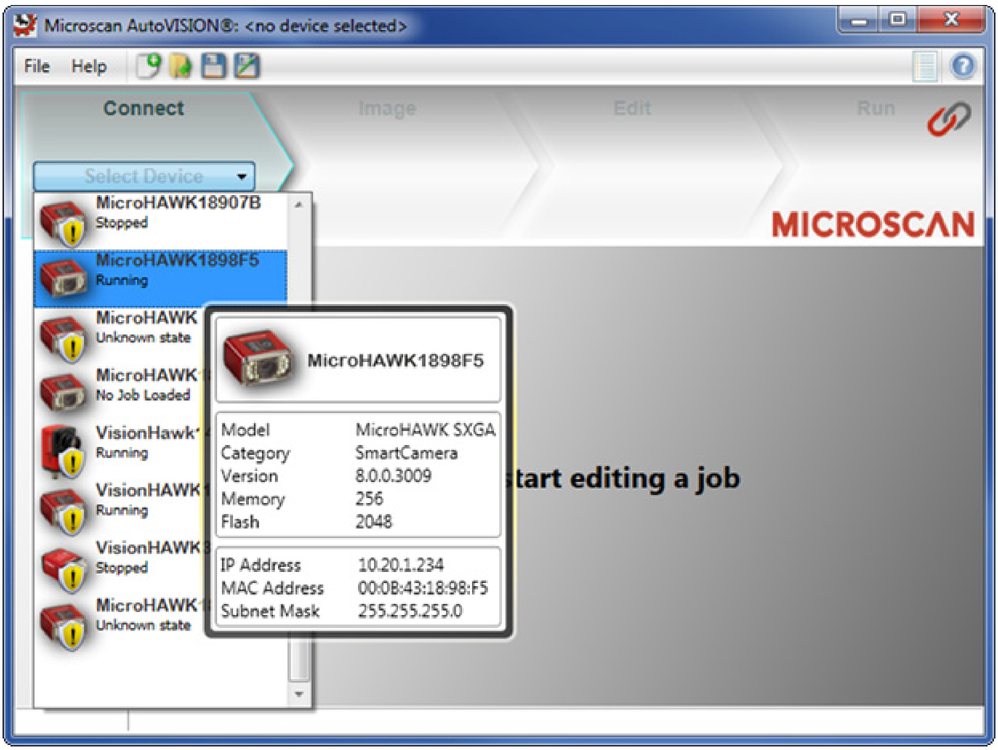
- **Prepare the PLC.**

Integrate the camera into the PLC environment with TIA Portal software and the GSD file.

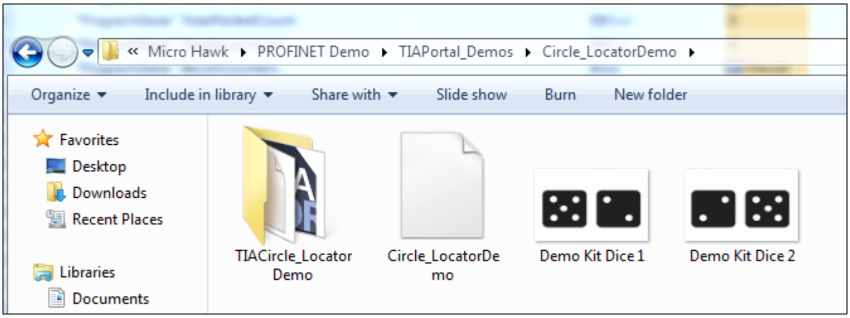
- **Run the camera.**

Trigger the camera while it is online with the PLC and observe changes in the Inspection status as the PLC reconfigures the vision job's parameters. Prior to starting AutoVISION, make sure the camera is either connected to the PLC or both PLC and camera are on the same physical network. Ensure that the PC, PLC, and camera have the same network class and corresponding subnet addresses.

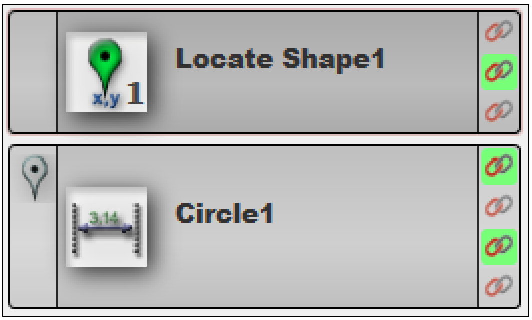
Open AutoVISION and select the camera.



From the Image view, click the **Load a Job** button. Then navigate to **C:\Microscan\Vscape\Tutorials And Samples\MicroHAWK\PROFINET demo\TIAPortal_Demos\Circle_LocatorDemo**. Select **Circle_LocatorDemo.avp**.




The demo job will include two tools: **Locate Shape** and **Circle**.

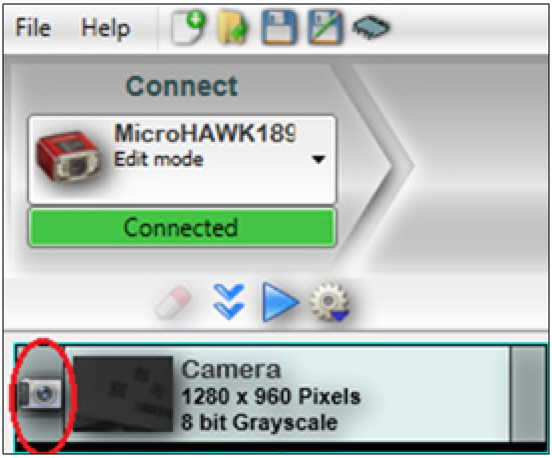



The PROFINET I/O structure is shown here:

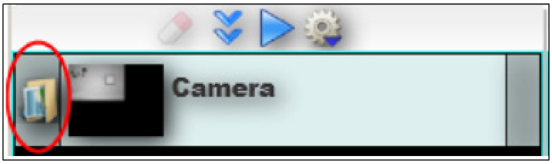
| Tool Result | PLC IN |
|---|---|
| Instance Point1 | "UserData".Instance1Point1(%DB2.DBD2) |
| Instance Point2 | "UserData".Instance1Point2(%DB2.DBD6) |
| Instance Point3 | "UserData".InstancePoint3(%DB2.DBD10) |
| Status | "UserData".Status[0](%DB2.DBX0.0) |
| RadiumDatum | "UserData".RadiusDatum(%DB2.DBD14) |

This data is transferred cyclically between the camera and PLC.

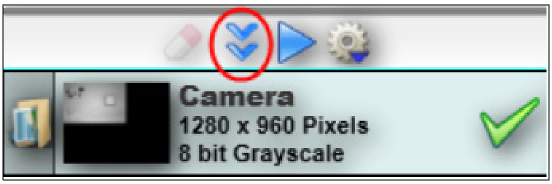
Once the job has loaded, the next step is to link the pre-saved images on the local PC. On the **Camera** button, click the far left icon to select and load an image.



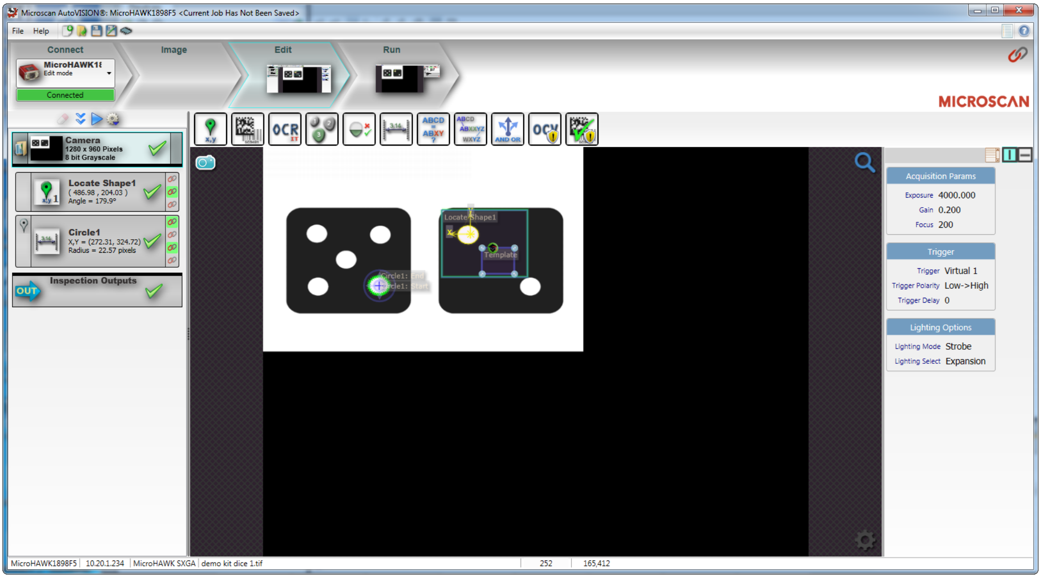

A file browser will open. Then navigate to the same folder where the demo job was loaded PROFINET I/O Demo. If the images are located, the icon will change from a camera to a folder.

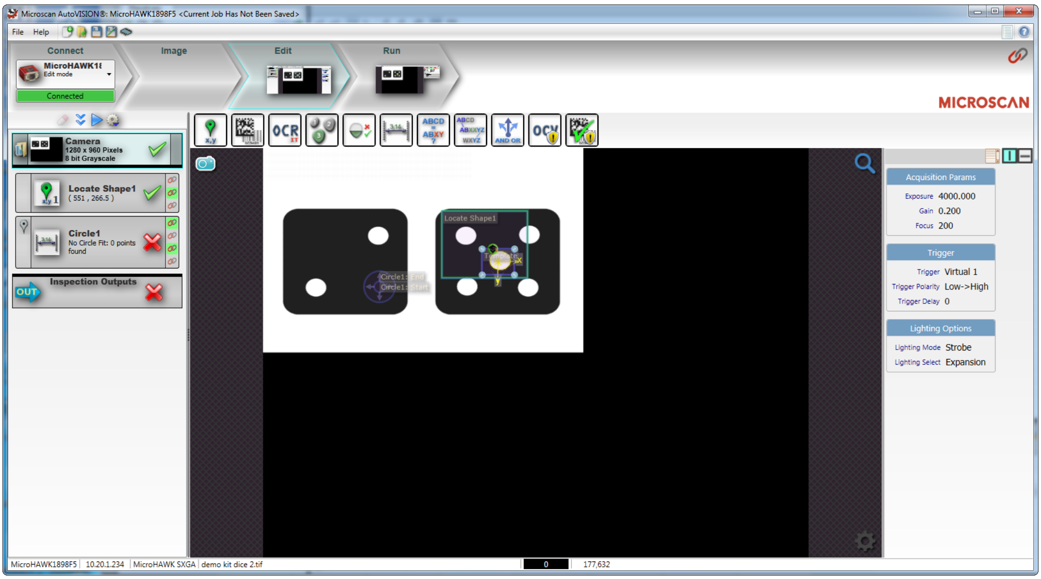



By clicking the **Try Job Once** icon, the application will cycle through the entire job with the loaded image.



There are two images on file. One will pass the inspection and one will fail.



Now click the **Run** button on the top ribbon. This will download the job to the camera. At this point the job is ready to run and can be tested. However, in order to run the job change demo this job needs to be loaded into **Slot 1**. Click back to the **Edit** view from the top ribbon bar.

Click the slot icon and select Slot 1 or **New Slot** if no slots are currently in use.

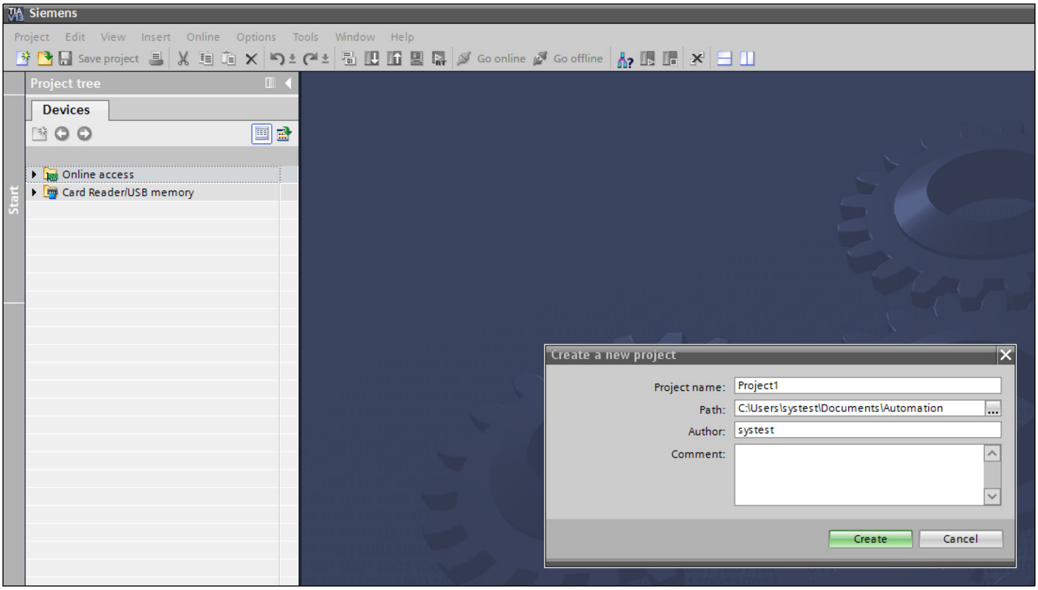

Now the job and images will be saved to the flash memory of the camera.

## TIA Portal V13 Setup

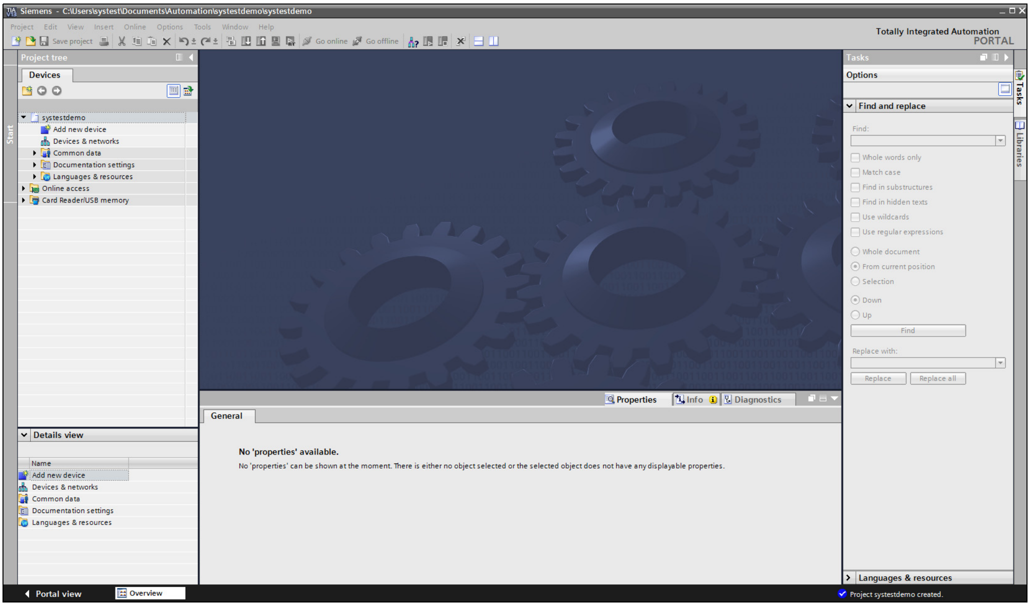
This section was created using Siemens TIA Portal software version 13 and an ET200SP CPU catalog number 6ES7 510-1DJ01-0AB0, HW version 2, FW version 1.8.2. This example shows the MicroHAWK device. However, the steps are similar for any device.

First launch the TIA Portal from the desktop.

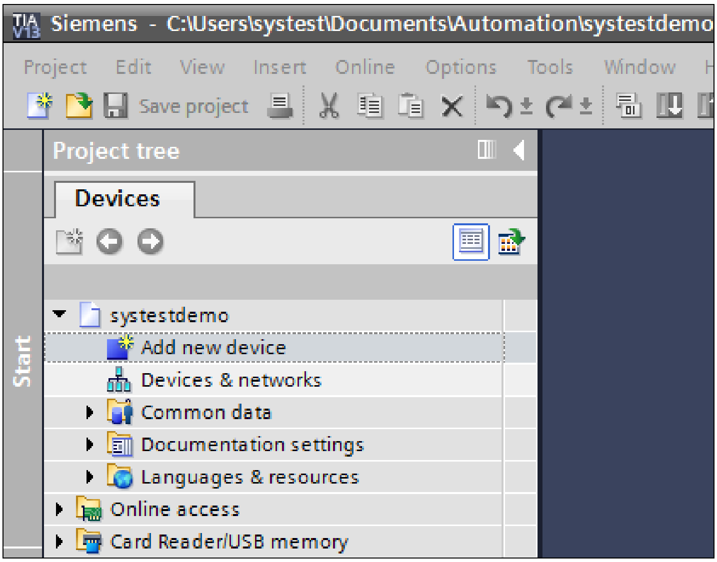

Create a new project from the menu item **Project->New**. The default name is Project1 as seen below. Fill out the required information, then click **Create**.



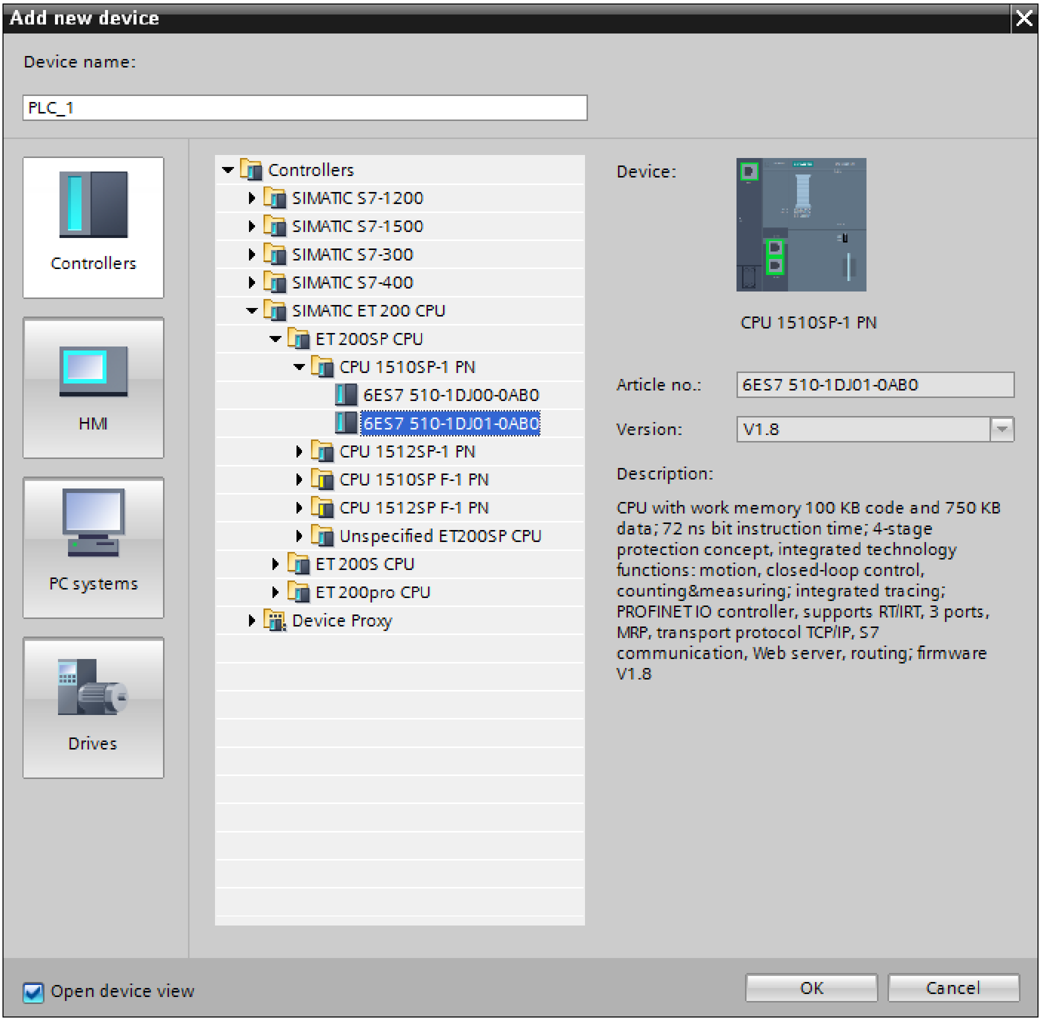
The screen below shows the main screen you will see when a project is opened.



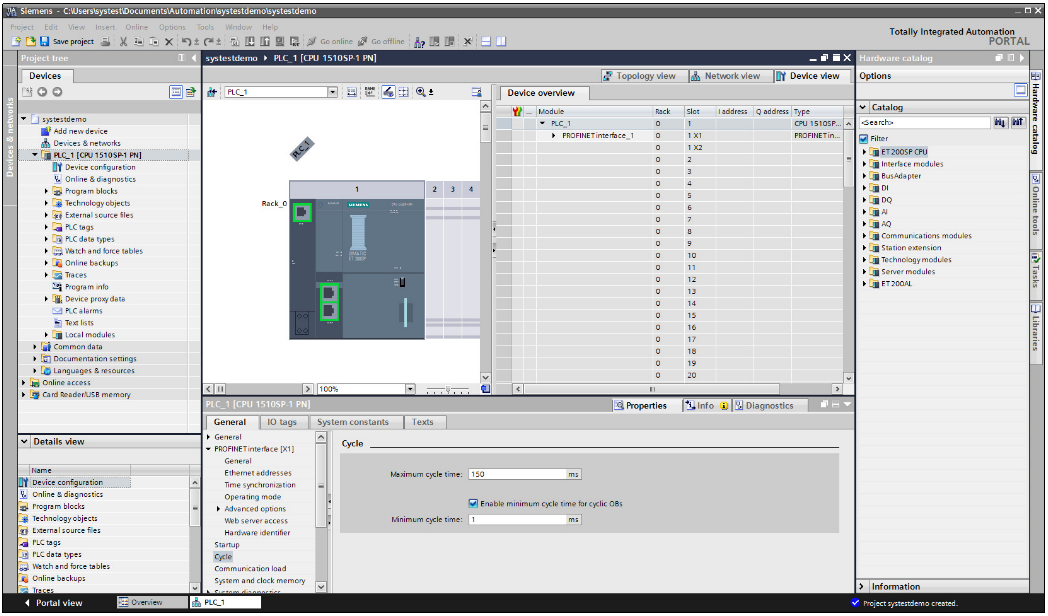

Double-click **Add new device**.



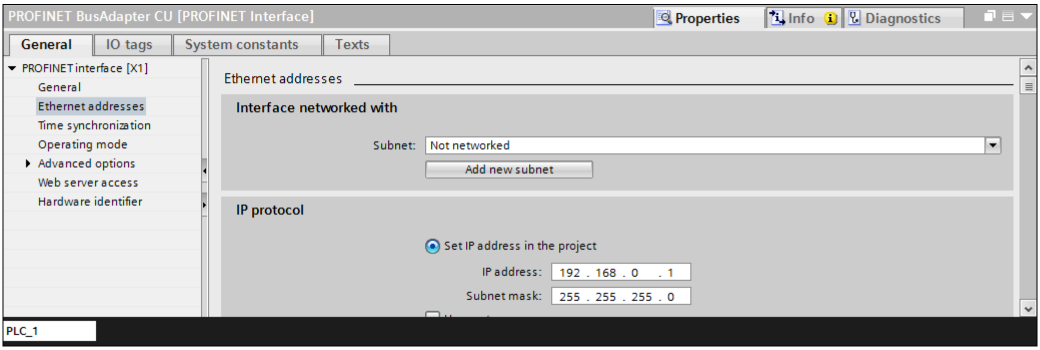
Open the **SIMATIC ET200 CPU** arrow, then open the **ET200SP CPU** arrow, then open the **CPU 1510SP-1 PN**, then click on **6ES7 510-1DJ01-0AB0**. You will see the screen below. Click **OK**.



You will see the following screen.

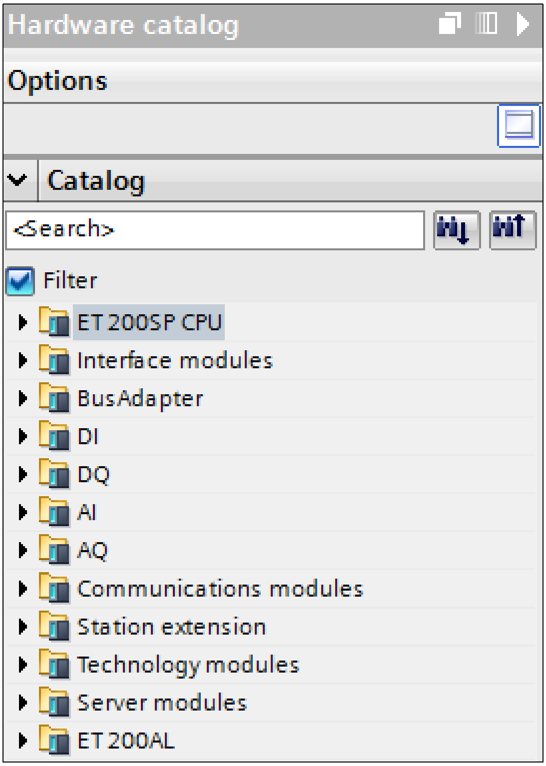



Click the lower green box on the PLC picture. The following will be displayed in the lower pane.




Click **Add new subnet**. The Subnet field will show PN/IE_1. Now go to the IP address field, and type the IP address you wish to use for this device. Click **Save Project** often to save your work.

To the right of the screen, you will see the following:

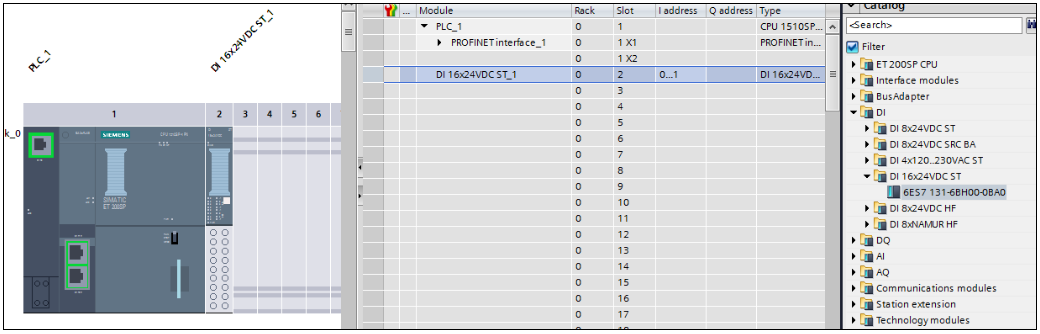



The modules in our test device are as follows:

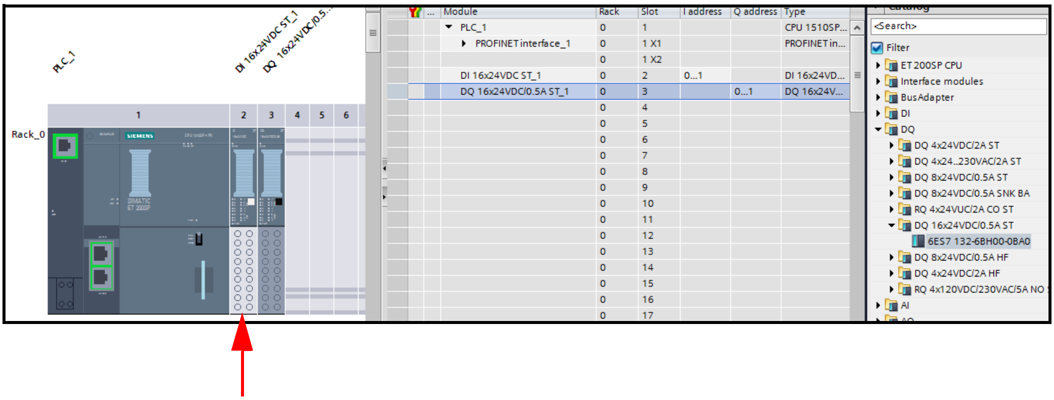
1. DI 16x24VDC ST order number 6ES7 131-6BH00-0BA0
2. DQ 16x24VDC ST order number 6ES7 132-6BH00-0BA0

To add these modules, first click the arrow next to **DI**, keep clicking on arrows until you find the order number for device 1. Click and hold down the left mouse key. Drag it to slot 2 just to the left.

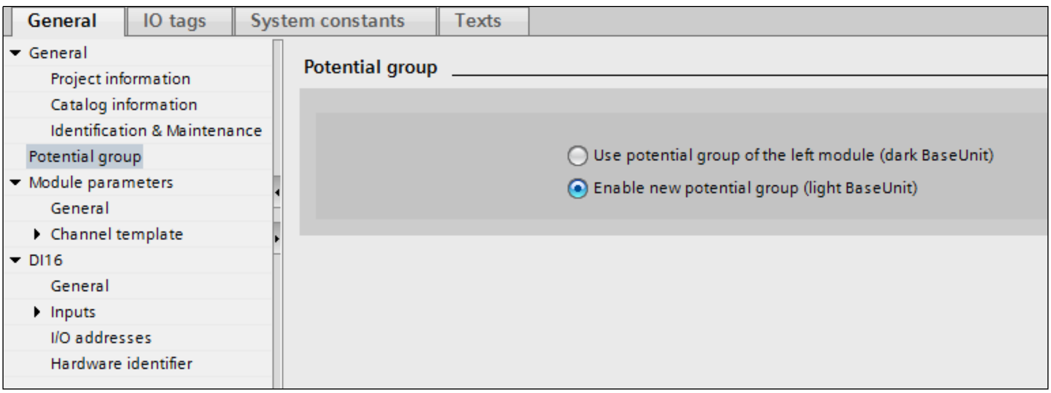
Your display should look like this:




Do the same thing for device 2, looking under the **DQ** arrow. Your display should now look like this:

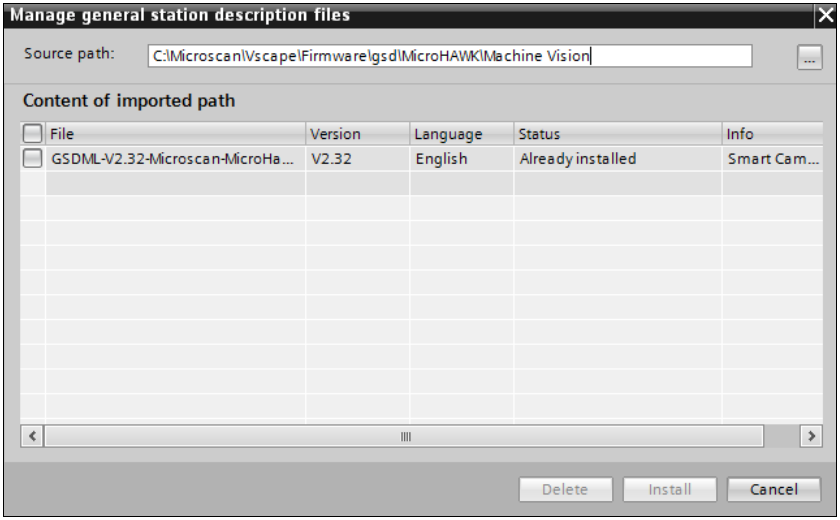



Now click on the PLC picture on slot 2. You will see the following screen:



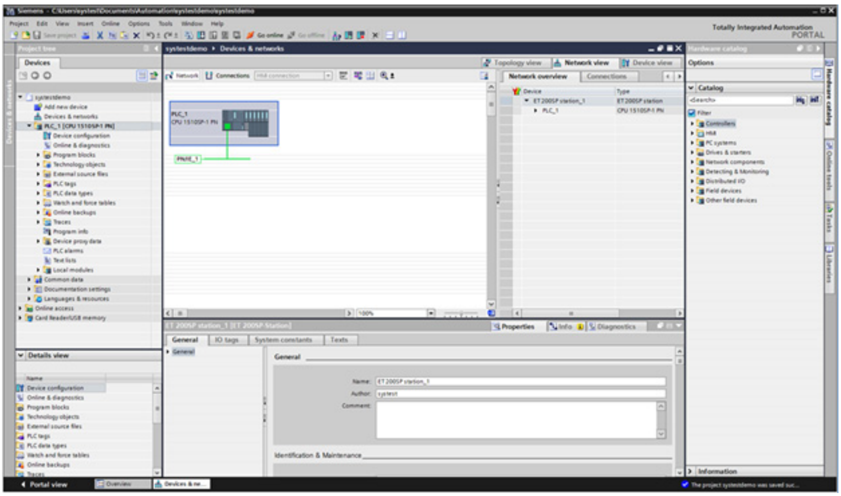
Be sure the **Enable new potential group (light Base Unit)** is selected. Do the same for Slot 3. Save the project.

Now click on **Options->Manage general station description files (GSD)**.

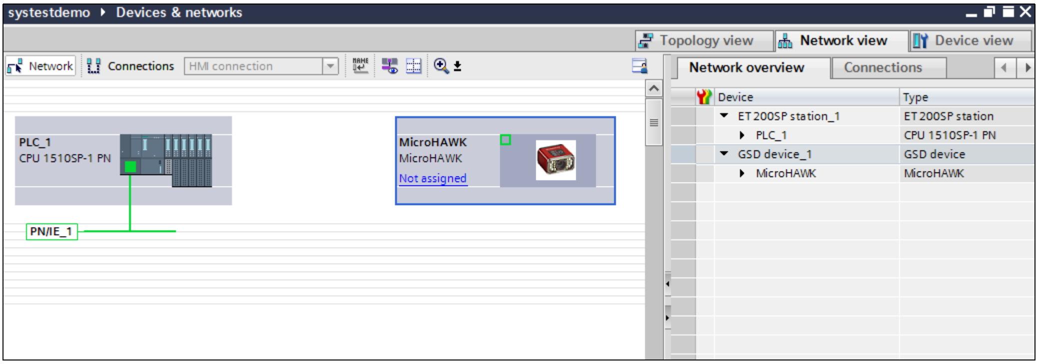



Browse to the following path: **C:\Microscan\Vscape\Firmware\gsd\MicroHAWK**. Select the appropriate GSD file by clicking the box next to the name, and click **Install**.

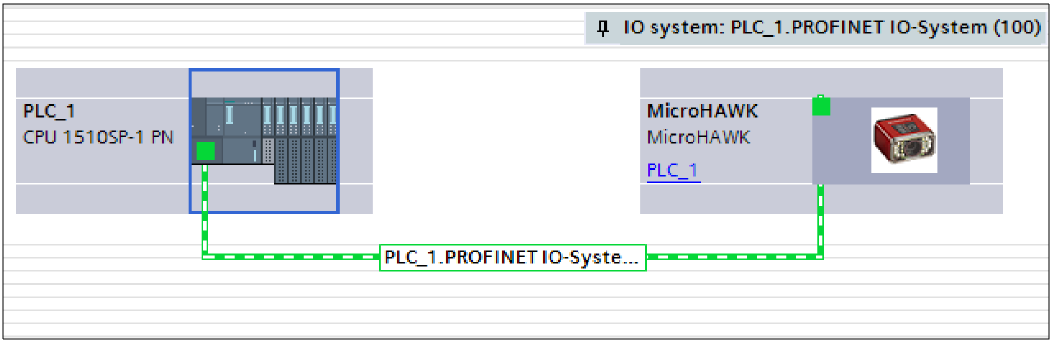
Click the tab that says **Network View**:



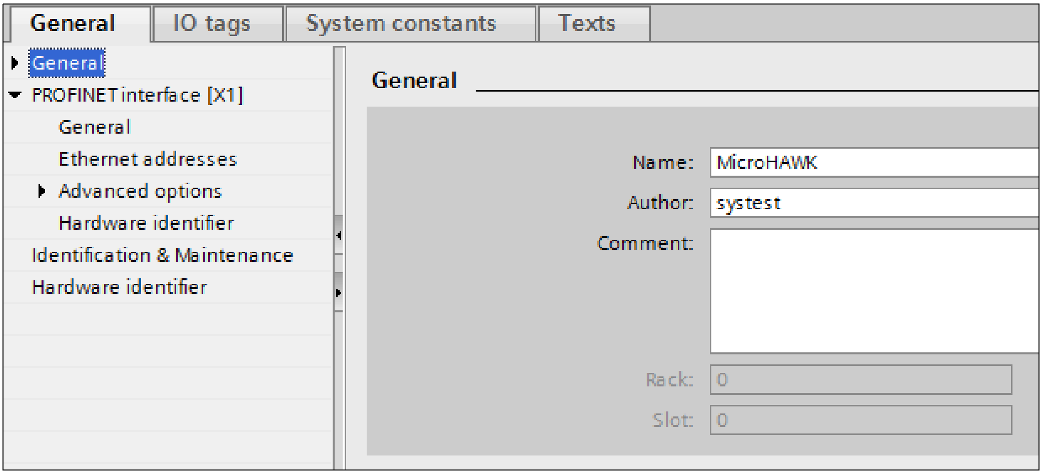
Look at the right panel and click **Hardware Catalog**. Use the arrows to open up the following area. **Other field devices->PROFINET IO ->General->MICROSCAN->Smart Camera->MicroHAWK MV-40**. Left-click on MicroHAWK MV-40 and hold it. Drag it to the left into the Devices and networks panel. Your screen will look like this:




On the MicroHAWK picture, click the **Not assigned** link, then click on **PLC_1.PROFINET IO-System**. This has just connected the MicroHAWK to the PROFINET network.



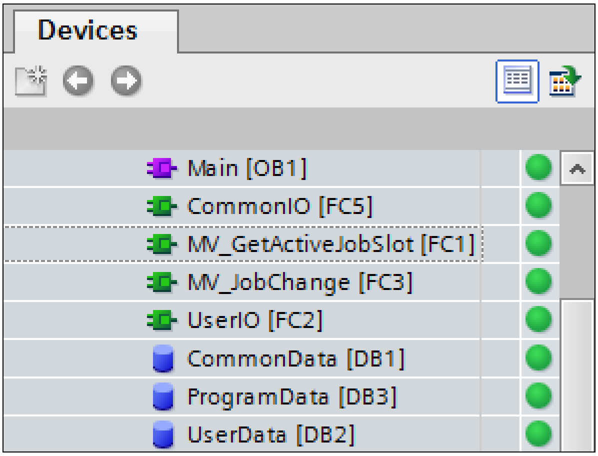
Double-click the picture of the MicroHAWK. In the bottom pane you will see the following:




Enter the name of the device in the Name: field. Then click Ethernet addresses on the left, and type in the IP address of the camera. This will allow the PLC to set the camera IP.

## Importing an Example Program

From TIA Portal, open the example program from the menu **Project > Open**. Click the **Browse** button to locate the (**Circle_LocatorDemo.ap13**) program. The Circle_LocatorDemo is located in **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\TIAPortal_Demos\Circle_LocatorDemo\TIACircle_LocatorDemo**. In the left pane in the **Devices** tab, open the PLC_1[CPU 1510SP-1 PN] arrow. Open the Program blocks arrow. Select all the blocks as shown below.

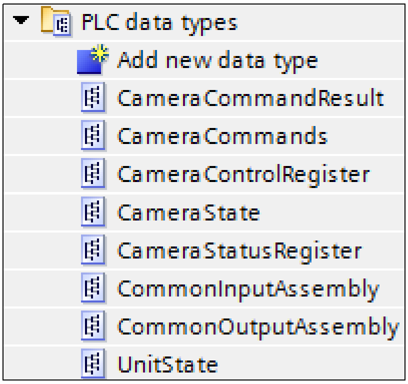


Right-click and copy. Go to your new project and open the same arrows, right-click on Program blocks, then click paste. <u>Be sure to save your work</u>. You may need to delete the Main [OB1] in the new project first.

Now back in the left pane of the Circle_LocatorDemo project, click on PLC data types, right-click, copy, and paste them into the new project PLC data types.

In the left pane of the Circle_LocatorDemo project, click on Watch and force tables. Right-click on Watch table_1, then copy. Paste into new project Watch and force tables. <u>Be sure to save the project</u>.

Your data in the new project should look like this:

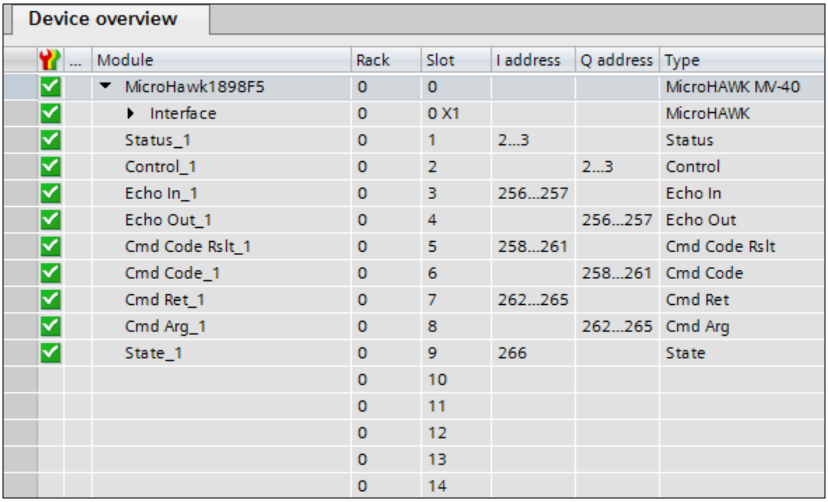


## Checking the Mapping of IO

Depending on the order the PLC can load things, we may need to change the IO addresses in our block to match the Device Overview in the Device view screen.

Double-click on the MicroHAWK picture, and the Device overview will be displayed to the right.

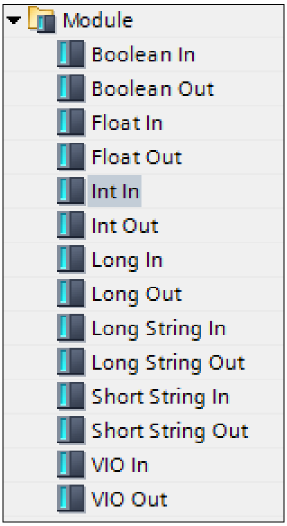
The image below is intended as an example only and may not exactly resemble your Device Overview.



Device overview

| Module | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|
| MicroHawk1898F5 | 0 | 0 | | | MicroHAWK MV-40 |
| Interface | 0 | 0 X1 | | | MicroHAWK |
| Status_1 | 0 | 1 | 2...3 | | Status |
| Control_1 | 0 | 2 | | 2...3 | Control |
| Echo In_1 | 0 | 3 | 256...257 | | Echo In |
| Echo Out_1 | 0 | 4 | | 256...257 | Echo Out |
| Cmd Code Rslt_1 | 0 | 5 | 258...261 | | Cmd Code Rslt |
| Cmd Code_1 | 0 | 6 | | 258...261 | Cmd Code |
| Cmd Ret_1 | 0 | 7 | 262...265 | | Cmd Ret |
| Cmd Arg_1 | 0 | 8 | | 262...265 | Cmd Arg |
| State_1 | 0 | 9 | 266 | | State |
| | 0 | 10 | | | |
| | 0 | 11 | | | |
| | 0 | 12 | | | |
| | 0 | 13 | | | |
| | 0 | 14 | | | |

From the Hardware catalog display to the far right, open the **Module** selection. You will see the following:



You must manually drag and drop “Boolean In_1” and “Float In_1” into the Device Overview pane. Click on “Boolean In_1”, and TIA will highlight the slot to drag it to. Repeat this for “Float In_1”. TIA will only allow you to insert them into their proper slot position.

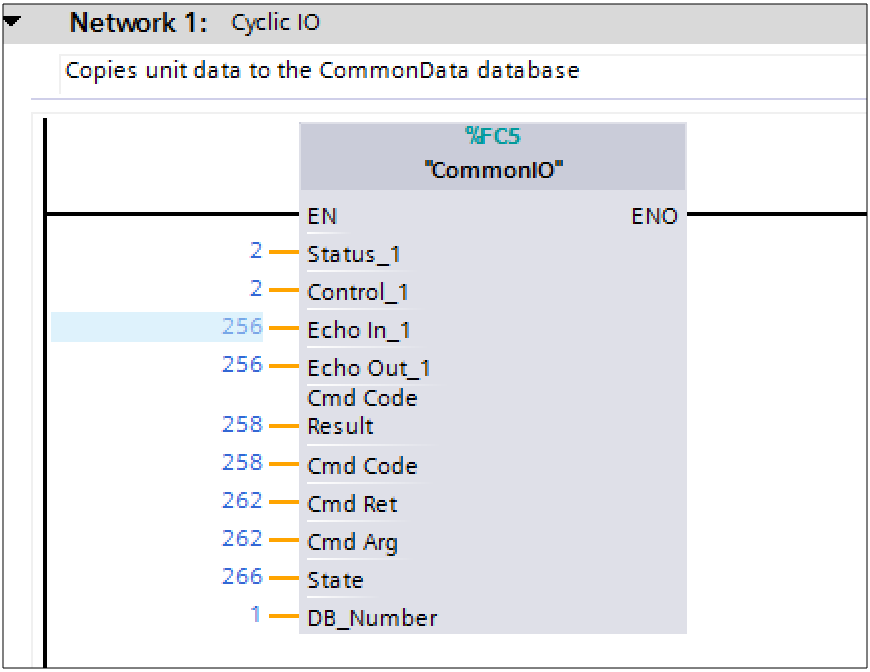
Your device view should now look as follows. The example addresses shown will not be the same for your device.

**Device overview**

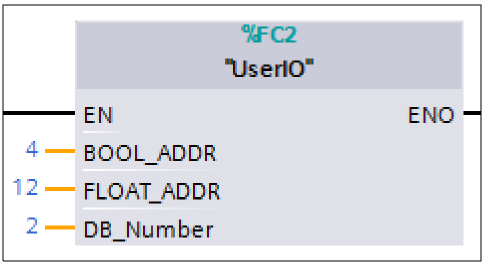
| Module | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|
| MicroHawk1898F5 | 0 | 0 | | | MicroHAWK MV-40 |
| Interface | 0 | 0 X1 | | | MicroHAWK |
| Status_1 | 0 | 1 | 2...3 | | Status |
| Control_1 | 0 | 2 | | 2...3 | Control |
| Echo In_1 | 0 | 3 | 256...257 | | Echo In |
| Echo Out_1 | 0 | 4 | | 256...257 | Echo Out |
| Cmd Code Rslt_1 | 0 | 5 | 258...261 | | Cmd Code Rslt |
| Cmd Code_1 | 0 | 6 | | 258...261 | Cmd Code |
| Cmd Ret_1 | 0 | 7 | 262...265 | | Cmd Ret |
| Cmd Arg_1 | 0 | 8 | | 262...265 | Cmd Arg |
| State_1 | 0 | 9 | 266 | | State |
| | 0 | 10 | | | |
| | 0 | 11 | | | |
| | 0 | 12 | | | |
| Boolean In_1 | 0 | 13 | 4...11 | | Boolean In |
| | 0 | 14 | | | |
| | 0 | 15 | | | |
| | 0 | 16 | | | |
| | 0 | 17 | | | |
| | 0 | 18 | | | |
| Float In_1 | 0 | 19 | 12...107 | | Float In |
| | 0 | 20 | | | |
| | 0 | 21 | | | |
| | 0 | 22 | | | |
| | 0 | 23 | | | |
| | 0 | 24 | | | |

You need to modify the PLC program blocks to agree with this display. On the left under the Devices pane, open the Program blocks arrow, and double-click on OB1 [OB1]. This will open a screen in the middle pane. Scroll this middle pane down until you see Network 1.

Change the numbers here to match with the Device overview display, as shown in the example below. Leave the DB_Number as-is.

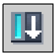



Repeat this for Network 4, using the Boolean In_1 address and the Float In_1address from the Device Overview and put it in the BOOL_ADDR and FLOAT_ADDR respectively. Leave the DB_Number as-is.

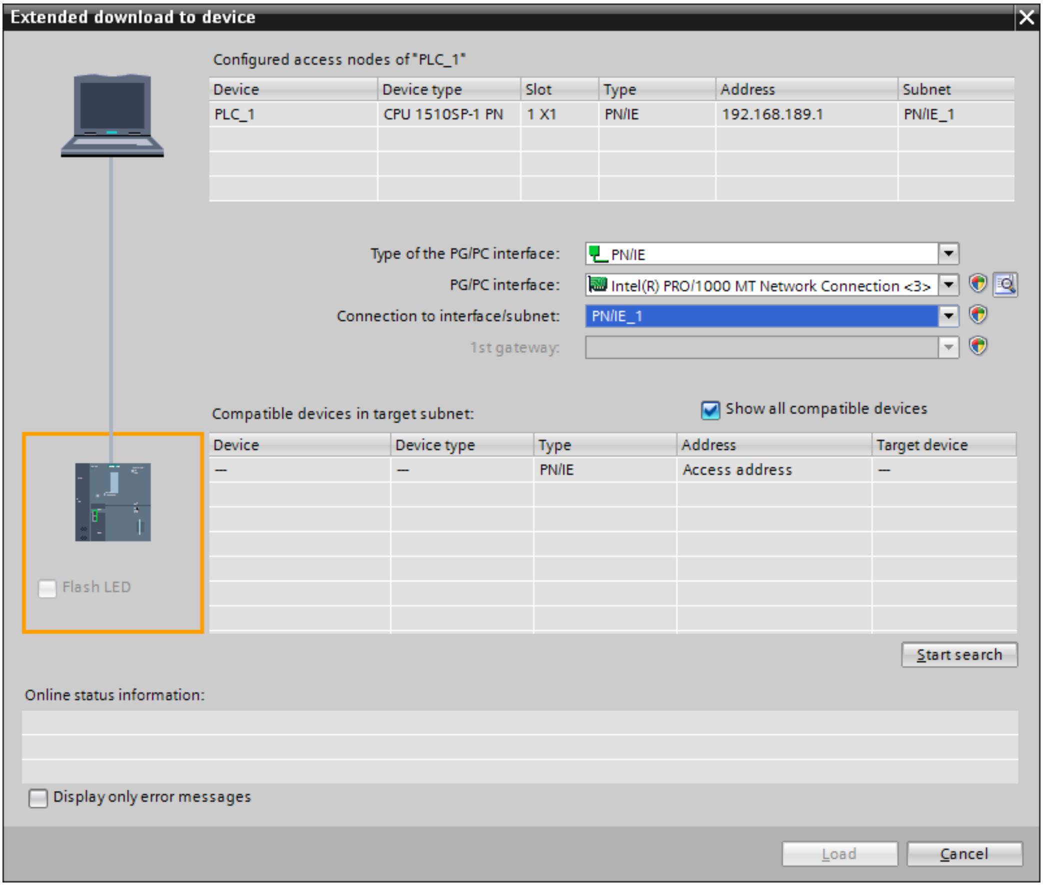


Be sure to save the project. Click the **PLC_1 {CPU 1510SP-1 PN]** top level, then click the compile icon.

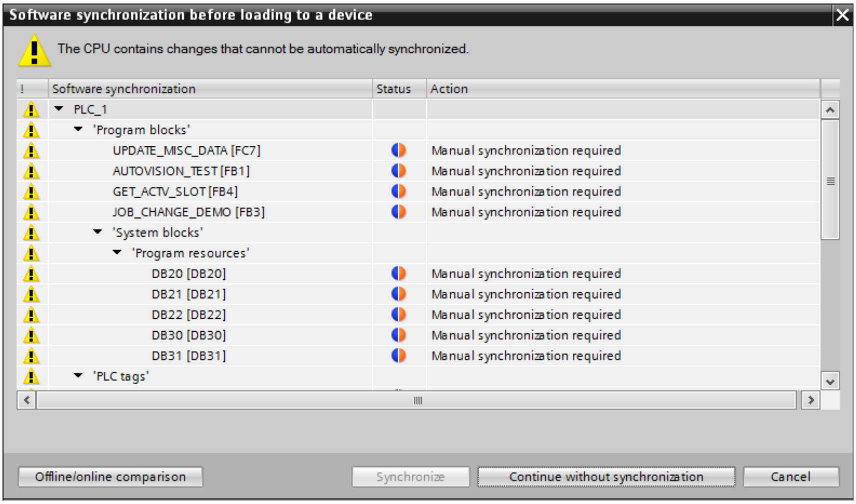
There should be NO compile errors. If there are no errors, click the download icon.



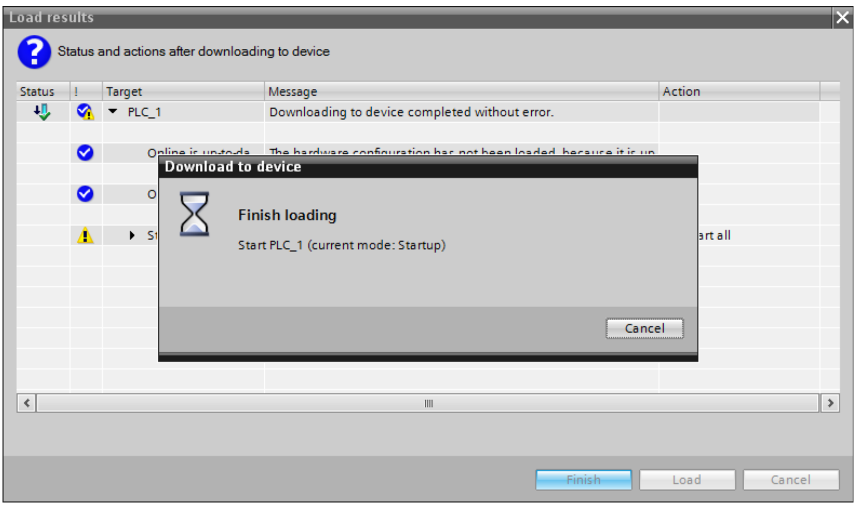
You will see the following screen:



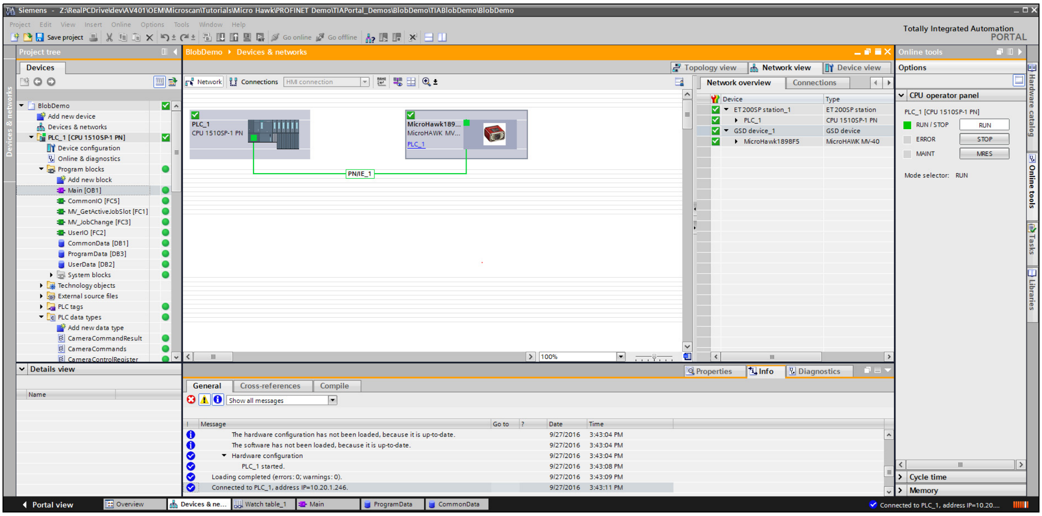
Select your PC network interface, and change the Connection to interface/subnet to **PN/IE_1**. Click **Start Search**. PLC_1 should be displayed with the address you gave it. Click **Load**. If you see a screen similar to below, do a memory reset (MRES) on the PLC, and try to load again. If this continues to be an issue, click **Continue without synchronization** and continue.




When downloading is working correctly you will see the following:



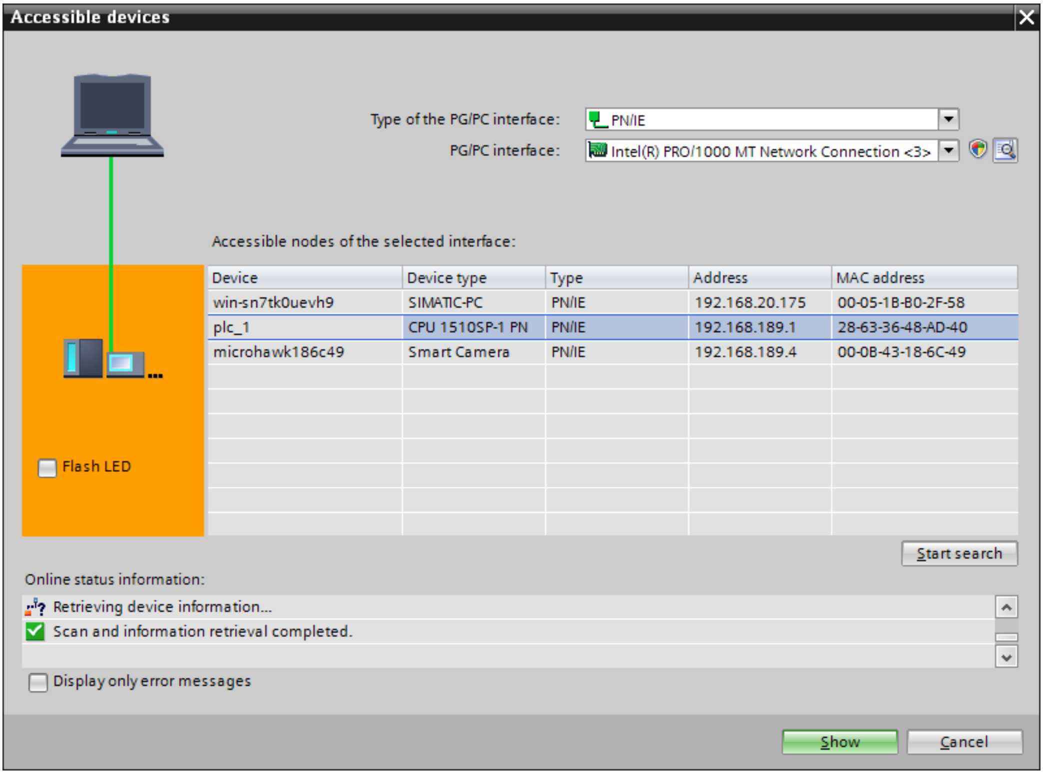
All LEDS on the PLC should be **GREEN**. Click **Go online**. You should see the following:



# Troubleshooting

If all LEDs are not green, go offline and click the Accessible devices icon.

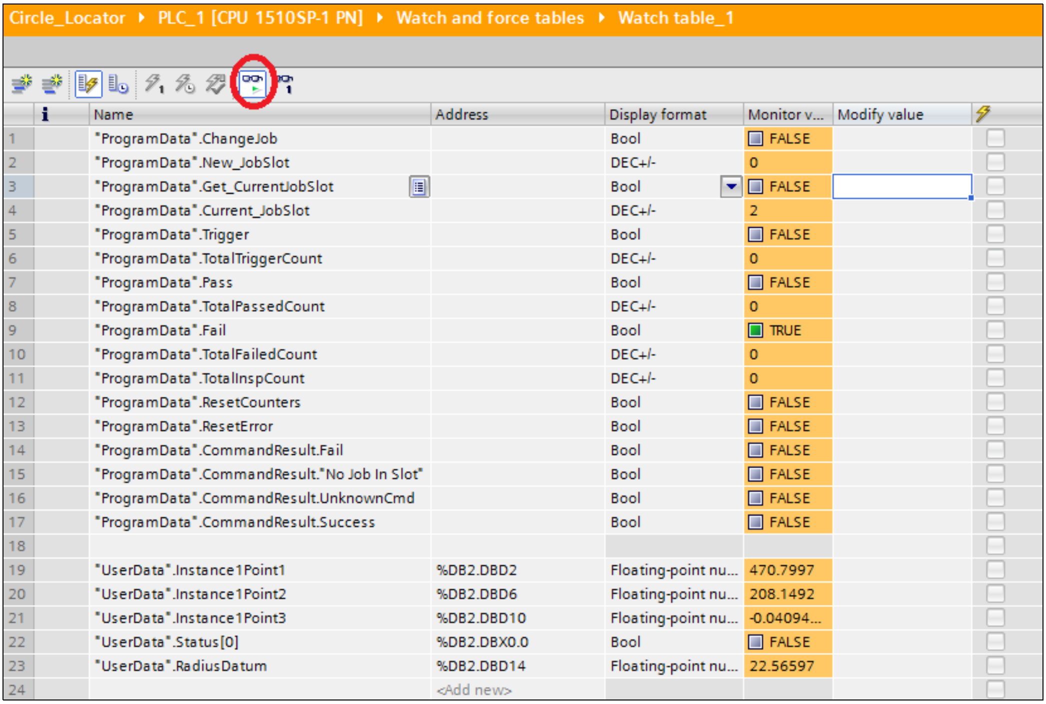
Click **Start search**. The PLC and the MicroHAWK should be displayed here.




If the name of the MicroHAWK is not correct, you will not connect. The default name is microhawk with the last 3 of the MAC address. Ex: microhawk186c49. Go back to the general settings of the MicroHAWK in the device view, and change the name if required. Re-compile, and re-load. Try running again. This is the most common reason for not connecting.

## Running the Demo

In the left pane click the Watch and force tables arrow. Double-click Watch table_1. The watch table will open in the center pane. Click on the monitor variables icon.



Circle_Locator ▸ PLC_1 [CPU 1510SP-1 PN] ▸ Watch and force tables ▸ Watch table_1

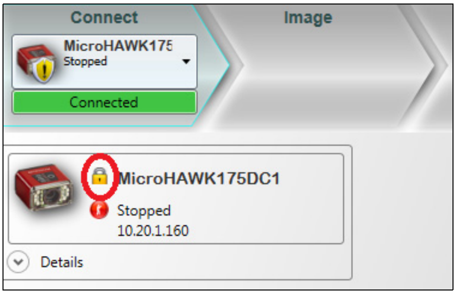
| | i | Name | Address | Display format | Monitor v... | Modify value | |
|---|---|---|---|---|---|---|---|
| 1 | | "ProgramData".ChangeJob | | Bool | FALSE | | |
| 2 | | "ProgramData".New_JobSlot | | DEC+/- | 0 | | |
| 3 | | "ProgramData".Get_CurrentJobSlot | | Bool | FALSE | | |
| 4 | | "ProgramData".Current_JobSlot | | DEC+/- | 2 | | |
| 5 | | "ProgramData".Trigger | | Bool | FALSE | | |
| 6 | | "ProgramData".TotalTriggerCount | | DEC+/- | 0 | | |
| 7 | | "ProgramData".Pass | | Bool | FALSE | | |
| 8 | | "ProgramData".TotalPassedCount | | DEC+/- | 0 | | |
| 9 | | "ProgramData".Fail | | Bool | TRUE | | |
| 10 | | "ProgramData".TotalFailedCount | | DEC+/- | 0 | | |
| 11 | | "ProgramData".TotalInspCount | | DEC+/- | 0 | | |
| 12 | | "ProgramData".ResetCounters | | Bool | FALSE | | |
| 13 | | "ProgramData".ResetError | | Bool | FALSE | | |
| 14 | | "ProgramData".CommandResult.Fail | | Bool | FALSE | | |
| 15 | | "ProgramData".CommandResult."No Job In Slot" | | Bool | FALSE | | |
| 16 | | "ProgramData".CommandResult.UnknownCmd | | Bool | FALSE | | |
| 17 | | "ProgramData".CommandResult.Success | | Bool | FALSE | | |
| 18 | | | | | | | |
| 19 | | "UserData".Instance1Point1 | %DB2.DBD2 | Floating-point nu... | 470.7997 | | |
| 20 | | "UserData".Instance1Point2 | %DB2.DBD6 | Floating-point nu... | 208.1492 | | |
| 21 | | "UserData".Instance1Point3 | %DB2.DBD10 | Floating-point nu... | -0.04094... | | |
| 22 | | "UserData".Status[0] | %DB2.DBX0.0 | Bool | FALSE | | |
| 23 | | "UserData".RadiusDatum | %DB2.DBD14 | Floating-point nu... | 22.56597 | | |
| 24 | | | <Add new> | | | | |

The following operations can be tried here:

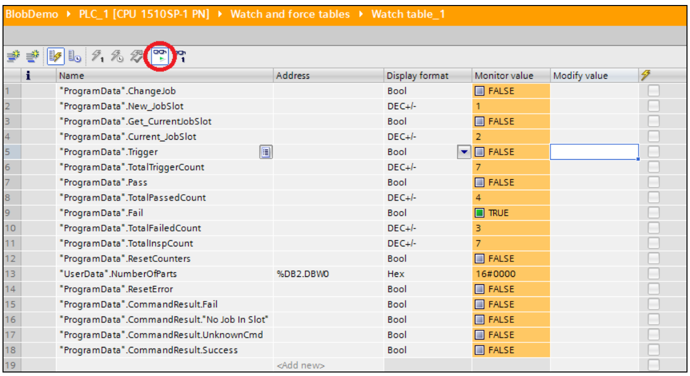
| | |
|---|---|
| Change Job | Switch jobs from one slot to another |
| Get Current Job Slot | Displays the slot of the currently running job |
| Trigger an Inspection | Causes the camera to cycle through its inspection. |
| Reset Counters | Resets the TotalTriggerCount, TotalPassedCount, TotalFailedCount, and TotalInspCount to 0, and also sends a command to the camera to reset its internal counters. |
| Reset an Error condition | If an error occurs during run time, this will reset it, so demo may be continued. |

## Change Jobs + Get Current Job Slot

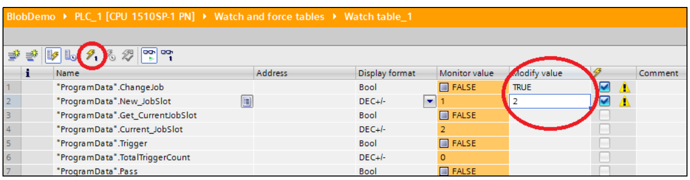
Be sure to store a job in a slot in the camera memory as described earlier. To execute this test properly, load another job into **Slot 2** of the camera. For example, the **BlobDemo** job. You must release control of the device by clicking on the **Connect** tab and change the lock icon to **closed** as in the following image, then click the **Run** tab again for job change to work properly.



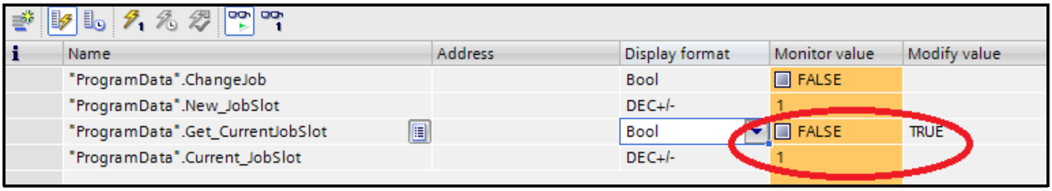

Go to **Watch table_1**, and click the button with the glasses and "play" symbol. This begins the monitoring of data.




Using the Watch table_1 elements **ChangeJob** and **New_JobSlot**, click on the **Modify Value** column next to **New_JobSlot**, type 2, and type **Enter**. Click on the Modify value to the right of ChangeJob and enter **TRUE**. Click on the lightning bolt with a **1** at the top. This will set the job slot to 2. Then click the lightning bolt button again.



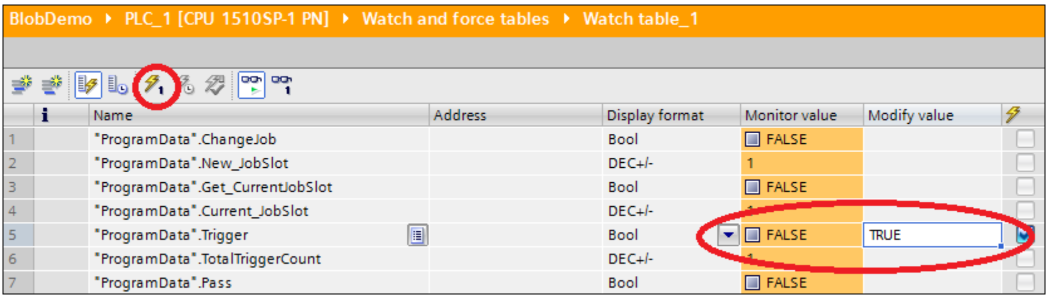
The current job will be changed to the job in slot 2. To verify that this clears all changes made to the Modify value column, click on Modify value to the right of **Get_CurrentJobSlot**. Enter **TRUE**, and click on the lightning bolt button again.




You will see the current job listed to the right of **Current_JobSlot**. You can repeat this procedure, changing the slot back to **1** and verify that the current job slot is now **1**.

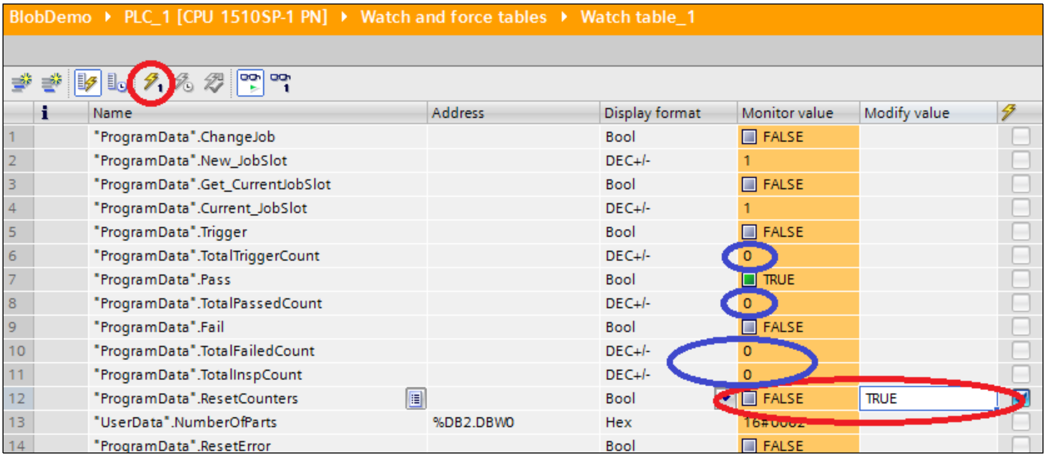
### Trigger an Inspection and Reset Counters

Clear any changes made to the Modify value column. If you want to watch the inspection, be sure AutoVISION is running and connected to the device. Be sure the job is running. If you choose not to run AutoVISION, you can look at the LEDs on the device to confirm your triggers. Change the value to the right of **Trigger** to be **TRUE**, and click the lightning bolt button again.




You will see the **InstancePoints**, **Status**, and **RadiusDatum** change based on the picture. The pass and fail bools will indicate whether the inspection has passed or failed. Also, the appropriate counters will have incremented. Each click of the lightning bolt button will cause an inspection which will bounce from one image to the next. One will pass and the next will fail.

To reset the counters, clear all settings in the Modify value column, and change the one next to **ResetCounters** to **TRUE**, and click the lightning bolt button again. You will see all the counters in blue reset to **0**.

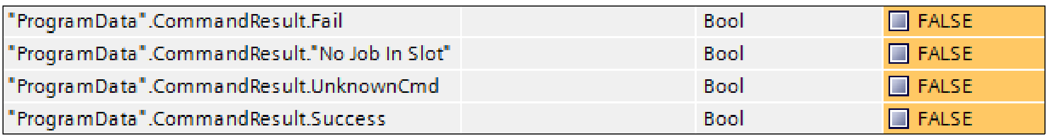

BlobDemo ▸ PLC_1 [CPU 1510SP-1 PN] ▸ Watch and force tables ▸ Watch table_1

| | i | Name | Address | Display format | Monitor value | Modify value |
|---|---|---|---|---|---|---|
| 1 | | "ProgramData".ChangeJob | | Bool | FALSE | |
| 2 | | "ProgramData".New_JobSlot | | DEC+/- | 1 | |
| 3 | | "ProgramData".Get_CurrentJobSlot | | Bool | FALSE | |
| 4 | | "ProgramData".Current_JobSlot | | DEC+/- | 1 | |
| 5 | | "ProgramData".Trigger | | Bool | FALSE | |
| 6 | | "ProgramData".TotalTriggerCount | | DEC+/- | 0 | |
| 7 | | "ProgramData".Pass | | Bool | TRUE | |
| 8 | | "ProgramData".TotalPassedCount | | DEC+/- | 0 | |
| 9 | | "ProgramData".Fail | | Bool | FALSE | |
| 10 | | "ProgramData".TotalFailedCount | | DEC+/- | 0 | |
| 11 | | "ProgramData".TotalInspCount | | DEC+/- | 0 | |
| 12 | | "ProgramData".ResetCounters | | Bool | FALSE | TRUE |
| 13 | | "UserData".NumberOfParts | %DB2.DBW0 | Hex | 16#0002 | |
| 14 | | "ProgramData".ResetError | | Bool | FALSE | |

If you clear this again, set the Trigger to TRUE, and click the lightning bolt button again, you will notice that the counts start at **1** on both the PLC counters and AutoVISION counters.

### Reset Error

If at any time there is an error indicated by a TRUE in any of the values below, you can clear the error by entering a TRUE to the right of **ResetError** and clicking the lightning bolt button. Reset it to **FALSE** and click the button again. Any errors in the values below should be cleared.



| | | | |
|---|---|---|---|
| "ProgramData".CommandResult.Fail | | Bool | FALSE |
| "ProgramData".CommandResult."No Job In Slot" | | Bool | FALSE |
| "ProgramData".CommandResult.UnknownCmd | | Bool | FALSE |
| "ProgramData".CommandResult.Success | | Bool | FALSE |

11

CHAPTER 11

# PROFINET I/O Blob Count Demo Using Step 7 and MicroHAWK

This section describes how to use Omron Microscan demo PLC code with a vision job and camera target.

The PROFINET I/O demo files can be found where AutoVISION is installed, in the folder: **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\Step7_Demos\BlobDemo**. Open **BlobDemo.avp** and download it to the camera.

**Notes:**

- The camera communications protocol must be enabled for PROFINET I/O before it can be used in this environment. Refer to **Chapter 1, Enabling Industrial Protocols**, for information about enabling and switching communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate PROFINET I/O communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.
  **Important:** PROFINET I/O allows the IP address and subnet mask of the camera to be assigned automatically by the PLC. In order to allow this, the MicroHAWK may not have an assigned IP address on reboot until the PLC is set to Run mode. During this time, the MicroHAWK will not be visible on the network for AutoVISION or Visionscape FrontRunner.

# Overview

In this demonstration, you will learn how to load a saved job into the camera, establish connectivity via **PROFINET I/O** to a **Siemens S7 PLC**, and run some example programs that interface with the camera.

While evaluating PROFINET I/O capabilities, you will:

- **Begin with AutoVISION.**

Open a sample demo vision job in AutoVISION and use the Try Out feature to learn what to expect from the camera before it is connected to the PLC.

- **Prepare the PLC.**

Integrate the camera into the PLC environment with STEP 7 software and the GSD file.

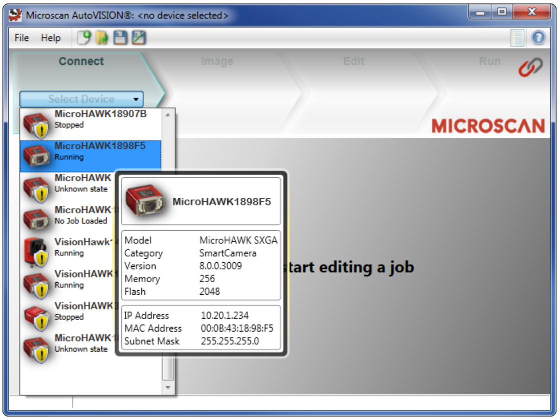
- **Run the camera.**

Trigger the camera while it is online with the PLC and observe changes in the Inspection status as the PLC reconfigures the vision job's parameters.

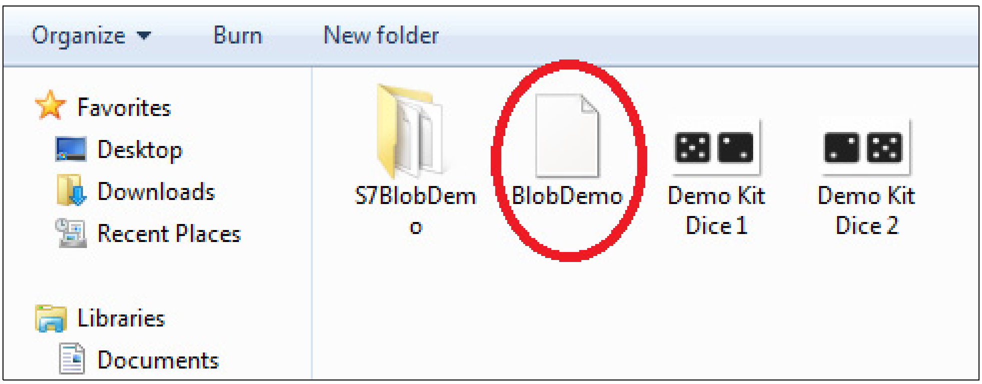
# AutoVISION Setup

Prior to starting AutoVISION, make sure the camera is either connected to the PLC or both PLC and camera are on the same physical network. Ensure that the PC, PLC, and camera have the same network class and corresponding subnet addresses.

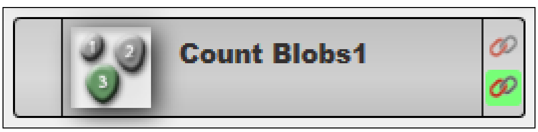
Open AutoVISION and select the camera.




From the Image view, click the **Load a Job** button. Then navigate to **C:\Microscan\Vscape\Tutorials And Samples\MicroHAWK\PROFINET Demo\Step7_Demos\BlobDemo**. Select **BlobDemo.avp**.




The demo job will include one tool: **Count Blobs**.

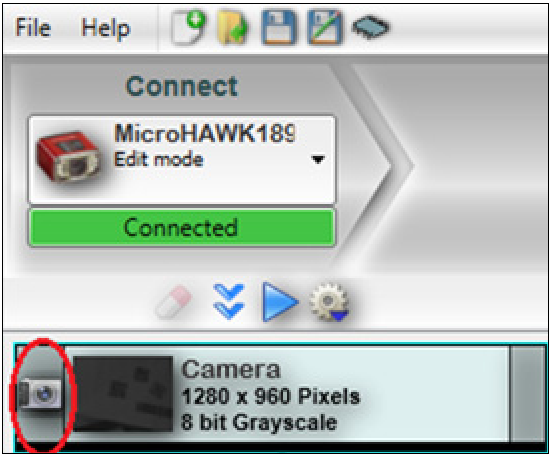


The Blob tool data item linked to the PROFINET I/O structure as shown here:

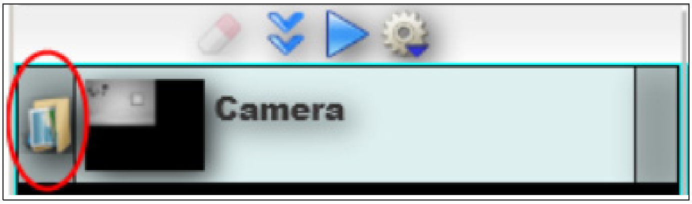
| Tool Result | PLC IN |
|---|---|
| Number of parts | "UserData".NumberOfParts (%DB2.DBW0) |

This data is transferred cyclically between the camera and PLC.

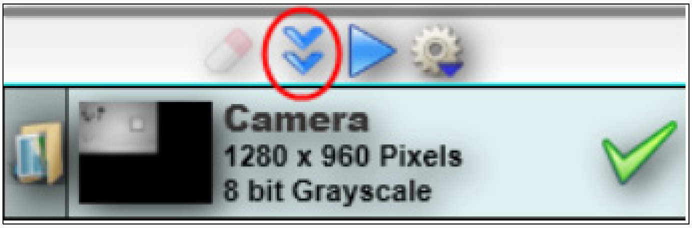
Once the job has loaded, the next step is to link the pre-saved images on the local PC. On the **Camera** button, click the far left icon to select and load an image.



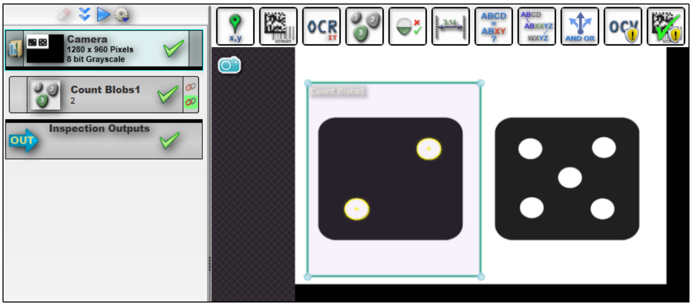

A file browser will open. Then navigate to the same folder where the demo job was loaded PROFINET I/O Demo. If the images are located, the icon will change from a camera to a folder.

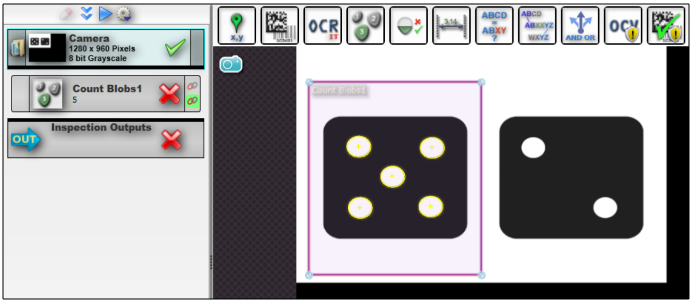



By clicking the **Try Job Once** icon, the application will cycle through the entire job with the loaded image.



There are two images on file. One will pass the inspection and one will fail.




Now click the **Run** button on the top ribbon. This will download the job to the camera. At this point the job is ready to run and can be tested. However, in order to run the job change demo this job needs to be loaded into **Slot 1**. Click back to the **Edit** view from the top ribbon bar.

Click the slot icon and select Slot 1 or **New Slot** if no slots are currently in use.

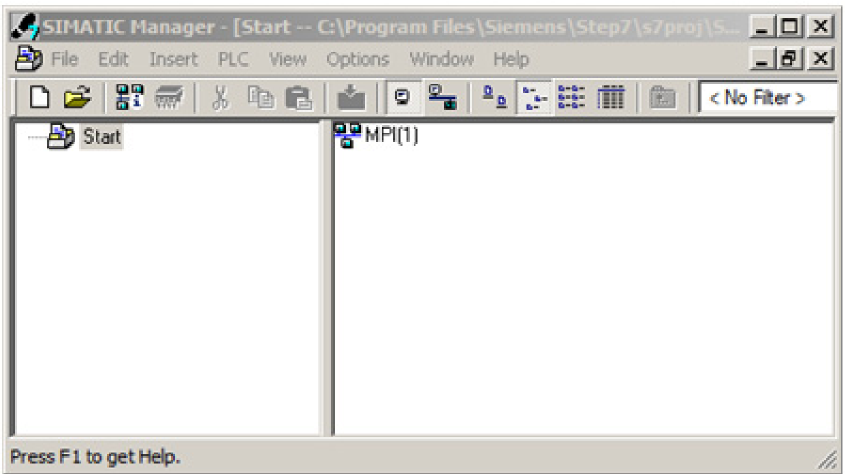

Now the job and images will be saved to the flash memory of the camera.

# STEP 7 Setup

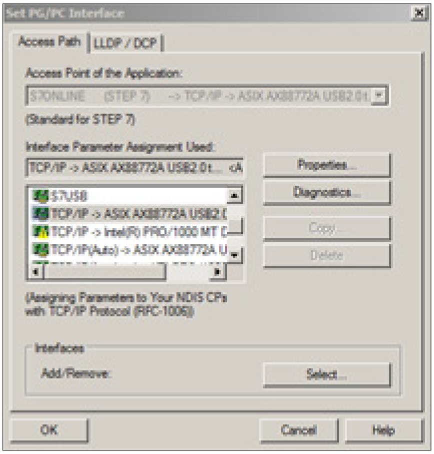
This section was created using Siemens STEP 7 software version 5.5 + SP2 and an ET200S PLC, catalog number 6ES7 151-8AB01-0AB0, CPU Version 3.2. It was tested with a 315-2 PN/DP PLC, catalog number 6ES7-315-2EH13-0AB0, CPU Version 2.6.

First launch the SIMATIC Manager from the desktop.

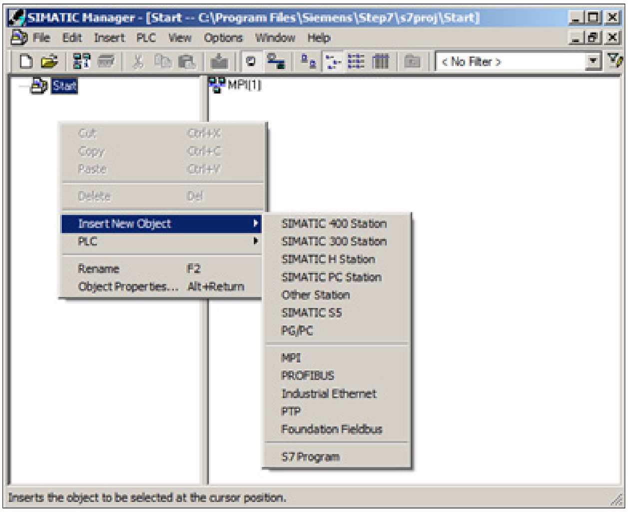

Create a new project from the menu item **File > New**. Select the project location on disk then enter the name and click the **OK** button. In the example below, the name is **Start**. Once the project is first created, you will see the dialog. This dialog is the main entry point into the PLC program and hardware settings.



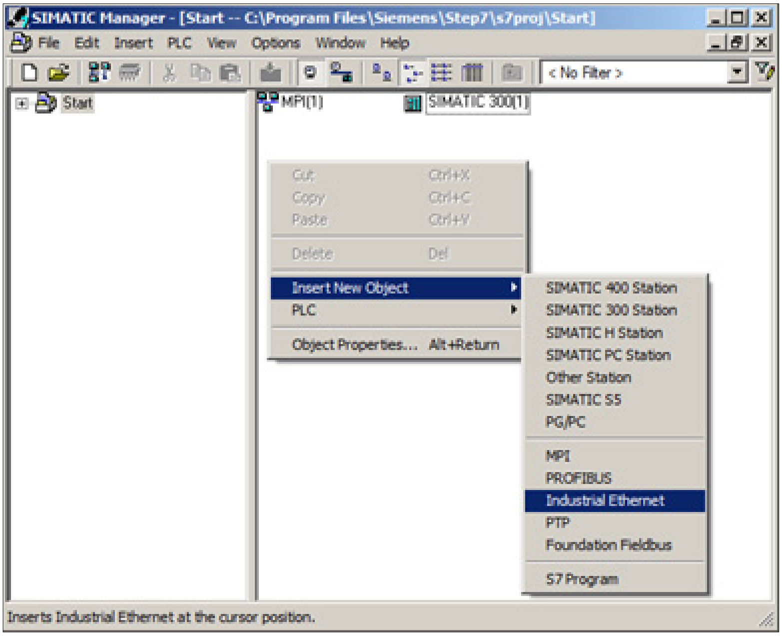
The PC may have multiple network cards so it's imperative to map the correct one to the SIMATIC software. From the menu, select Options > Set PG/PC Interface. This will open the Set PG/PC Interface dialog and list the available network cards. On the Access Path tab select the NIC card with (TCP/IP >) in the name.



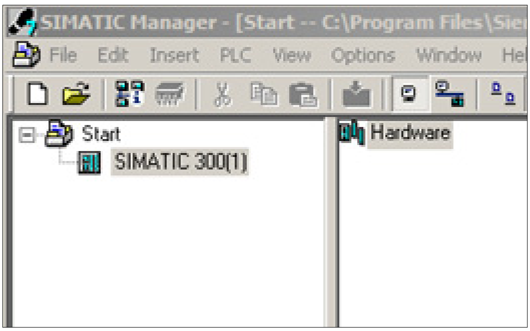

In the SIMATIC Manager dialog, right-click and select Insert New Object. This is where you will select the base station. For example, when configuring an ET200-s, select SIMATIC 300 Station since it's based on the 300 series CPU.



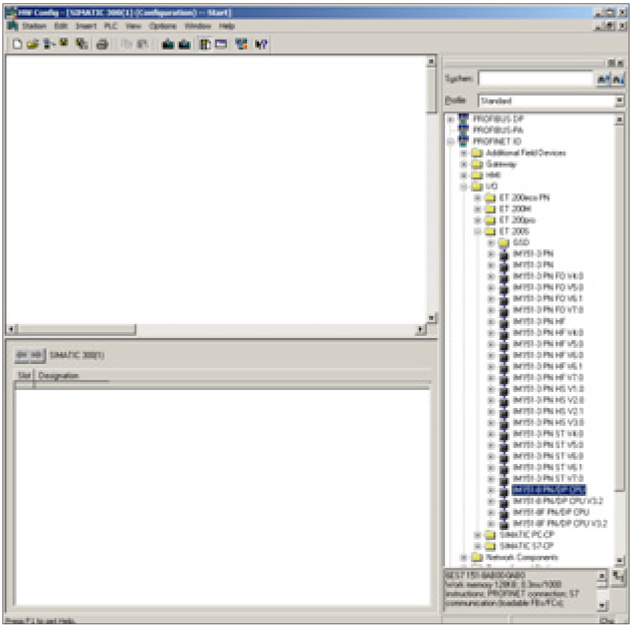
After the station has been added, right-click and add **Industrial Ethernet**.



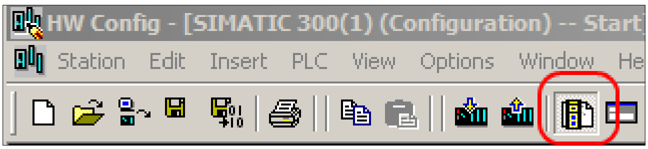

Expand the Start tree node on the left pane and click the station. On the right pane you will see a Hardware icon.



Double-click the **Hardware** icon to launch the **HW Config** dialog.



Make sure the **Catalog** is selected in the ribbon bar on the top. This will add a tree view on the right pane with all the available hardware devices.

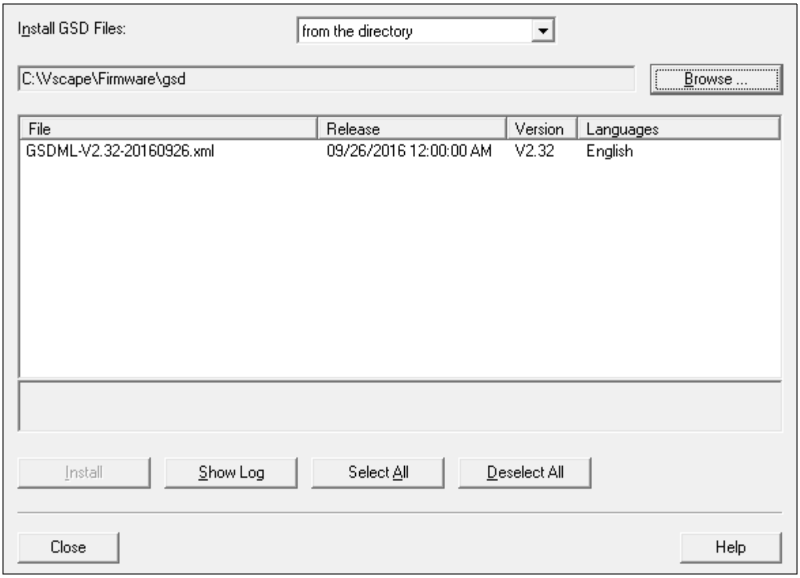



If the latest GSDML file hasn't been imported, follow the next steps to import.

## GSDML

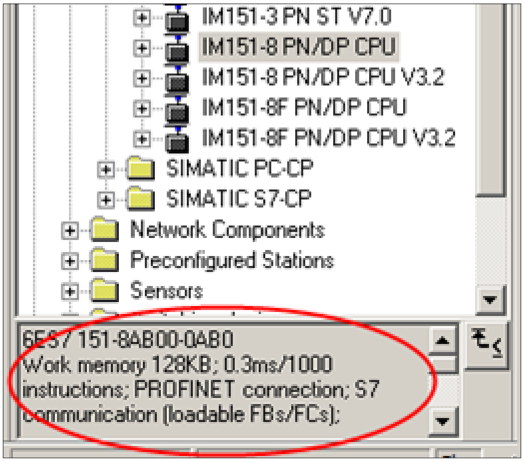
- Install the GSD file from the menu item **Options > Install GSD File**.
- Click the Browse button to locate the GSDML file **\Microscan\Vscape\Firmware\gsd\MicroHAWK**.

- The center pane will list the available GSDML files.




- Select the file and click the **Install** button.
- When finished, close the dialog.
- The camera should now be present in the PROFINET I/O section under **Additional Field Devices > General**.

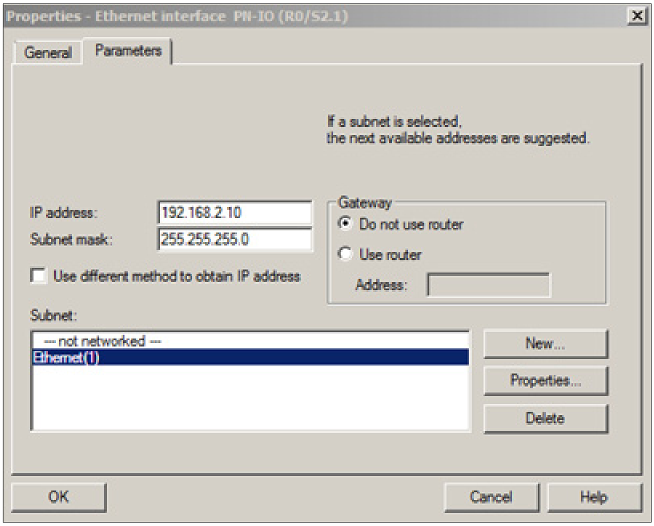
Add the CPU from the catalog view by drag and drop or double-click. Make sure the catalog number and version matches the PLC exactly. The catalog number will be displayed on the bottom of the view.




Some CPUs are modules that will require a generic rack to be added prior. If your PLC requires a rack, you will be prompted to add the rack prior to being allowed to insert the CPU module.

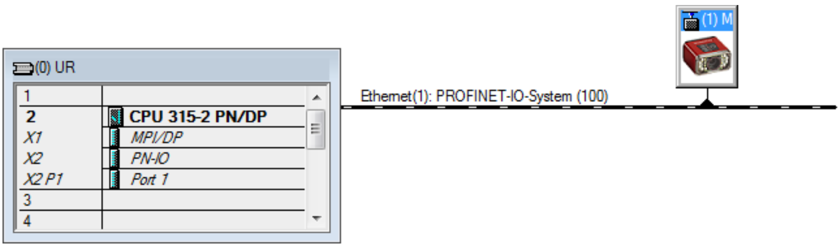
After the CPU is added, a popup dialog will prompt the Properties relating to the IP information.

Select **Ethernet(1)** on the bottom list box and enter the correct IP address of the CPU.

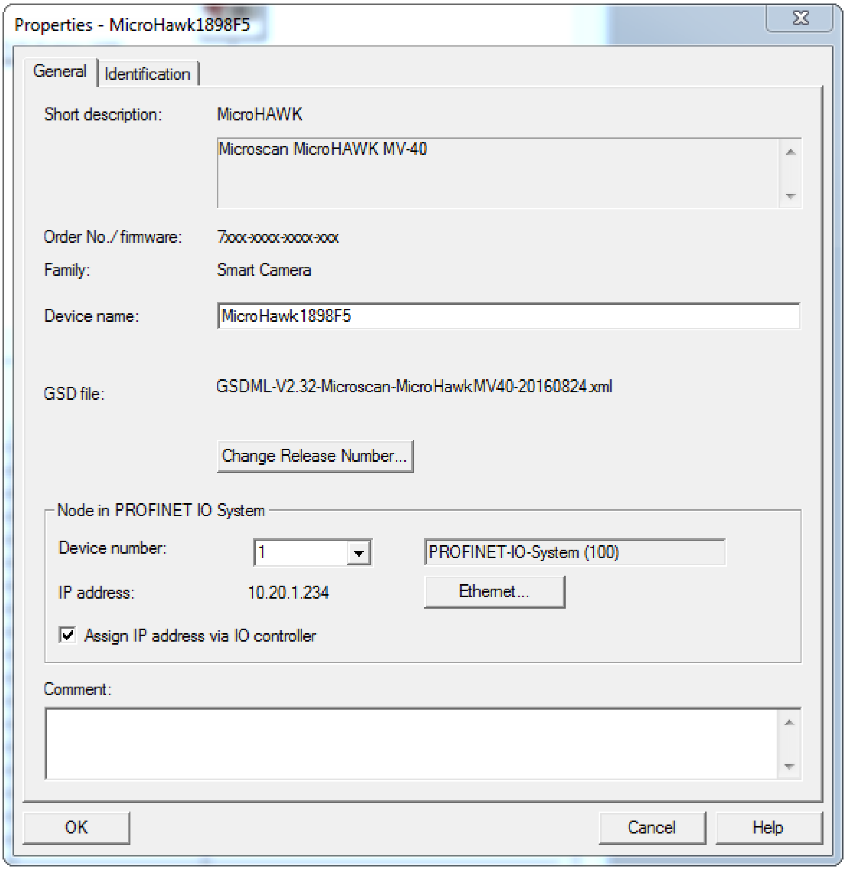



Once the CPU has been added, add any additional expansion modules, if installed. Click the CPU on the left pane and the bottom pane will list the available ports and slots.

Add the camera in the location (**PROFINET I/O > Additional Field Devices > General > Microscan**). The camera can be added by drag and drop to **Ethernet(1)**: **PROFINET-I/O System (100)** or by selecting the **Ethernet(1)** line first and double-clicking the camera. Once the camera has been added the icon will appear on the configuration dialog.

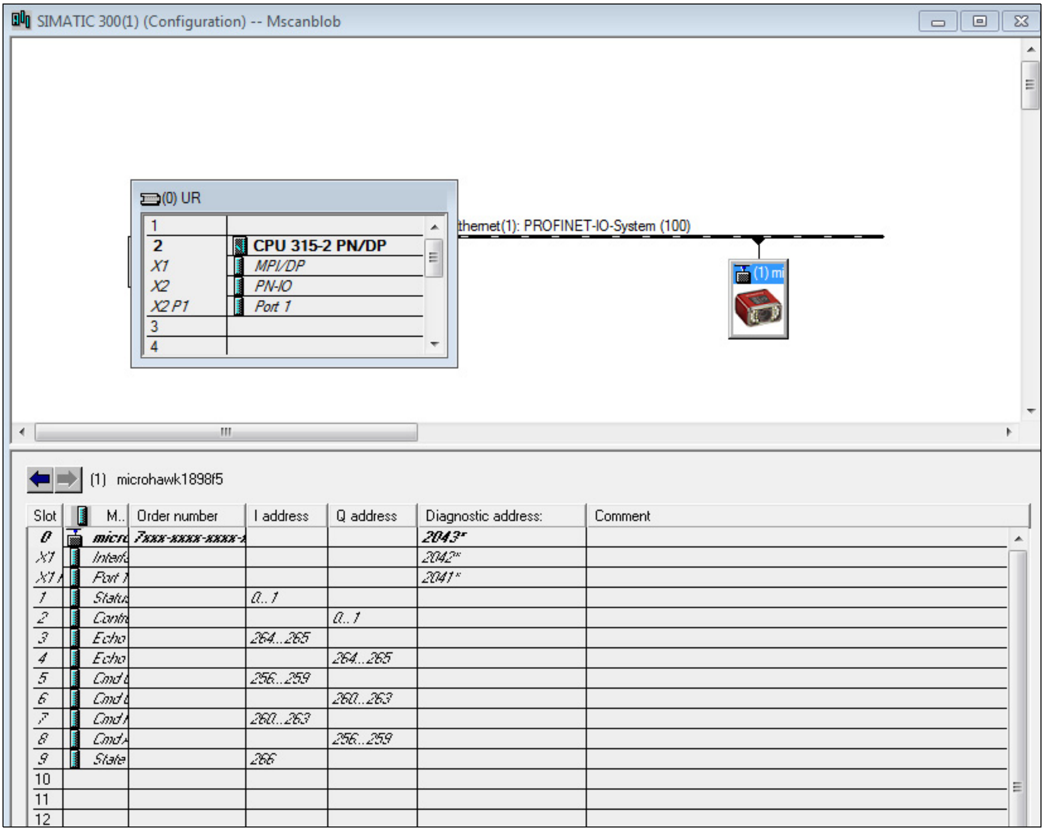


Once the camera has been added, double-click the icon to open the properties dialog. Under the device name, enter the existing name of the camera or a unique name.

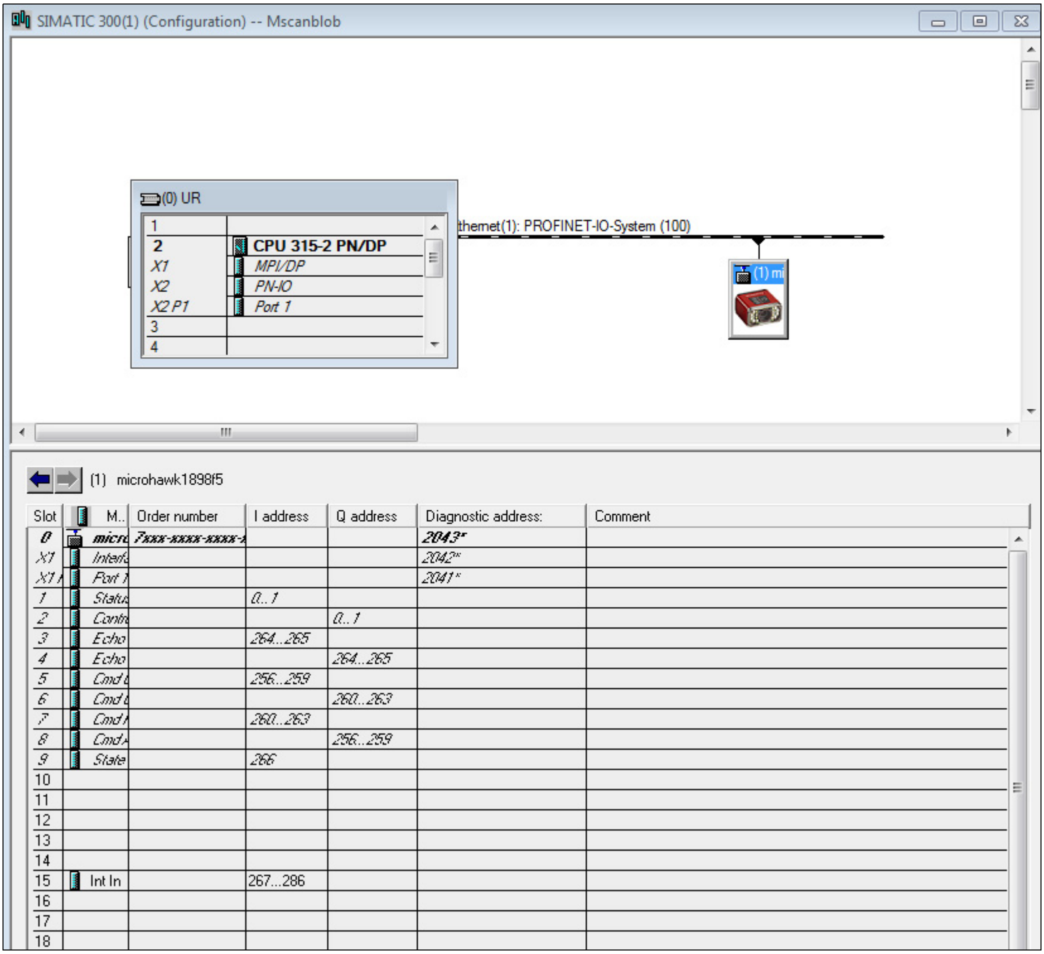



If a unique name is used, the device has to be manually updated. View Updating camera name section.

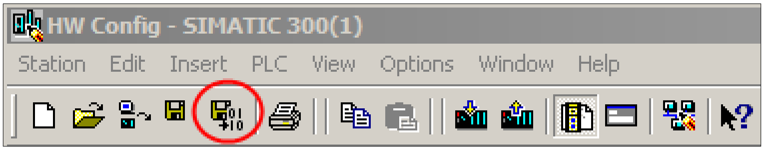
Click the camera icon and the data slot address mapping will be displayed below. Take note of the address values since they will be needed in the demo application. Since there is an infinite combination of modules and slot configurations, the addressing is unique to every setup.



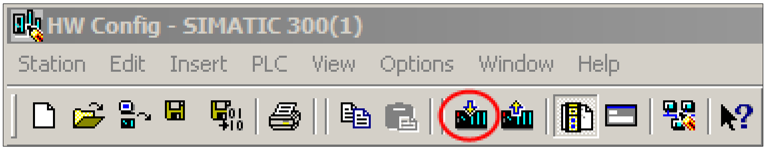
This demo uses the **Int In** data. To the right under PROFINET I/O MicroHAWK, click Int In. This will highlight its appropriate slot. Drag and drop this highlighted slot. The table should resemble the following:



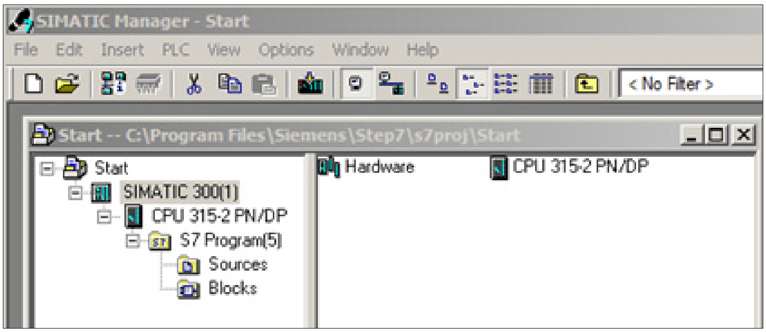
Once the hardware configuration has been completed, it's time to compile and download. Click the compile and save icon on the ribbon. If there are any configuration conflicts, the application will prompt a warning at this point.



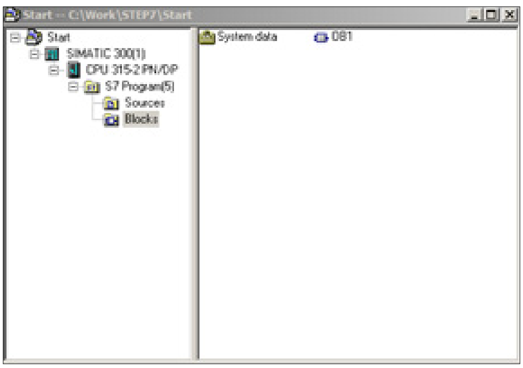

Once compiled, click the download icon on the ribbon to send the information to the PLC.




At this point close or minimize the **HW Config** dialog and re-visit the **SIMATIC Manager** dialog. The CPU should be added next to the Hardware icon and in the tree view in the left pane. Expand the CPU tree item and remaining child items below it.



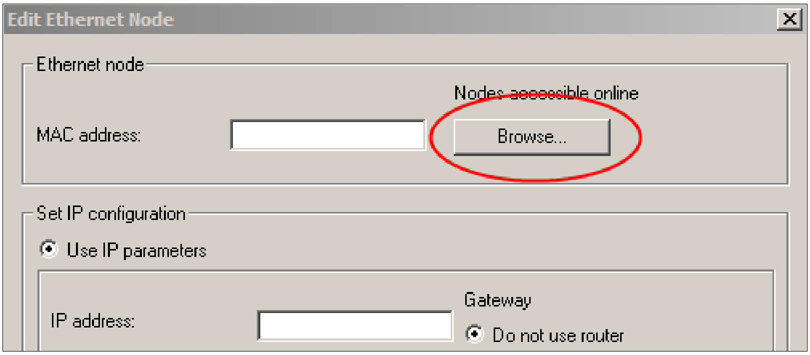

Click the Blocks node to view the program objects.



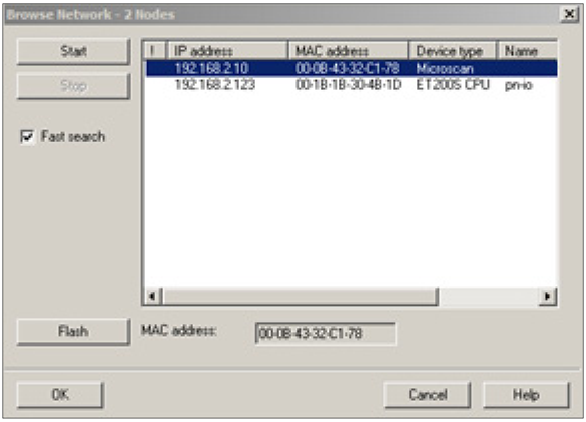
## Updating Camera Name

From the HW Config dialog open **Edit Ethernet Node** from the menu item **PLC > Ethernet > Edit Ethernet Node**. From the SIMATIC Manager dialog open Edit Ethernet Node from the menu item **PLC > Edit Ethernet Node**.

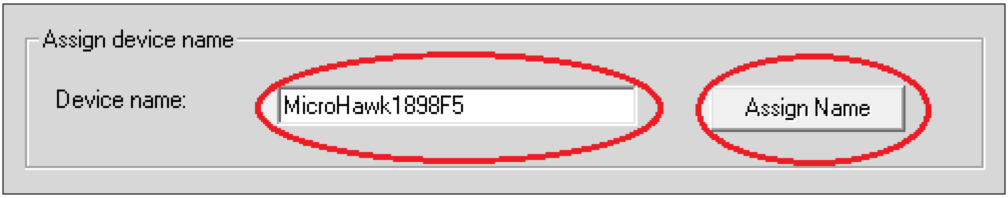
Click the **Browse** button to open the node selection dialog.




Select the camera and click the **OK** button.

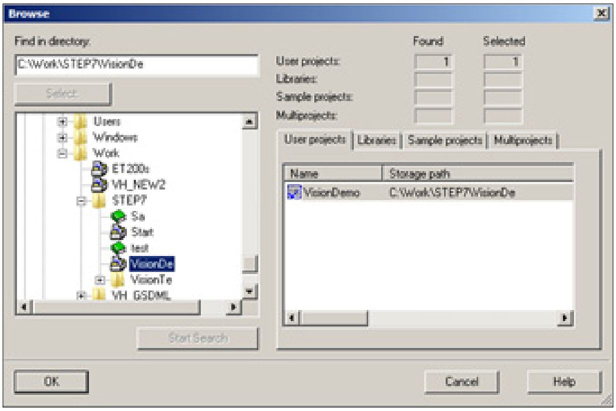



Edit the **Device Name**, if necessary, to match the name entered previously in the MicroHAWK properties dialog, and click the **Assign Name** button.

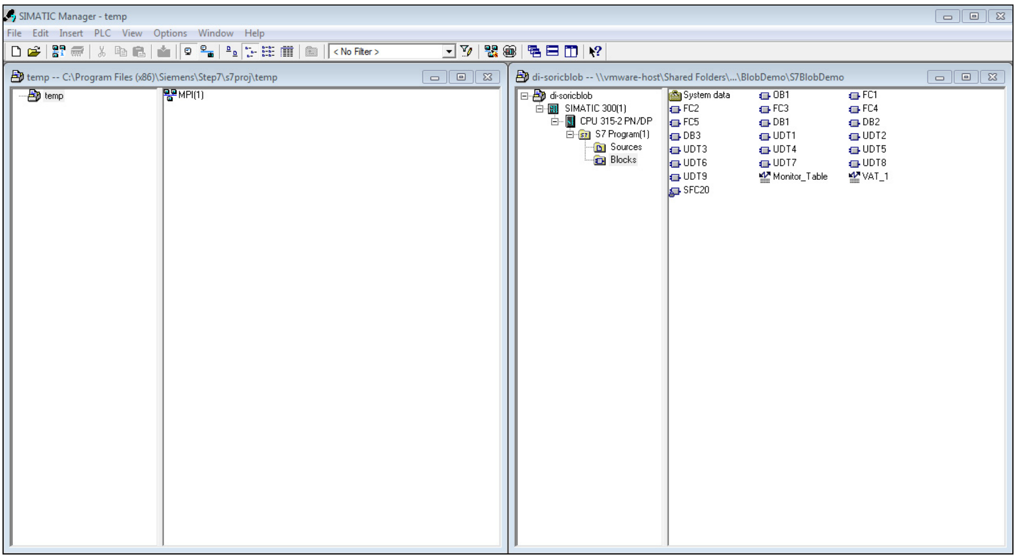


## Importing Example Program

From the SIMATIC Manager, open the example program from the menu **File > Open**. Click the **Browse** button to locate the (BlobDemo) program. The BlobDemo is located in **\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\Step7_Demos\BlobDemo\S7BlobDemo\BlobDemo.s7p**.

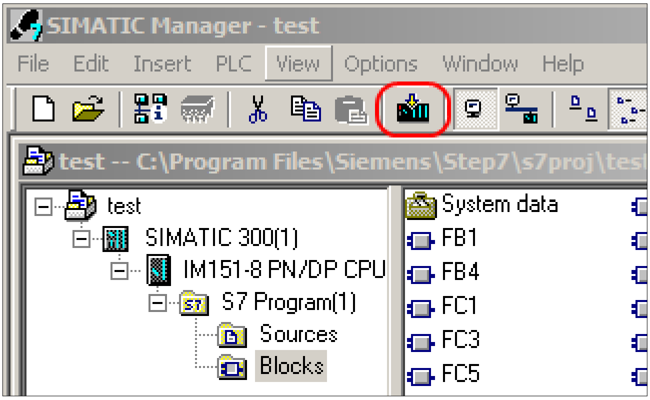



Split both example and current projects in the dialog as shown.

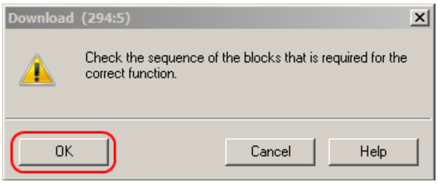


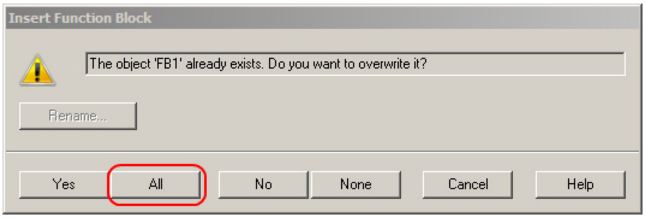

Select the objects minus the **System** data object and copy with a right-click and Copy or CTRL + C. Paste into the current project with a right-click > paste or CTRL + V. When prompted that **OB1** already exists, click the **Yes** button to overwrite it. Now the example program is imported to the current project.

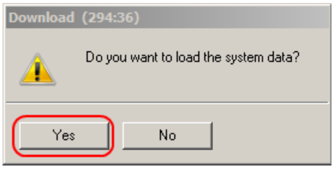
Select the **Blocks** icon on the left pane. Then click the download button on the top ribbon. This will download the new copied functions and system data to the PLC. Now it's time to update the new addressing from the hardware installation prior. Double-click the **OB1** block to open the **LAD/STL/FBD** editor. OB1 is the main routine of the PLC program.

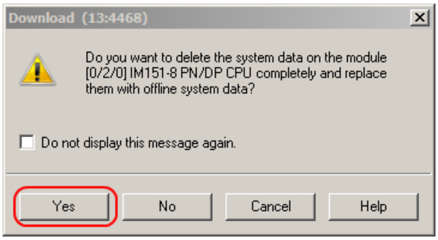


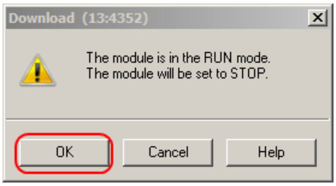
Scroll down to **Network 1**. This is where the data is mapped from the camera to the local data structure. **FC1** is a function that pushes the input data from the camera to the program structure. Click the numbers to match the address on the hardware as shown. Click the following buttons on the popup dialogs:

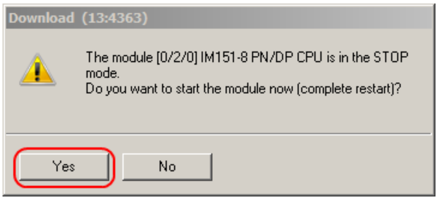


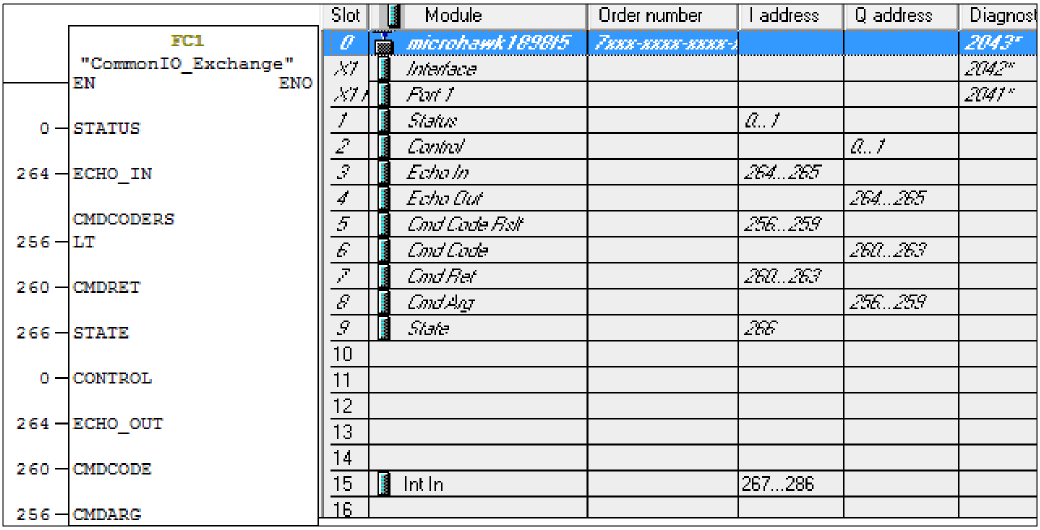



**Note:** This is the address mapping view of the MicroHAWK module which is derived from the hardware config view.




| Slot | Module | Order number | I address | Q address | Diagnost |
|---|---|---|---|---|---|
| 0 | microhawk189815 | 7xxx-xxxx-xxxx-x | | | 2043* |
| X1 | Interface | | | | 2042* |
| X1 | Port 1 | | | | 2041* |
| 1 | Status | | 0...1 | | |
| 2 | Control | | | 0...1 | |
| 3 | Echo In | | 264...265 | | |
| 4 | Echo Out | | | 264...265 | |
| 5 | Cmd Code Rslt | | 256...259 | | |
| 6 | Cmd Code | | | 260...263 | |
| 7 | Cmd Ret | | 260...263 | | |
| 8 | Cmd Arg | | | 256...259 | |
| 9 | State | | 266 | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | Int In | | 267...286 | | |
| 16 | | | | | |

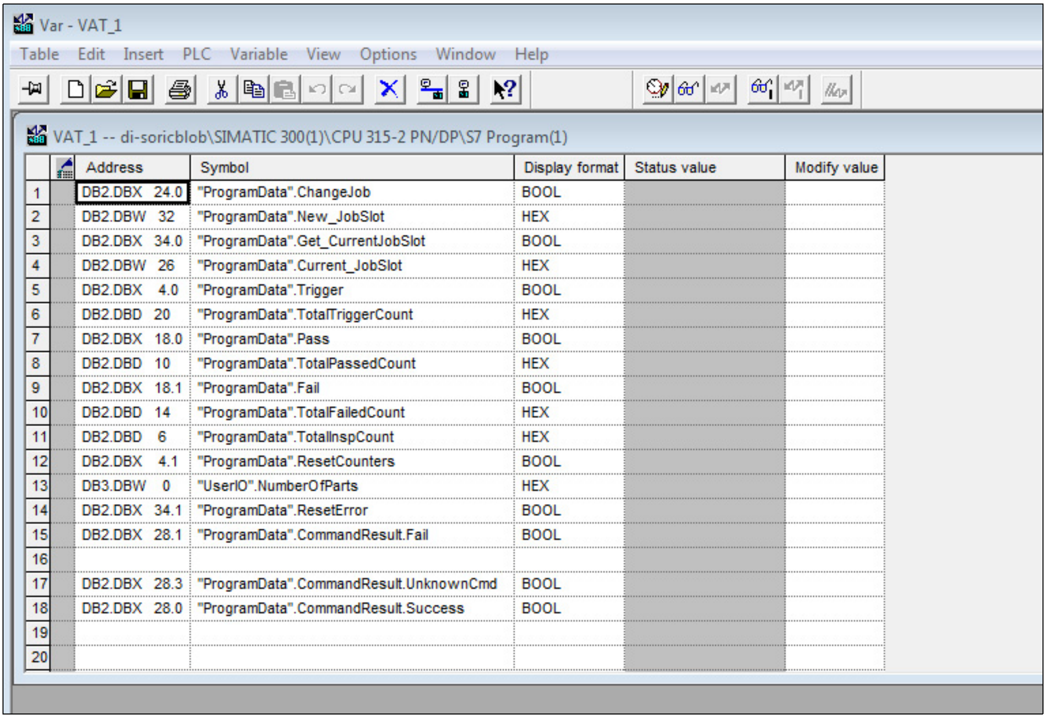
The addresses on the right must be copied to **Network 1** to match. Ex: HW config shows Status as address 0..1. Copy **0** into Network 1, **STATUS** value.

Repeat the same steps for **Network 6** to update the output data mapping. Keep in mind that all input addresses are under the (**I address**) column and the output addresses are under the (**Q address**) column. Save to the PC and download to the PLC.

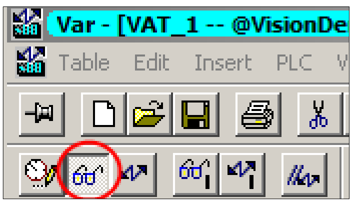
Do a master reset and set the PLC to RUN mode. Make sure all LEDs indicate **good**.

## Running the Demo

In the **SIMATIC Manager** dialog, double-click the **VAT_1** icon. This will open the variable table dialog for the data type demo. Maximize the internal dialog to increase the viewing area.




To establish a live connection to the PLC, click the **Monitor Variable** button on the ribbon. This will update data from the PLC to the dialog. The top title bar will go blue and the bottom status will show run with a green progress bar.

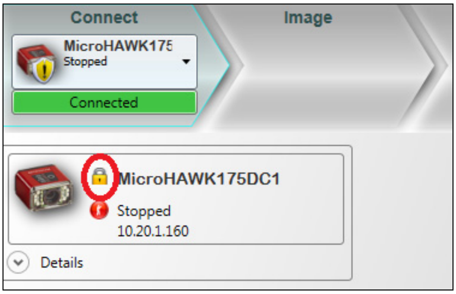


The following operations can be tried here:

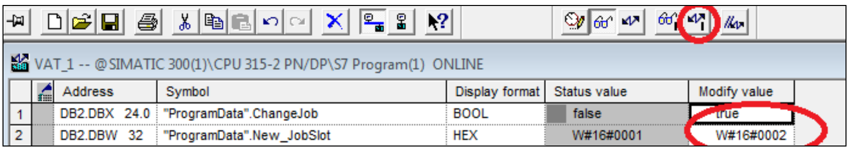
| | |
|---|---|
| Change Job | Switch jobs from one slot to another |
| Get Current Job Slot | Displays the slot of the currently running job |
| Trigger an Inspection | Causes the camera to cycle through its inspection. |
| Reset Counters | Resets the TotalTriggerCount, TotalPassedCount, TotalFailedCount, and TotalInspCount to 0, and also sends a command to the camera to reset its internal counters. |
| Reset an Error condition | If an error occurs during run time, this will reset it, so demo may be continued. |

**Change Jobs + Get Current Job Slot**

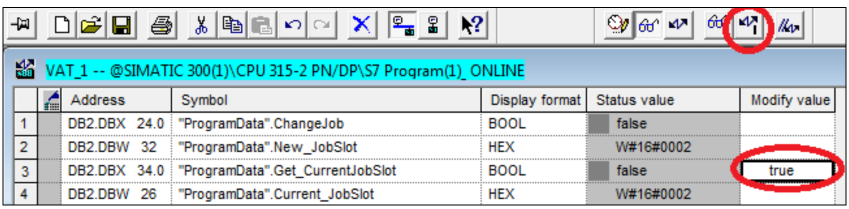
Be sure to follow instructions in the AutoVISION Setup section to store a job in a slot in the camera memory. To properly execute this test, load another job into **slot 2** of the camera. For example, the Circle_LocatorDemo job. You must release control of the device by clicking on the **Connect** tab and change the lock icon to **closed** as in the following image, then click the **Run** tab again for job change to work properly.




Using the VAT_1 elements **ChangeJob** and **New_JobSlot**. Click the **Modify value** column next to New_JobSlot, type 2, and type **Enter**. Click on the Modify Value column to the right of ChangeJob and enter TRUE. Click on the icon indicated at the top.



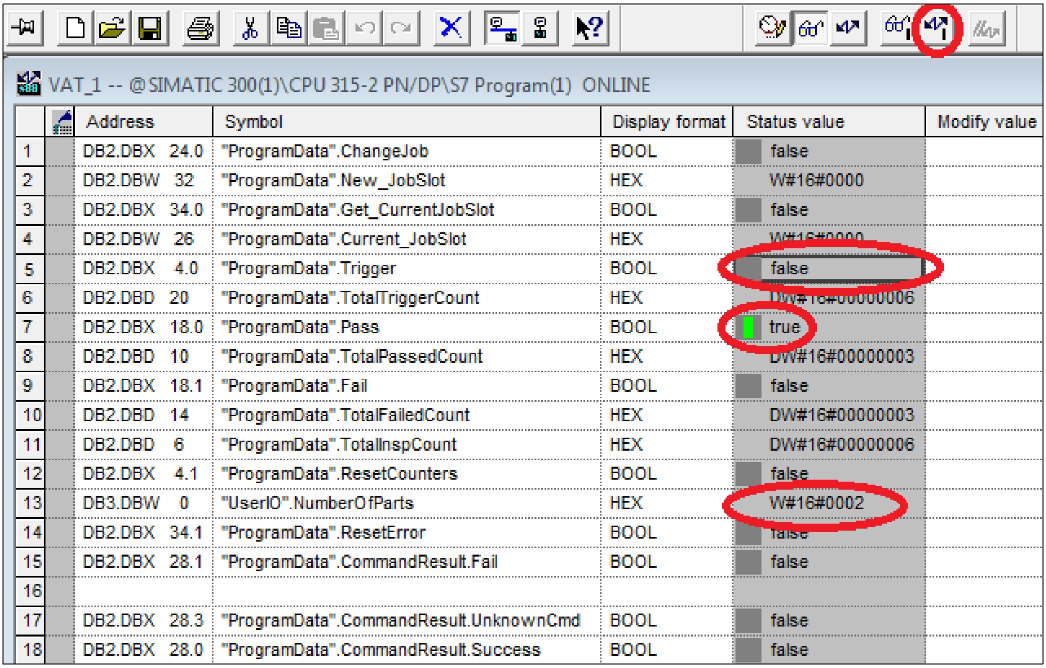

The current job will be changed to the job in **slot 2**. To verify this, clear all changes made to the **Modify value** column, and click Modify value to the right of **Get_CurrentJobSlot**. Enter **TRUE**, and click on the indicated icon again.




You will see the current job listed to the right of **Current_JobSlot**. You may repeat this procedure changing the slot back to **1** and verify the current job slot is now **1**.

## Trigger an Inspection and Reset Counters

Be sure the current job selected is **slot 1**. Clear any changes made to the **Modify value** column. If you want to watch the inspection, be sure AutoVISION is running, and connected to the device. Be sure the job is running. If you choose not to run AutoVISION, you may look at the LEDs on the device to confirm your triggers. Change the value to the right of Trigger to **TRUE**, and click the indicated icon.



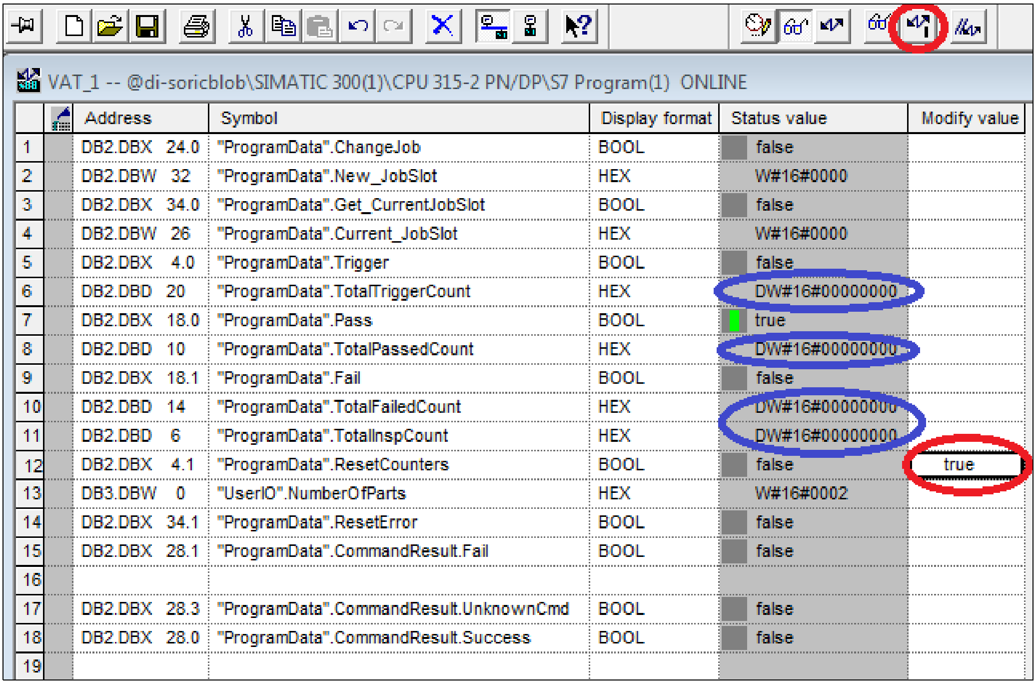
VAT_1 -- @SIMATIC 300(1)\CPU 315-2 PN/DP\S7 Program(1) ONLINE

| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | DB2.DBX 24.0 | "ProgramData".ChangeJob | BOOL | false | |
| 2 | DB2.DBW 32 | "ProgramData".New_JobSlot | HEX | W#16#0000 | |
| 3 | DB2.DBX 34.0 | "ProgramData".Get_CurrentJobSlot | BOOL | false | |
| 4 | DB2.DBW 26 | "ProgramData".Current_JobSlot | HEX | W#16#0000 | |
| 5 | DB2.DBX 4.0 | "ProgramData".Trigger | BOOL | false | |
| 6 | DB2.DBD 20 | "ProgramData".TotalTriggerCount | HEX | DW#16#00000006 | |
| 7 | DB2.DBX 18.0 | "ProgramData".Pass | BOOL | true | |
| 8 | DB2.DBD 10 | "ProgramData".TotalPassedCount | HEX | DW#16#00000003 | |
| 9 | DB2.DBX 18.1 | "ProgramData".Fail | BOOL | false | |
| 10 | DB2.DBD 14 | "ProgramData".TotalFailedCount | HEX | DW#16#00000003 | |
| 11 | DB2.DBD 6 | "ProgramData".TotalInspCount | HEX | DW#16#00000006 | |
| 12 | DB2.DBX 4.1 | "ProgramData".ResetCounters | BOOL | false | |
| 13 | DB3.DBW 0 | "UserIO".NumberOfParts | HEX | W#16#0002 | |
| 14 | DB2.DBX 34.1 | "ProgramData".ResetError | BOOL | false | |
| 15 | DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | false | |
| 16 | | | | | |
| 17 | DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | false | |
| 18 | DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | false | |

You will see the **NumberOfParts** change to either **2** or **5**. The pass and fail bools will indicate whether the inspection has passed or failed. Also, the appropriate counters will have incremented. Each press of the icon will cause an inspection which will bounce from one image to the next. One will pass and the next will fail.

## Reset Counters

To reset the counters, clear all settings in the Modify value column, and change the one next to **ResetCounters** to **TRUE**, and click the icon again. You will see all the counters in blue reset to **0**.



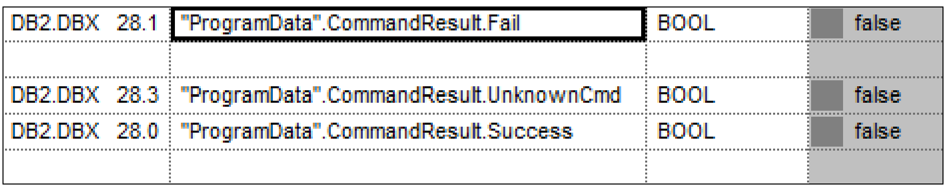
VAT_1 -- @di-soricblob\SIMATIC 300(1)\CPU 315-2 PN/DP\S7 Program(1) ONLINE

| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | DB2.DBX 24.0 | "ProgramData".ChangeJob | BOOL | false | |
| 2 | DB2.DBW 32 | "ProgramData".New_JobSlot | HEX | W#16#0000 | |
| 3 | DB2.DBX 34.0 | "ProgramData".Get_CurrentJobSlot | BOOL | false | |
| 4 | DB2.DBW 26 | "ProgramData".Current_JobSlot | HEX | W#16#0000 | |
| 5 | DB2.DBX 4.0 | "ProgramData".Trigger | BOOL | false | |
| 6 | DB2.DBD 20 | "ProgramData".TotalTriggerCount | HEX | DW#16#00000000 | |
| 7 | DB2.DBX 18.0 | "ProgramData".Pass | BOOL | true | |
| 8 | DB2.DBD 10 | "ProgramData".TotalPassedCount | HEX | DW#16#00000000 | |
| 9 | DB2.DBX 18.1 | "ProgramData".Fail | BOOL | false | |
| 10 | DB2.DBD 14 | "ProgramData".TotalFailedCount | HEX | DW#16#00000000 | |
| 11 | DB2.DBD 6 | "ProgramData".TotalInspCount | HEX | DW#16#00000000 | |
| 12 | DB2.DBX 4.1 | "ProgramData".ResetCounters | BOOL | false | true |
| 13 | DB3.DBW 0 | "UserIO".NumberOfParts | HEX | W#16#0002 | |
| 14 | DB2.DBX 34.1 | "ProgramData".ResetError | BOOL | false | |
| 15 | DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | false | |
| 16 | | | | | |
| 17 | DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | false | |
| 18 | DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | false | |
| 19 | | | | | |

If you clear this again and set the Trigger to **TRUE**, and click the lightning bolt button again, you will notice that the counts start at **1** on both the PLC counters and AutoVISION counters.

## Reset Error

If at any time there is an error indicated by a TRUE in any of the values below, you can clear the error by entering a TRUE to the right of **ResetError** and clicking the icon. Reset it to **FALSE** and click the button again. Any errors in the values below should be cleared.



| | | | |
|---|---|---|---|
| DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | false |
| | | | |
| DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | false |
| DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | false |
| | | | |

12

CHAPTER 12

# PROFINET I/O Circle Locate Demo Using Step 7 and MicroHAWK

This section describes how to use Omron Microscan demo PLC code with a vision job and camera target.

The PROFINET I/O demo files can be found where AutoVISION is installed, in the folder: **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\Step7_Demos\Circle_LocatorDemo**. Open **Circle_LocatorDemo.avp** and download it to the camera.

**Notes:**

- The camera communications protocol must be enabled for PROFINET I/O before it can be used in this environment. Refer to **Chapter 1, Enabling Industrial Protocols**, for information about enabling and switching communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate PROFINET I/O communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*.
  **Important:** PROFINET I/O allows the IP address and subnet mask of the camera to be assigned automatically by the PLC. In order to allow this, the MicroHAWK may not have an assigned IP address on reboot until the PLC is set to Run mode. During this time, the MicroHAWK will not be visible on the network for AutoVISION or Visionscape FrontRunner.

## Overview

In this demonstration, you will learn how to load a saved job into the camera, establish connectivity via **PROFINET I/O** to a **Siemens S7 PLC**, and run some example programs that interface with the camera.

While evaluating PROFINET I/O capabilities, you will:

- **Begin with AutoVISION.**

Open a sample demo vision job in AutoVISION and use the Try Out feature to learn what to expect from the camera before it is connected to the PLC.

- **Prepare the PLC.**

Integrate the camera into the PLC environment with STEP 7 software and the GSD file.

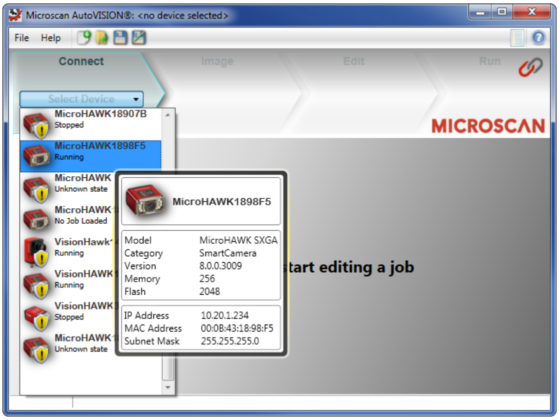
- **Run the camera.**

Trigger the camera while it is online with the PLC and observe changes in the Inspection status as the PLC reconfigures the vision job’s parameters.

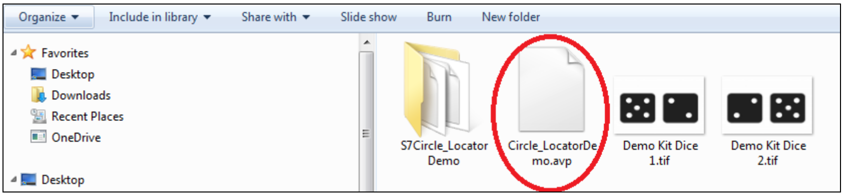
# AutoVISION Setup

Prior to starting AutoVISION, make sure the camera is either connected to the PLC or both PLC and camera are on the same physical network. Ensure that the PC, PLC, and camera have the same network class and corresponding subnet addresses.

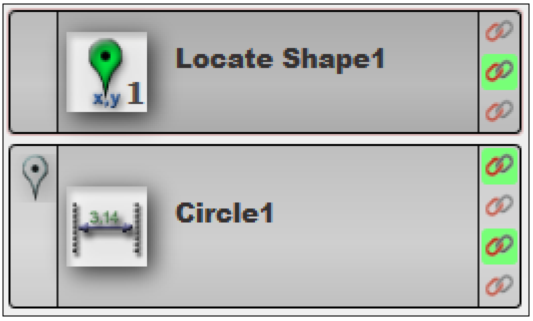
Open AutoVISION and select the camera.



From the Image view, click the **Load a Job** button. Then navigate to **C:\Microscan\Vscape\Tutorials And Samples\MicroHAWK\PROFINET demo\Step7_Demos\Circle_LocatorDemo**. Select **Circle_LocatorDemo.avp**.




The demo job will include two tools: **Locate Shape** and **Circle**.

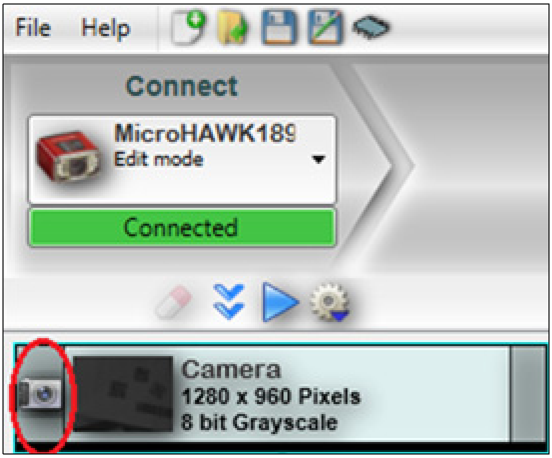



The PROFINET I/O structure is shown here:

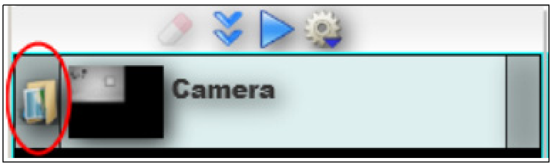
| Tool Result | PLC IN |
|---|---|
| Instance Point1 | "UserData".Instance1Point1(%DB2.DBD2) |
| Instance Point2 | "UserData".Instance1Point2(%DB2.DBD6) |
| Instance Point3 | "UserData".InstancePoint3(%DB2.DBD10) |
| Status | "UserData".Status[0](%DB2.DBX0.0) |
| RadiumDatum | "UserData".RadiusDatum(%DB2.DBD14) |

This data is transferred cyclically between the camera and PLC.

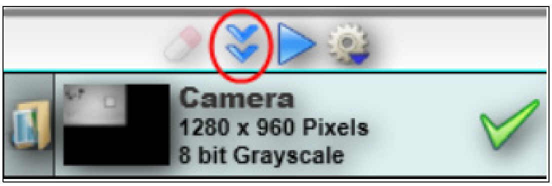
Once the job has loaded, the next step is to link the pre-saved images on the local PC. On the **Camera** button, click the far left icon to select and load an image.



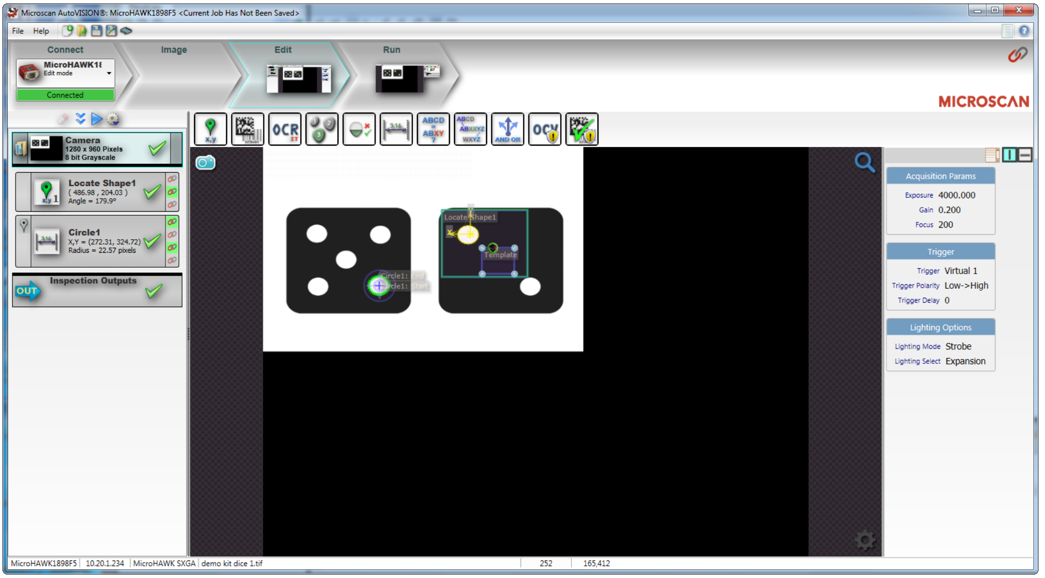

A file browser will open. Then navigate to the same folder where the demo job was loaded PROFINET I/O Demo. If the images are located, the icon will change from a camera to a folder.

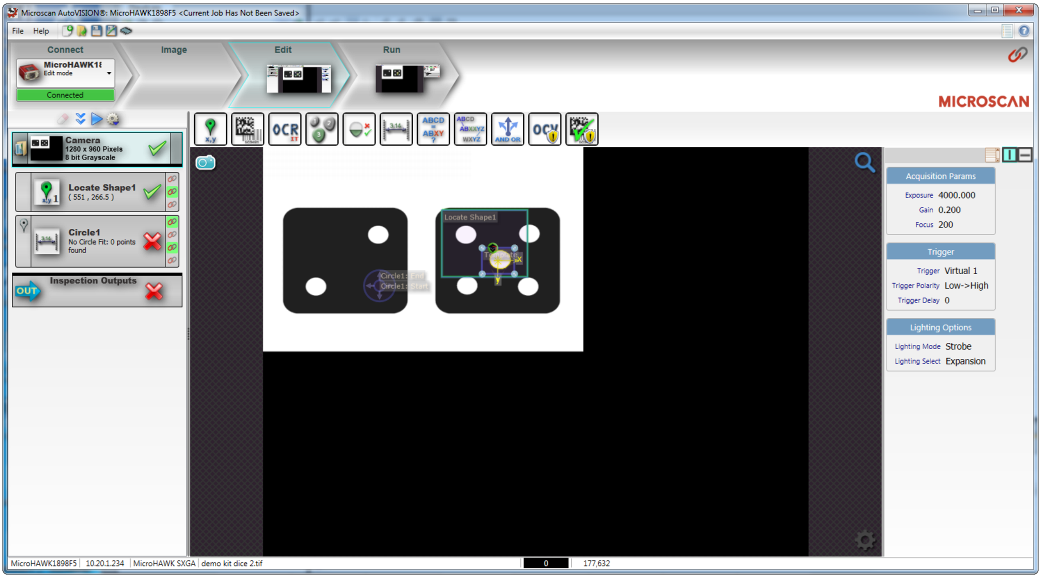



By clicking the **Try Job Once** icon, the application will cycle through the entire job with the loaded image.



There are two images on file. One will pass the inspection and one will fail.



Now click the **Run** button on the top ribbon. This will download the job to the camera. At this point the job is ready to run and can be tested. However, in order to run the job change demo this job needs to be loaded into **Slot 1**. Click back to the **Edit** view from the top ribbon bar.

Click the slot icon and select Slot 1 or **New Slot** if no slots are currently in use.

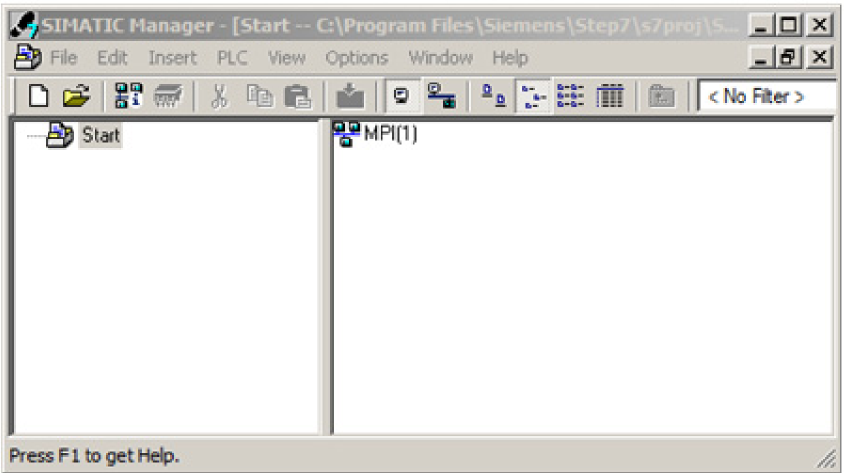

Now the job and images will be saved to the flash memory of the camera.

## STEP 7 Setup

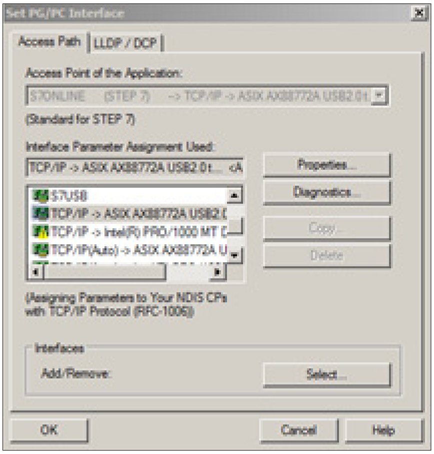
This section was created using Siemens STEP 7 software version 5.5 + SP2 and an ET200S PLC, catalog number 6ES7 151-8AB01-0AB0, CPU Version 3.2. It was tested with a 315-2 PN/DP PLC, catalog number 6ES7-315-2EH13-0AB0, CPU Version 2.6.

First launch the SIMATIC Manager from the desktop.

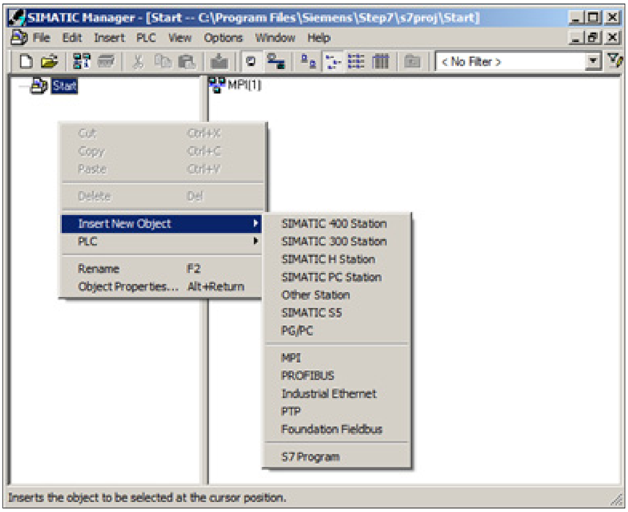

Create a new project from the menu item **File > New**. Select the project location on disk then enter the name and click the **OK** button. In the example below, the name is **Start**. Once the project is first created, you will see the dialog. This dialog is the main entry point into the PLC program and hardware settings.



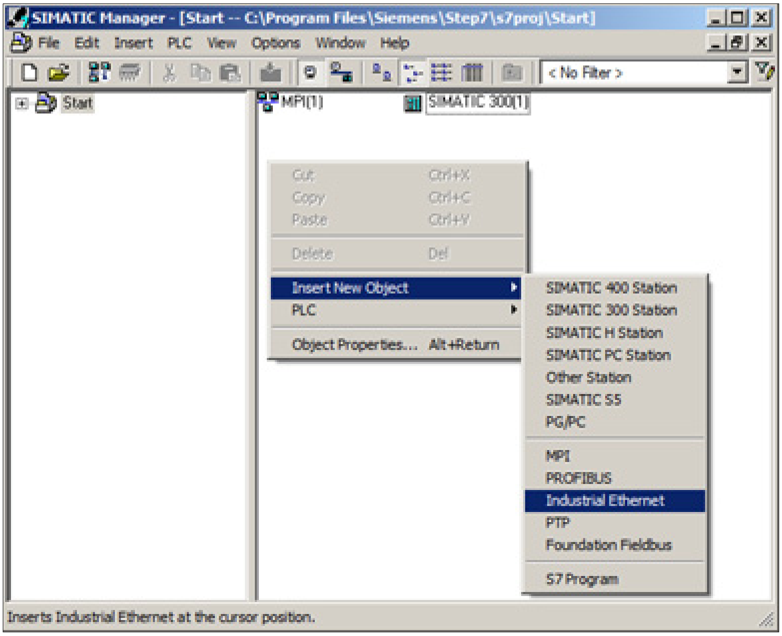
The PC may have multiple network cards so it’s imperative to map the correct one to the SIMATIC software. From the menu, select Options > Set PG/PC Interface. This will open the Set PG/PC Interface dialog and list the available network cards. On the Access Path tab select the NIC card with (TCP/IP >) in the name.



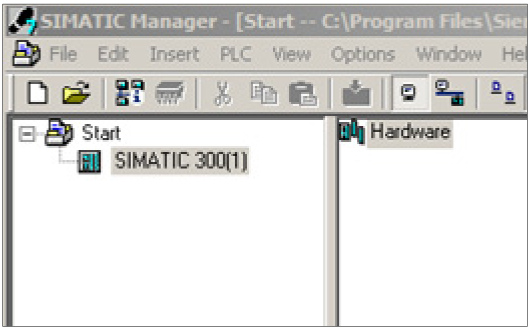

In the SIMATIC Manager dialog, right-click and select Insert New Object. This is where you will select the base station. For example, when configuring an ET200-s, select SIMATIC 300 Station since it’s based on the 300 series CPU.



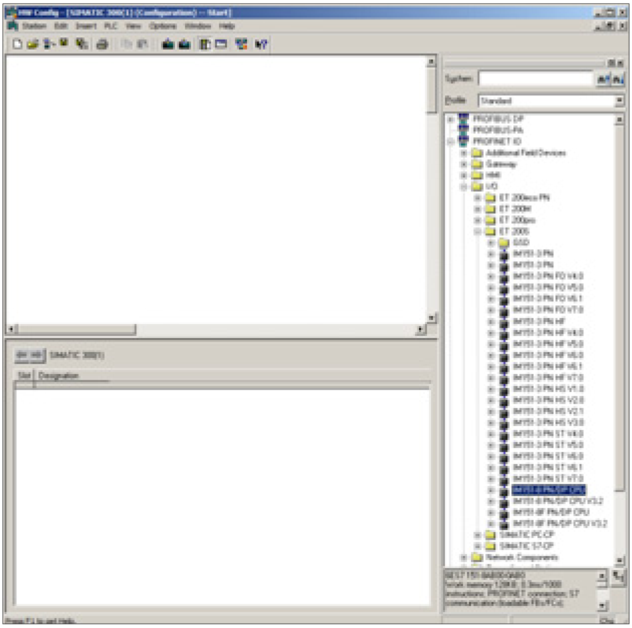
After the station has been added, right-click and add **Industrial Ethernet**.



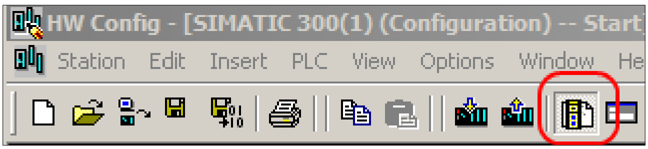

Expand the Start tree node on the left pane and click the station. On the right pane you will see a Hardware icon.



Double-click the **Hardware** icon to launch the **HW Config** dialog.



Make sure the **Catalog** is selected in the ribbon bar on the top. This will add a tree view on the right pane with all the available hardware devices.

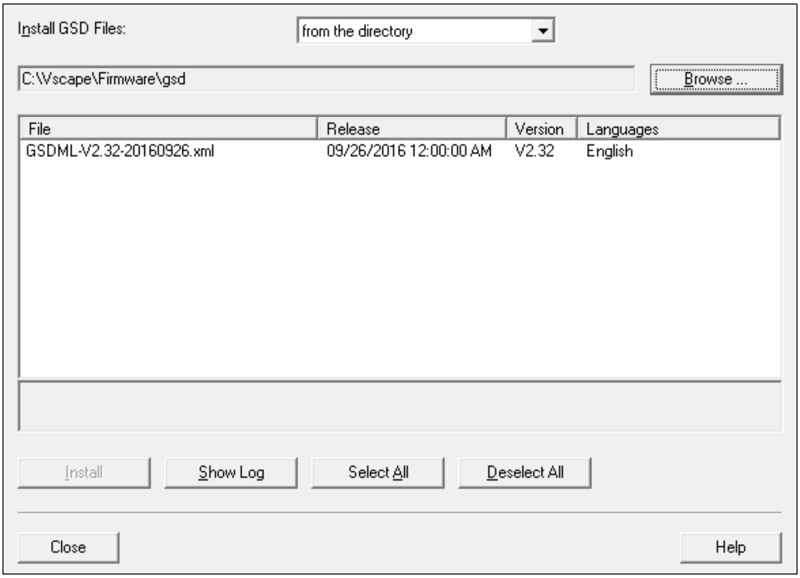



If the latest GSDML file hasn't been imported, follow the next steps to import.

## GSDML

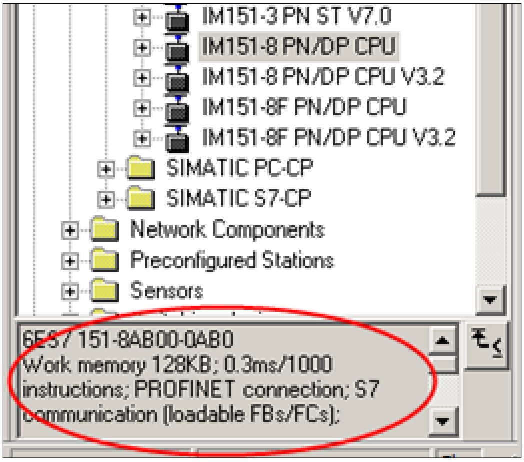
- Install the GSD file from the menu item **Options > Install GSD File**.
- Click the Browse button to locate the GSDML file **\Microscan\Vscape\Firmware\gsd\MicroHAWK**.

- The center pane will list the available GSDML files.




- Select the file and click the **Install** button.
- When finished, close the dialog.
- The camera should now be present in the PROFINET I/O section under **Additional Field Devices > General**.

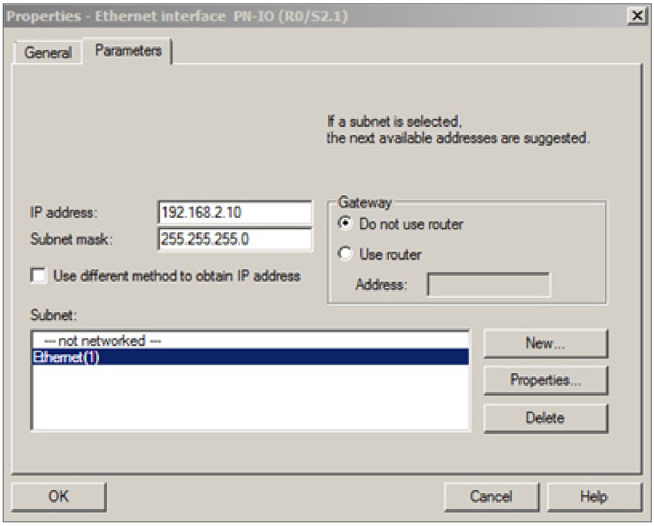
Add the CPU from the catalog view by drag and drop or double-click. Make sure the catalog number and version matches the PLC exactly. The catalog number will be displayed on the bottom of the view.




Some CPUs are modules that will require a generic rack to be added prior. If your PLC requires a rack, you will be prompted to add the rack prior to being allowed to insert the CPU module.

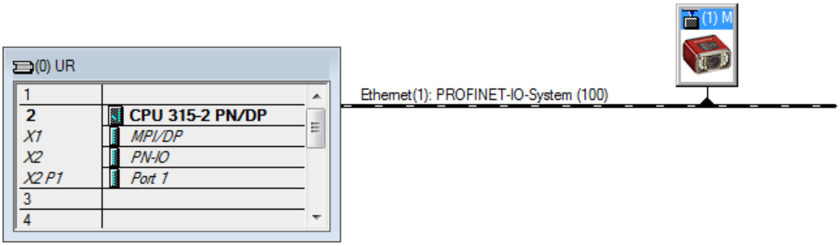
After the CPU is added, a popup dialog will prompt the Properties relating to the IP information.

Select **Ethernet(1)** on the bottom list box and enter the correct IP address of the CPU.

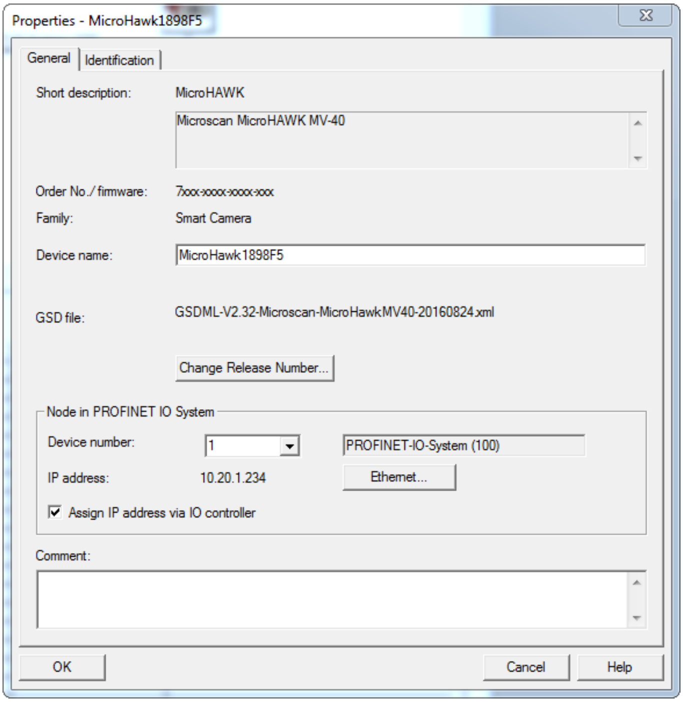


Once the CPU has been added, add any additional expansion modules, if installed. Click the CPU on the left pane and the bottom pane will list the available ports and slots.

Add the camera in the location (**PROFINET I/O > Additional Field Devices > General > Microscan**). The camera can be added by drag and drop to **Ethernet(1)**: **PROFINET-I/O System (100)** or by selecting the **Ethernet(1)** line first and double-clicking the camera. Once the camera has been added the icon will appear on the configuration dialog.

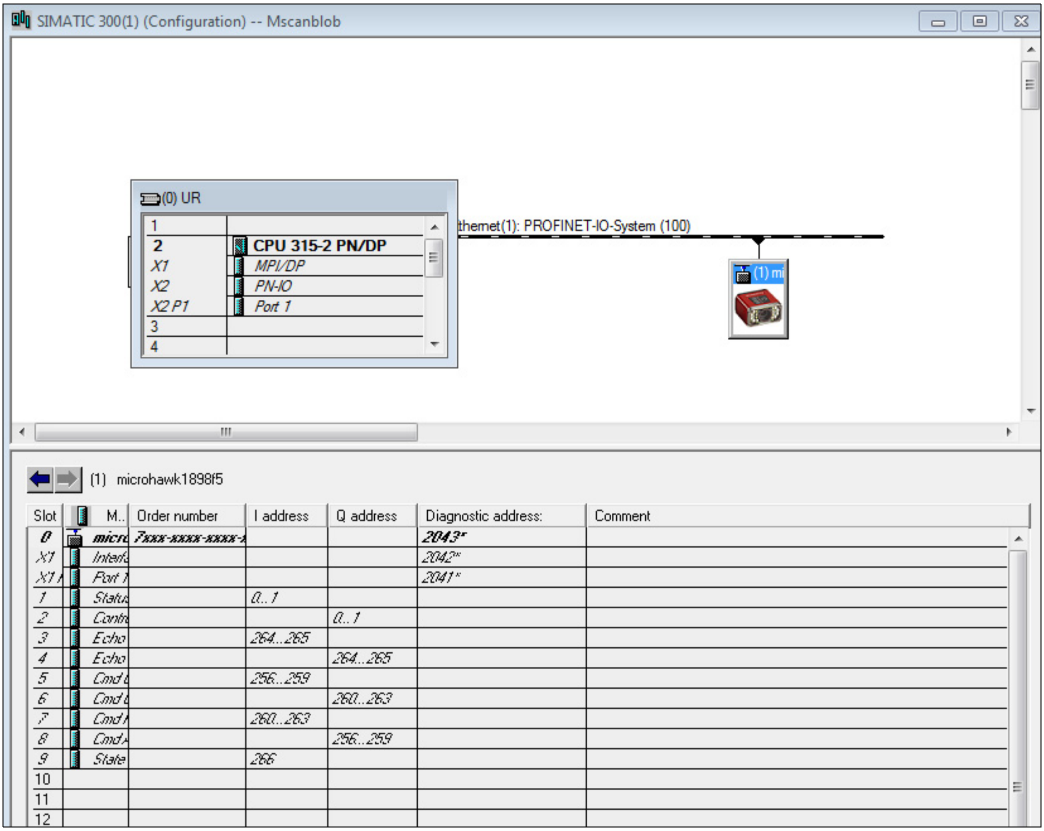



Once the camera has been added, double-click the icon to open the properties dialog. Under the device name, enter the existing name of the camera or a unique name.

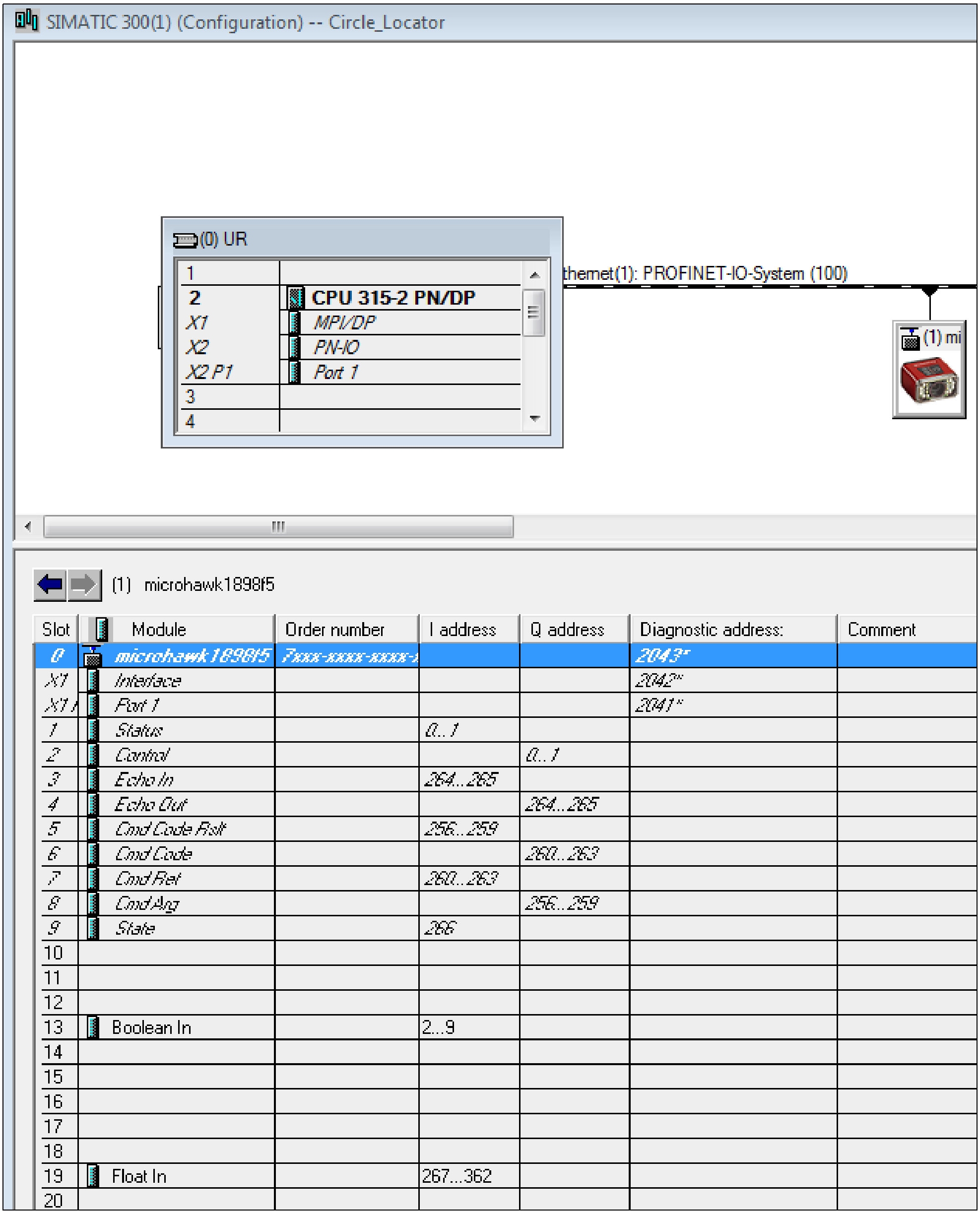



If a unique name is used, the device has to be manually updated. View Updating camera name section.

Click the camera icon and the data slot address mapping will be displayed below. Take note of the address values since they will be needed in the demo application. Since there is an infinite combination of modules and slot configurations, the addressing is unique to every setup.

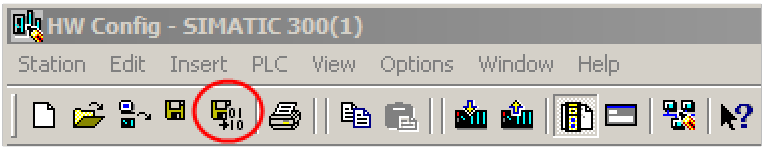


This demo uses the Boolean **In** and **Float In** for data. To the right under **PROFINET I/O MicroHAWK**, click on **Boolean In**. This will highlight its appropriate slot. Drag and drop this to the highlighted slot. Repeat this for the Float In. The table should now appear as shown below.

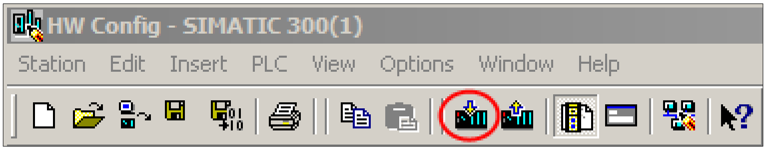



| Slot | Module | Order number | I address | Q address | Diagnostic address: | Comment |
|---|---|---|---|---|---|---|
| 0 | microhawk1898f5 | 7xxx-xxxx-xxxx-) | | | 2043* | |
| X1 | Interface | | | | 2042* | |
| X1 / | Port 1 | | | | 2041* | |
| 1 | Status | | 0...1 | | | |
| 2 | Control | | | 0...1 | | |
| 3 | Echo In | | 264...265 | | | |
| 4 | Echo Out | | | 264...265 | | |
| 5 | Cmd Code Rslt | | 256...259 | | | |
| 6 | Cmd Code | | | 260...263 | | |
| 7 | Cmd Ret | | 260...263 | | | |
| 8 | Cmd Arg | | | 256...259 | | |
| 9 | State | | 266 | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | Boolean In | | 2...9 | | | |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | Float In | | 267...362 | | | |
| 20 | | | | | | |

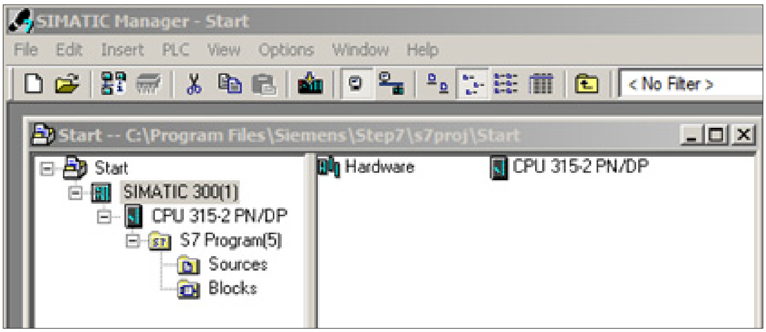
Once the hardware configuration has been completed, it's time to compile and download. Click the compile and save icon on the ribbon. If there are any configuration conflicts, the application will prompt a warning at this point.



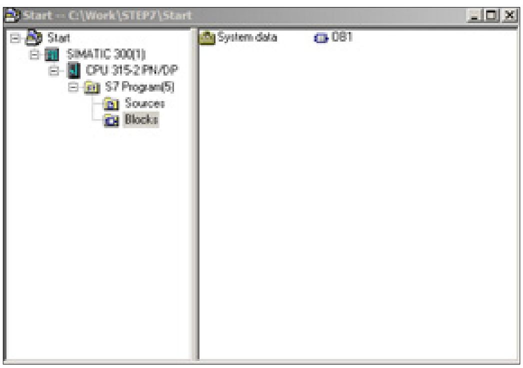

Once compiled, click the download icon on the ribbon to send the information to the PLC.




At this point close or minimize the **HW Config** dialog and re-visit the **SIMATIC Manager** dialog. The CPU should be added next to the Hardware icon and in the tree view in the left pane. Expand the CPU tree item and remaining child items below it.



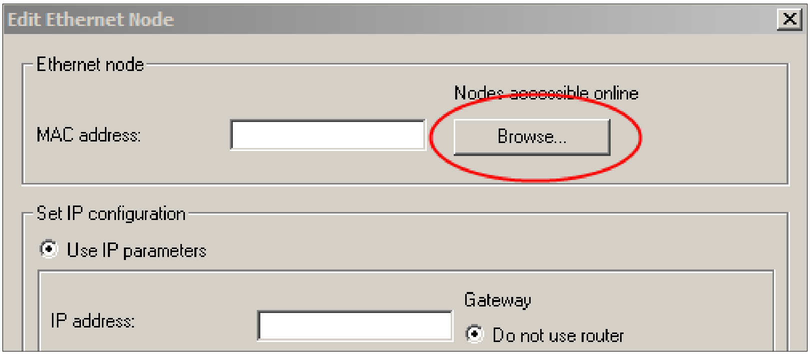

Click the Blocks node to view the program objects.



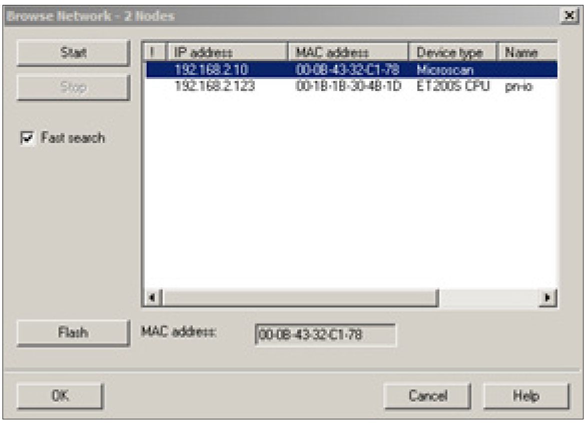
## Updating Camera Name

From the HW Config dialog open **Edit Ethernet Node** from the menu item **PLC > Ethernet > Edit Ethernet Node**. From the SIMATIC Manager dialog open Edit Ethernet Node from the menu item **PLC > Edit Ethernet Node**.

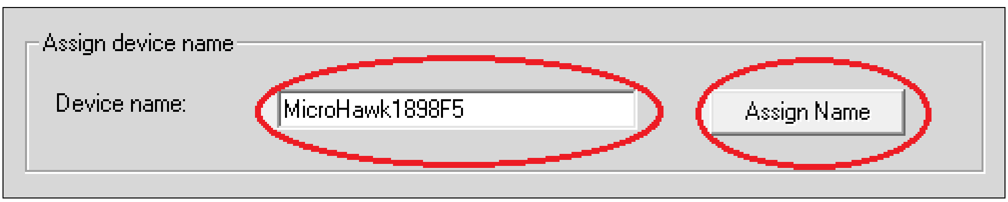
Click the **Browse** button to open the node selection dialog.




Select the camera and click the **OK** button.

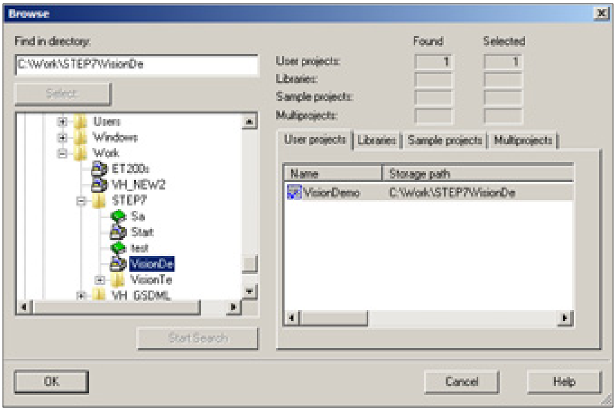



Edit the **Device Name**, if necessary, to match the name entered previously in the MicroHAWK properties dialog, and click the **Assign Name** button.

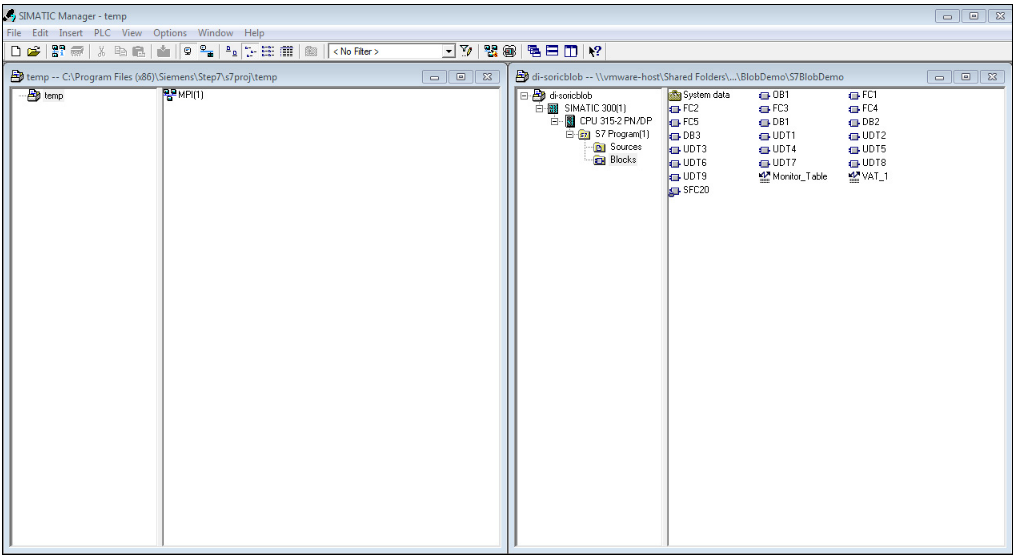


## Importing Example Program

From the SIMATIC Manager, open the example program from the menu **File > Open**. Click the **Browse** button to locate the (BlobDemo) program. The BlobDemo is located in **\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\Step7_Demos\Circle_LocatorDemo\S7Circle_LocatorDemo\Circle_LocatorDemo.s7p**.

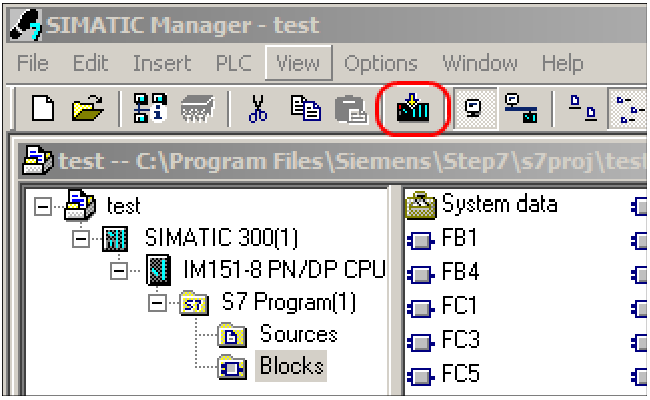



Split both example and current projects in the dialog as shown.

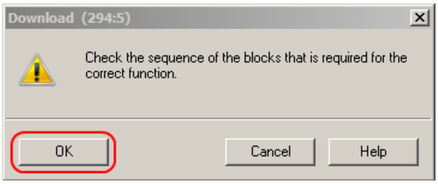


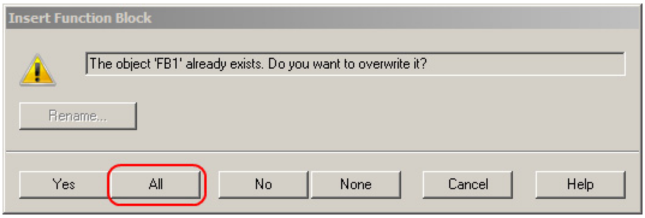

Select the objects minus the **System** data object and copy with a right-click and Copy or CTRL + C. Paste into the current project with a right-click > paste or CTRL + V. When prompted that **OB1** already exists, click the **Yes** button to overwrite it. Now the example program is imported to the current project.

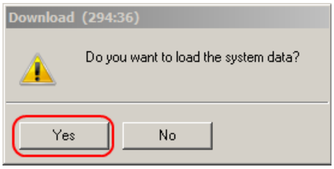
Select the **Blocks** icon on the left pane. Then click the download button on the top ribbon. This will download the new copied functions and system data to the PLC. Now it's time to update the new addressing from the hardware installation prior. Double-click the **OB1** block to open the **LAD/STL/FBD** editor. OB1 is the main routine of the PLC program.

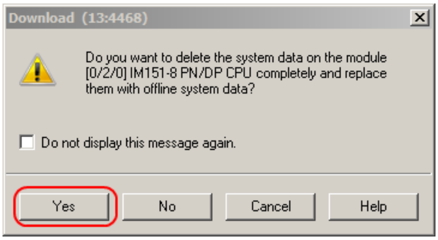


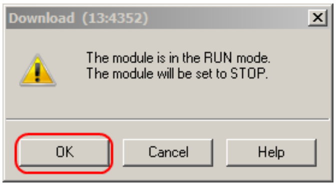
Scroll down to **Network 1**. This is where the data is mapped from the camera to the local data structure. **FC1** is a function that pushes the input data from the camera to the program structure. Click the numbers to match the address on the hardware as shown. Click the following buttons on the popup dialogs:

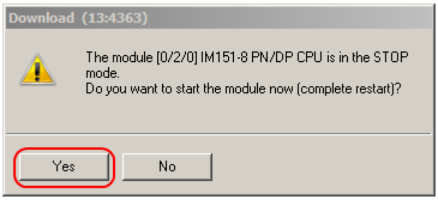


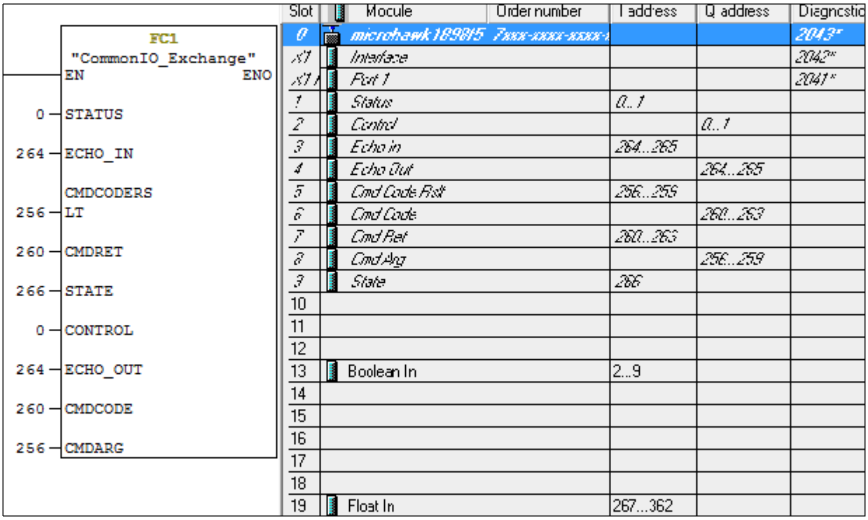



**Note:** This is the address mapping view of the MicroHAWK module which is derived from the hardware config view.




| Slot | Module | Order number | I address | Q address | Diagnostic |
|---|---|---|---|---|---|
| 0 | microhawk 189815 | 7xxx-xxxx-xxxx-x | | | 2043* |
| X1 | Interface | | | | 2042* |
| X1 P | Port 1 | | | | 2041* |
| 1 | Status | | 0..1 | | |
| 2 | Control | | | 0..1 | |
| 3 | Echo In | | 264...265 | | |
| 4 | Echo Out | | | 264...265 | |
| 5 | Cmd Code Rslt | | 256...259 | | |
| 6 | Cmd Code | | | 260...263 | |
| 7 | Cmd Ret | | 260...263 | | |
| 8 | Cmd Arg | | | 256...259 | |
| 9 | State | | 266 | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | Boolean In | | 2...9 | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | Float In | | 267...362 | | |

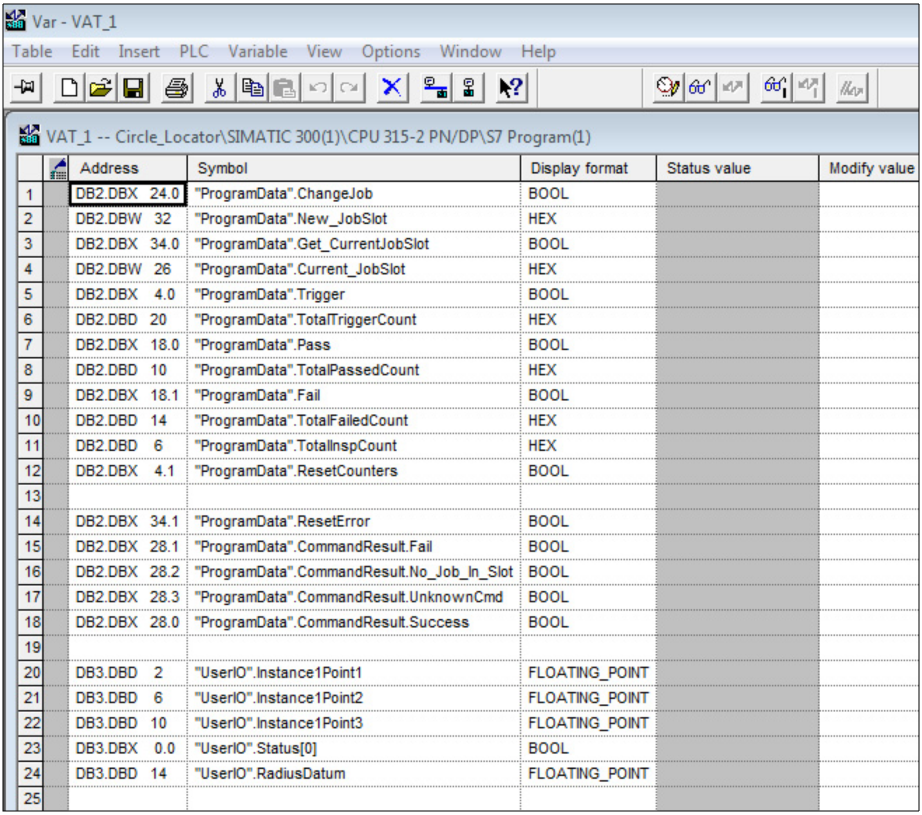
The addresses on the right must be copied to **Network 1** to match. Ex: HW config shows Status as address 0..1. Copy **0** into Network 1, **STATUS** value.

Repeat the same steps for **Network 6** to update the output data mapping. Keep in mind that all input addresses are under the (**I address**) column and the output addresses are under the (**Q address**) column. Save to the PC and download to the PLC.

Do a master reset and set the PLC to RUN mode. Make sure all LEDs indicate **good**.

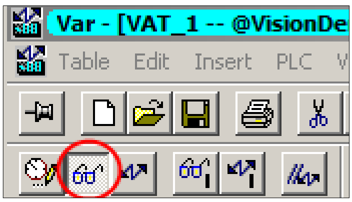
## Running the Demo

In the **SIMATIC Manager** dialog, double-click the **VAT_1** icon. This will open the variable table dialog for the data type demo. Maximize the internal dialog to increase the viewing area.




| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | DB2.DBX 24.0 | "ProgramData".ChangeJob | BOOL | | |
| 2 | DB2.DBW 32 | "ProgramData".New_JobSlot | HEX | | |
| 3 | DB2.DBX 34.0 | "ProgramData".Get_CurrentJobSlot | BOOL | | |
| 4 | DB2.DBW 26 | "ProgramData".Current_JobSlot | HEX | | |
| 5 | DB2.DBX 4.0 | "ProgramData".Trigger | BOOL | | |
| 6 | DB2.DBD 20 | "ProgramData".TotalTriggerCount | HEX | | |
| 7 | DB2.DBX 18.0 | "ProgramData".Pass | BOOL | | |
| 8 | DB2.DBD 10 | "ProgramData".TotalPassedCount | HEX | | |
| 9 | DB2.DBX 18.1 | "ProgramData".Fail | BOOL | | |
| 10 | DB2.DBD 14 | "ProgramData".TotalFailedCount | HEX | | |
| 11 | DB2.DBD 6 | "ProgramData".TotalInspCount | HEX | | |
| 12 | DB2.DBX 4.1 | "ProgramData".ResetCounters | BOOL | | |
| 13 | | | | | |
| 14 | DB2.DBX 34.1 | "ProgramData".ResetError | BOOL | | |
| 15 | DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | | |
| 16 | DB2.DBX 28.2 | "ProgramData".CommandResult.No_Job_In_Slot | BOOL | | |
| 17 | DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | | |
| 18 | DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | | |
| 19 | | | | | |
| 20 | DB3.DBD 2 | "UserIO".Instance1Point1 | FLOATING_POINT | | |
| 21 | DB3.DBD 6 | "UserIO".Instance1Point2 | FLOATING_POINT | | |
| 22 | DB3.DBD 10 | "UserIO".Instance1Point3 | FLOATING_POINT | | |
| 23 | DB3.DBX 0.0 | "UserIO".Status[0] | BOOL | | |
| 24 | DB3.DBD 14 | "UserIO".RadiusDatum | FLOATING_POINT | | |
| 25 | | | | | |

To establish a live connection to the PLC, click the **Monitor Variable** button on the ribbon. This will update data from the PLC to the dialog. The top title bar will go blue and the bottom status will show run with a green progress bar.

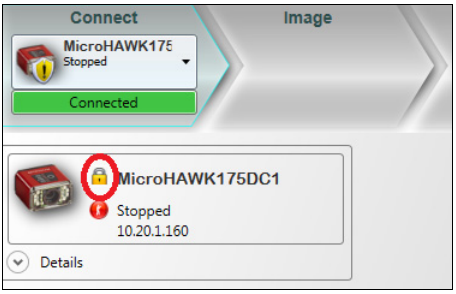


The following operations can be tried here:

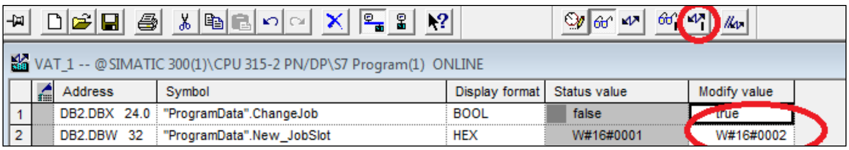
| | |
|---|---|
| Change Job | Switch jobs from one slot to another |
| Get Current Job Slot | Displays the slot of the currently running job |
| Trigger an Inspection | Causes the camera to cycle through its inspection. |
| Reset Counters | Resets the TotalTriggerCount, TotalPassedCount, TotalFailedCount, and TotalInspCount to 0, and also sends a command to the camera to reset its internal counters. |
| Reset an Error condition | If an error occurs during run time, this will reset it, so demo may be continued. |

## Change Jobs + Get Current Job Slot

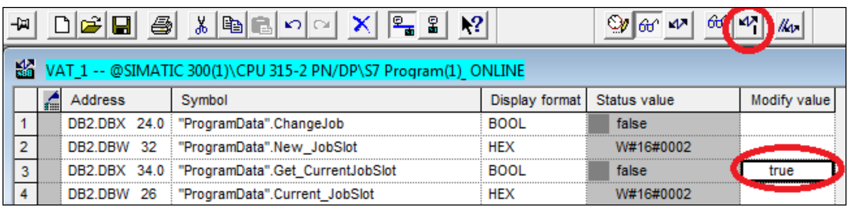
Be sure to follow instructions in the AutoVISION Setup section to store a job in a slot in the camera memory. To properly execute this test, load another job into **slot 2** of the camera. For example, the Circle_LocatorDemo job. You must release control of the device by clicking on the **Connect** tab and change the lock icon to **closed** as in the following image, then click the **Run** tab again for job change to work properly.




Using the VAT_1 elements **ChangeJob** and **New_JobSlot**. Click the **Modify value** column next to New_JobSlot, type 2, and type **Enter**. Click on the Modify Value column to the right of ChangeJob and enter TRUE. Click the icon indicated at the top.



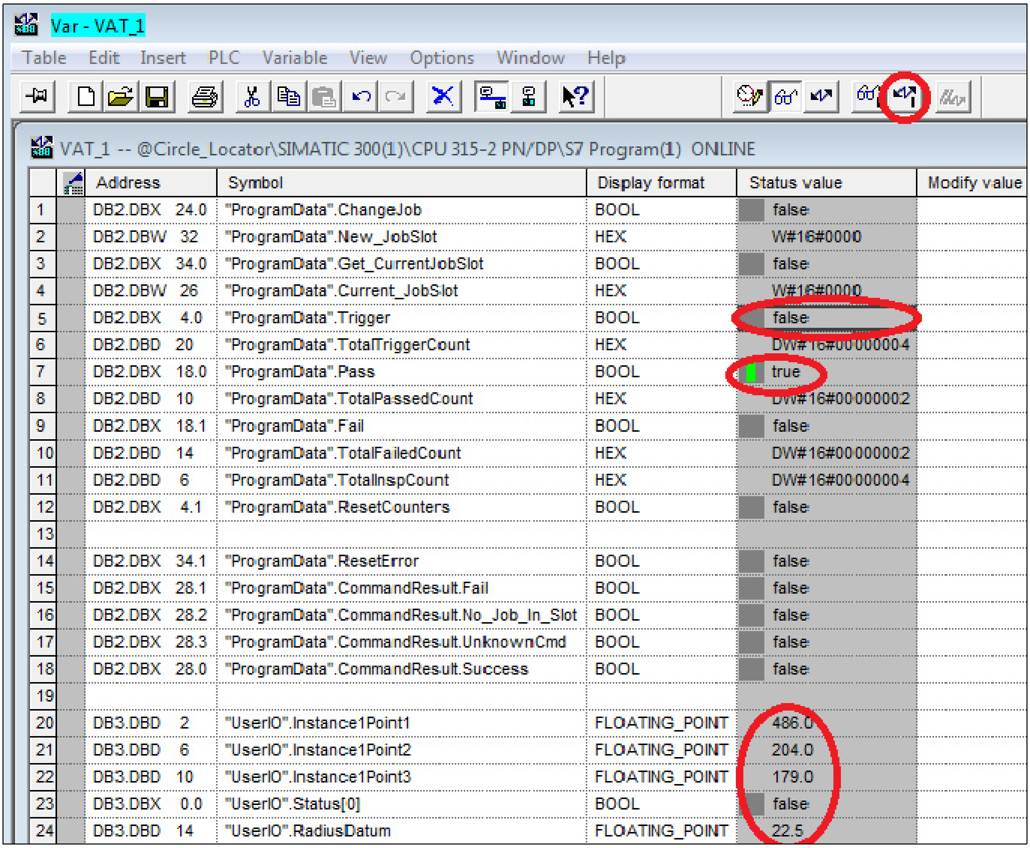

The current job will be changed to the job in **slot 2**. To verify this, clear all changes made to the **Modify value** column, and click Modify value to the right of **Get_CurrentJobSlot**. Enter **TRUE**, and click on the indicated icon again.




You will see the current job listed to the right of **Current_JobSlot**. You may repeat this procedure changing the slot back to **1** and verify the current job slot is now **1**.

## Trigger an Inspection and Reset Counters

Be sure the current job selected is **slot 1**. Clear any changes made to the **Modify value** column. If you want to watch the inspection, be sure AutoVISION is running, and connected to the device. Be sure the job is running. If you choose not to run AutoVISION, you may look at the LEDs on the device to confirm your triggers. Change the value to the right of Trigger to **TRUE**, and click the indicated icon.

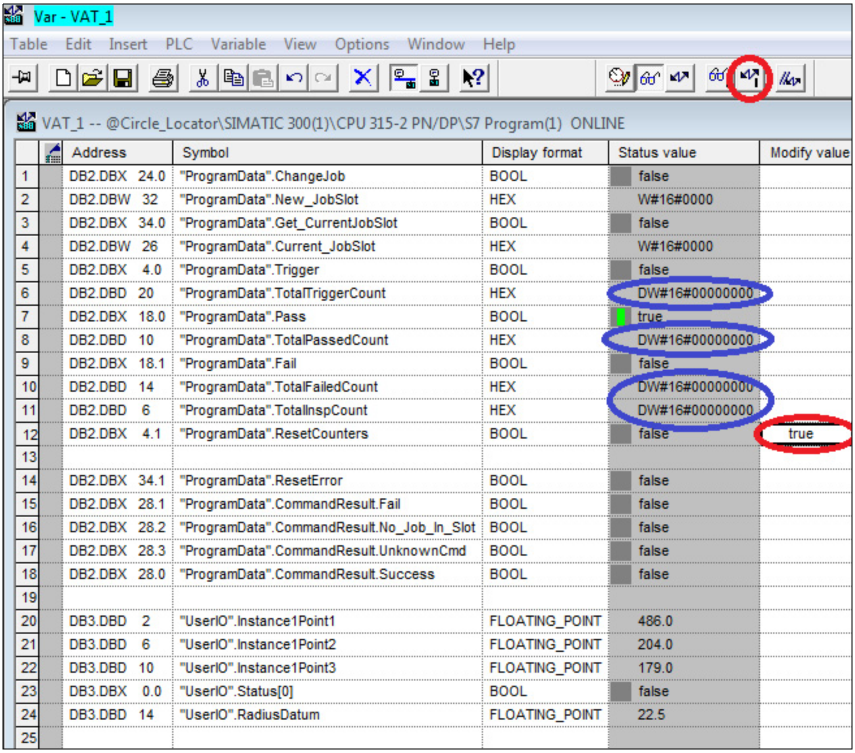



| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | DB2.DBX 24.0 | "ProgramData".ChangeJob | BOOL | false | |
| 2 | DB2.DBW 32 | "ProgramData".New_JobSlot | HEX | W#16#0000 | |
| 3 | DB2.DBX 34.0 | "ProgramData".Get_CurrentJobSlot | BOOL | false | |
| 4 | DB2.DBW 26 | "ProgramData".Current_JobSlot | HEX | W#16#0000 | |
| 5 | DB2.DBX 4.0 | "ProgramData".Trigger | BOOL | false | |
| 6 | DB2.DBD 20 | "ProgramData".TotalTriggerCount | HEX | DW#16#00000004 | |
| 7 | DB2.DBX 18.0 | "ProgramData".Pass | BOOL | true | |
| 8 | DB2.DBD 10 | "ProgramData".TotalPassedCount | HEX | DW#16#00000002 | |
| 9 | DB2.DBX 18.1 | "ProgramData".Fail | BOOL | false | |
| 10 | DB2.DBD 14 | "ProgramData".TotalFailedCount | HEX | DW#16#00000002 | |
| 11 | DB2.DBD 6 | "ProgramData".TotalInspCount | HEX | DW#16#00000004 | |
| 12 | DB2.DBX 4.1 | "ProgramData".ResetCounters | BOOL | false | |
| 13 | | | | | |
| 14 | DB2.DBX 34.1 | "ProgramData".ResetError | BOOL | false | |
| 15 | DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | false | |
| 16 | DB2.DBX 28.2 | "ProgramData".CommandResult.No_Job_In_Slot | BOOL | false | |
| 17 | DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | false | |
| 18 | DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | false | |
| 19 | | | | | |
| 20 | DB3.DBD 2 | "UserIO".Instance1Point1 | FLOATING_POINT | 486.0 | |
| 21 | DB3.DBD 6 | "UserIO".Instance1Point2 | FLOATING_POINT | 204.0 | |
| 22 | DB3.DBD 10 | "UserIO".Instance1Point3 | FLOATING_POINT | 179.0 | |
| 23 | DB3.DBX 0.0 | "UserIO".Status[0] | BOOL | false | |
| 24 | DB3.DBD 14 | "UserIO".RadiusDatum | FLOATING_POINT | 22.5 | |

You will see various changes as shown above. The pass and fail bools will indicate whether the inspection has passed or failed. Also, the appropriate counters will have incremented. Each click of the icon will cause an inspection which will bounce from one image to the next. One will pass and the next will fail.

## Reset Counters

To reset the counters, clear all settings in the Modify value column, change the one next to **ResetCounters** to **TRUE**, and click the icon again. You will see all the counters in blue reset to **0**.

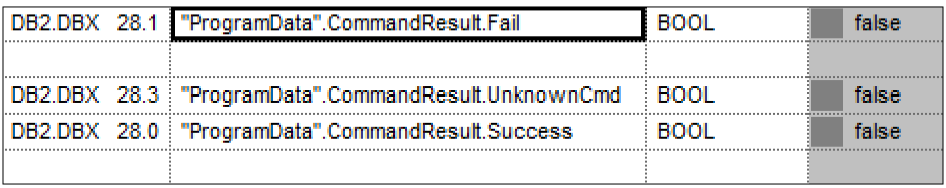



| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | DB2.DBX 24.0 | "ProgramData".ChangeJob | BOOL | false | |
| 2 | DB2.DBW 32 | "ProgramData".New_JobSlot | HEX | W#16#0000 | |
| 3 | DB2.DBX 34.0 | "ProgramData".Get_CurrentJobSlot | BOOL | false | |
| 4 | DB2.DBW 26 | "ProgramData".Current_JobSlot | HEX | W#16#0000 | |
| 5 | DB2.DBX 4.0 | "ProgramData".Trigger | BOOL | false | |
| 6 | DB2.DBD 20 | "ProgramData".TotalTriggerCount | HEX | DW#16#00000000 | |
| 7 | DB2.DBX 18.0 | "ProgramData".Pass | BOOL | true | |
| 8 | DB2.DBD 10 | "ProgramData".TotalPassedCount | HEX | DW#16#00000000 | |
| 9 | DB2.DBX 18.1 | "ProgramData".Fail | BOOL | false | |
| 10 | DB2.DBD 14 | "ProgramData".TotalFailedCount | HEX | DW#16#00000000 | |
| 11 | DB2.DBD 6 | "ProgramData".TotalInspCount | HEX | DW#16#00000000 | |
| 12 | DB2.DBX 4.1 | "ProgramData".ResetCounters | BOOL | false | true |
| 13 | | | | | |
| 14 | DB2.DBX 34.1 | "ProgramData".ResetError | BOOL | false | |
| 15 | DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | false | |
| 16 | DB2.DBX 28.2 | "ProgramData".CommandResult.No_Job_In_Slot | BOOL | false | |
| 17 | DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | false | |
| 18 | DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | false | |
| 19 | | | | | |
| 20 | DB3.DBD 2 | "UserIO".Instance1Point1 | FLOATING_POINT | 486.0 | |
| 21 | DB3.DBD 6 | "UserIO".Instance1Point2 | FLOATING_POINT | 204.0 | |
| 22 | DB3.DBD 10 | "UserIO".Instance1Point3 | FLOATING_POINT | 179.0 | |
| 23 | DB3.DBX 0.0 | "UserIO".Status[0] | BOOL | false | |
| 24 | DB3.DBD 14 | "UserIO".RadiusDatum | FLOATING_POINT | 22.5 | |
| 25 | | | | | |

If you clear this again and set the Trigger to **TRUE**, and click the lightning bolt button again, you will notice that the counts start at **1** on both the PLC counters and AutoVISION counters.

## Reset Error

If at any time there is an error indicated by a TRUE in any of the values below, you can clear the error by entering a TRUE to the right of **ResetError** and clicking the icon. Reset it to **FALSE** and click the button again. Any errors in the values below should be cleared.



| | | | |
|---|---|---|---|
| DB2.DBX 28.1 | "ProgramData".CommandResult.Fail | BOOL | false |
| | | | |
| DB2.DBX 28.3 | "ProgramData".CommandResult.UnknownCmd | BOOL | false |
| DB2.DBX 28.0 | "ProgramData".CommandResult.Success | BOOL | false |
| | | | |

13

CHAPTER 13

# Legacy PROFINET I/O Demos Using Step 7 and TIA Portal V13 with MicroHAWK

This section describes how to use Omron Microscan demo PLC code with a vision job and camera target. The PROFINET I/O demo files can be found where AutoVISION is installed, in these folders: **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\STEP7_Demos\Legacy_AVDemo** and **C:\Microscan\Tutorials and Samples\ MicroHAWK\PROFINET demo\TIAPortal_Demos\Legacy_AVDemo**. Open **Legacy_AVDemo.avp** with AutoVISION and download it to the camera.

**Notes:**

- The camera communications protocol must be enabled for PROFINET I/O before it can be used in this environment. Refer to **Chapter 1, Enabling Industrial Protocols**, for information about enabling and switching communications protocols.
- AutoVISION and FrontRunner jobs use Omron Microscan Link functionality to accommodate PROFINET I/O communications between the camera and the PLC. For information about how to connect job parameters and outputs to Omron Microscan Link tags, refer to the **Linking Datums to Omron Microscan Link Tags** section of **Chapter 2** in the *Visionscape FrontRunner User Manual*. **Important:** PROFINET I/O allows the IP address and subnet mask of the camera to be assigned automatically by the PLC. In order to allow this, the MicroHAWK may not have an assigned IP address on reboot until the PLC is set to Run mode. During this time, the MicroHAWK will not be visible on the network for AutoVISION or Visionscape FrontRunner.

# STEP 7 Overview

In this demonstration, you will learn how to load a saved job into the camera, establish connectivity via **PROFINET I/O** to a **Siemens S7 PLC**, and run some example programs that interface with the camera.

While evaluating PROFINET I/O capabilities, you will:

- **Begin with AutoVISION.**

Open a sample demo vision job in AutoVISION and use the Try Out feature to learn what to expect from the camera before it is connected to the PLC.

- **Prepare the PLC.**

Integrate the camera into the PLC environment with STEP 7 software and the GSD file.

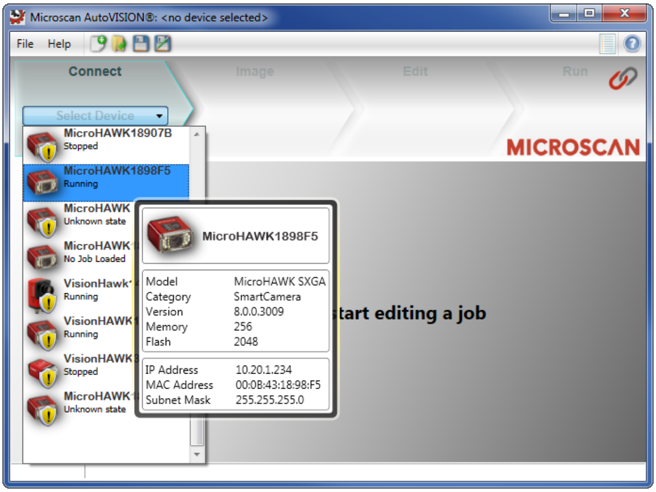
- **Run the camera.**

Trigger the camera while it is online with the PLC and observe changes in the Inspection status as the PLC reconfigures the vision job's parameters.

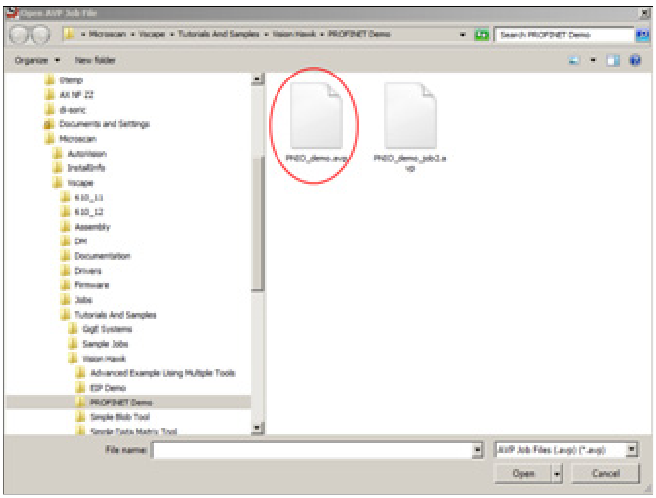
# AutoVISION Setup for Step 7 Demo

Prior to starting AutoVISION, make sure the camera is either connected to the PLC or both PLC and camera are on the same physical network. Ensure that the PC, PLC, and camera have the same network class and corresponding subnet addresses.

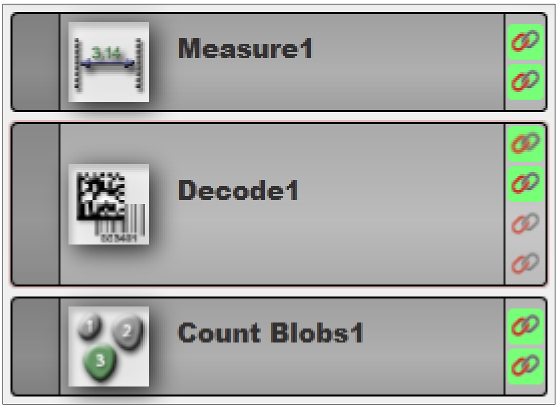
Open AutoVISION and select the camera.




From the Image view, click the **Load a Job** button. Then navigate to **\Microscan\Vscape\Tutorials And Samples\MicroHAWK\PROFINET demo\STEP7_Demos\Legacy_AVDemo**. Select **Legacy_AVDemo.avp**.



The demo job will include three tools: **Measure**, **Decode**, and **Count Blobs**.



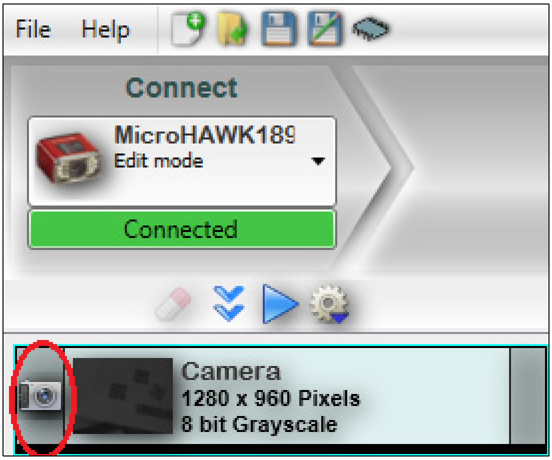

Each tool has data items linked to the PROFINET I/O structure as shown here.

| Tool Result | PLC/PNIO IN tag |
|---|---|
| Measurement Status | "USER".Demo.MeasureStatus (DB101.DBX 1001) |
| Measurement | "USER".Demo.ReadDistance (DB101.DBD 58) |
| Decode Status (matchcode) | "USER".Demo.DecodeStatus (DB101.DBX 100.2) |
| Decode | "USER".Demo.ReadString[32] (DB101.DBB 64-96) |
| Count Blob Status | "USER".Demo.CountBlobStatus (DB101.DBX 100.3) |
| Count Blob Count | "USER".Demo.ReadBlobCount (DB101.DBW 62) |

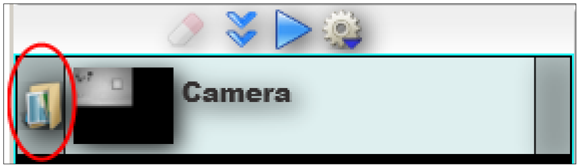
| Tool Parameter | PLC/PNIO OUT tag |
|---|---|
| Measurement Tolerance Low | "USER".Demo.MinDistance (DB101.DBD 46) |
| Measurement Tolerance High | "USER".Demo.MaxDistance (DB101.DBD 50) |
| Decode Matchcode | "USER".Demo.MatchCode [32] (DB101.DBB 12-44) |
| Count Blob Lower Tolerance | "USER".Demo.MinBlobCount (DB101.DBW 54) |
| Count Blob Upper Tolerance | "USER".Demo.MaxBlobCount (DB101.DBW 56) |

This data is transferred cyclically between the camera and PLC.

Once the job has loaded, the next step is to link the pre-saved images on the local PC. On the **Camera** button, click the far left icon to select and load an image.




A file browser will open. Then navigate to the same folder where the demo job was loaded – **PROFINET demo**. If the images are located, the icon will change from a camera to a folder.




By clicking the **Try Job Once** icon, the application will cycle through the entire job with the loaded image.

There are two images on file. One will pass all three inspections and one will fail all three.


Now click the **Run** button on the top ribbon. This will download the job to the camera.

At this point the job is ready to run and can be tested. However, in order to run the job change demo this job needs to be loaded into **Slot 1**. Click back to the Edit view from the top ribbon bar.

Click the slot icon and select Slot 1 or **New Slot** if no slots are currently in use.

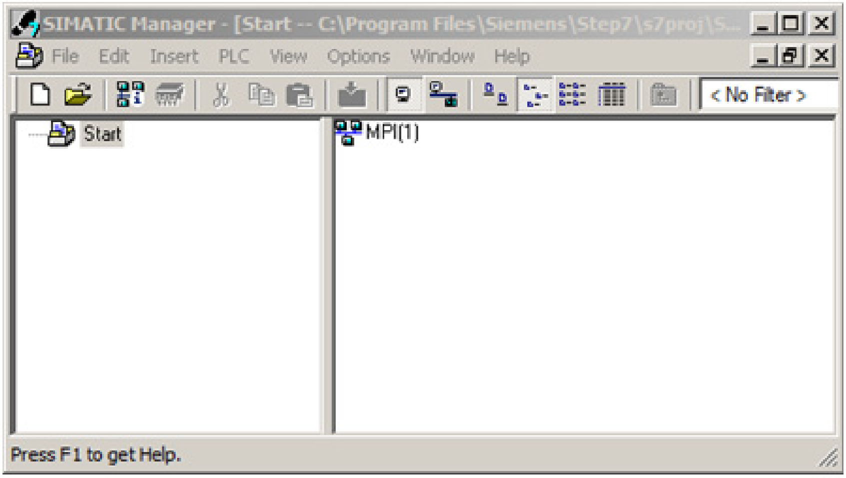

Now the job and images will be saved to the flash memory of the camera.

# STEP 7 Setup

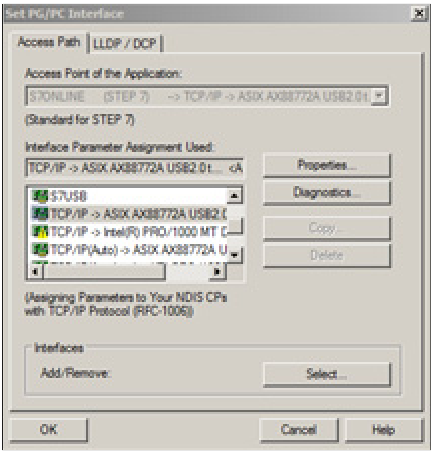
This section was created using Siemens STEP 7 software version 5.5 + SP2 and an ET200S PLC, catalog number 6ES7 151-8AB01-0AB0, CPU Version 3.2. It was tested with a 315-2 PN/DP PLC, catalog number 6ES7-315-2EH13-0AB0, CPU Version 2.6.

First launch the SIMATIC Manager from the desktop.

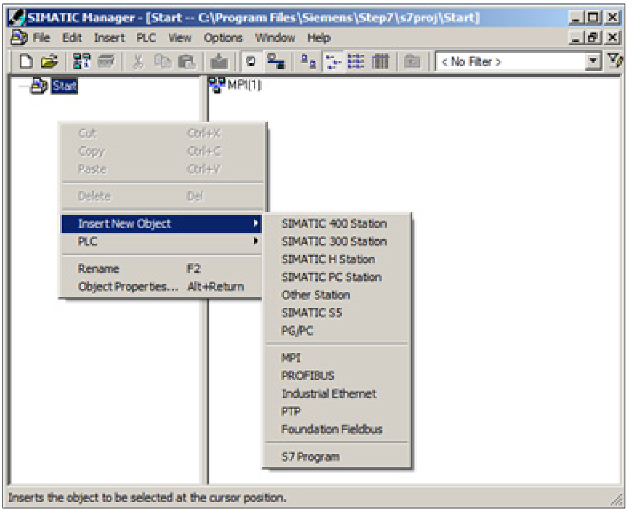

Create a new project from the menu item **File > New**. Select the project location on disk then enter the name and click the **OK** button. In the example below, the name is **Start**. Once the project is first created, you will see the dialog. This dialog is the main entry point into the PLC program and hardware settings.



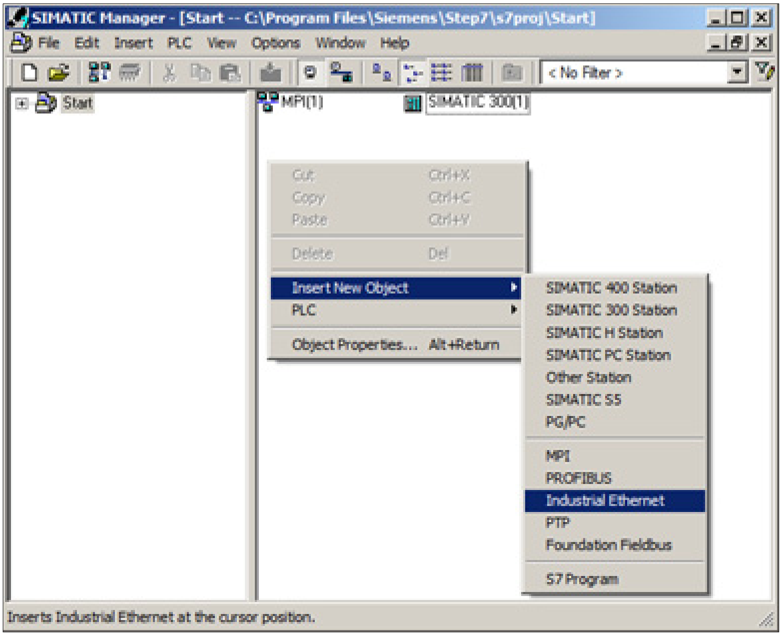
The PC may have multiple network cards so it's imperative to map the correct one to the SIMATIC software. From the menu, select Options > Set PG/PC Interface. This will open the Set PG/PC Interface dialog and list the available network cards. On the Access Path tab select the NIC card with (TCP/IP >) in the name.



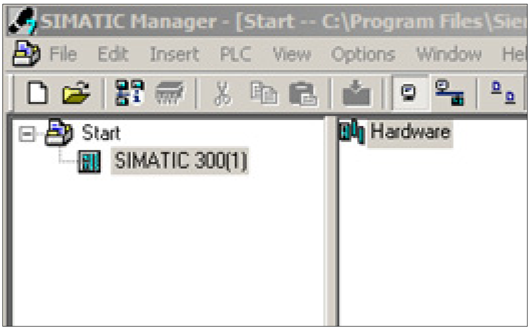

In the SIMATIC Manager dialog, right-click and select Insert New Object. This is where you will select the base station. For example, when configuring an ET200-s, select SIMATIC 300 Station since it's based on the 300 series CPU.



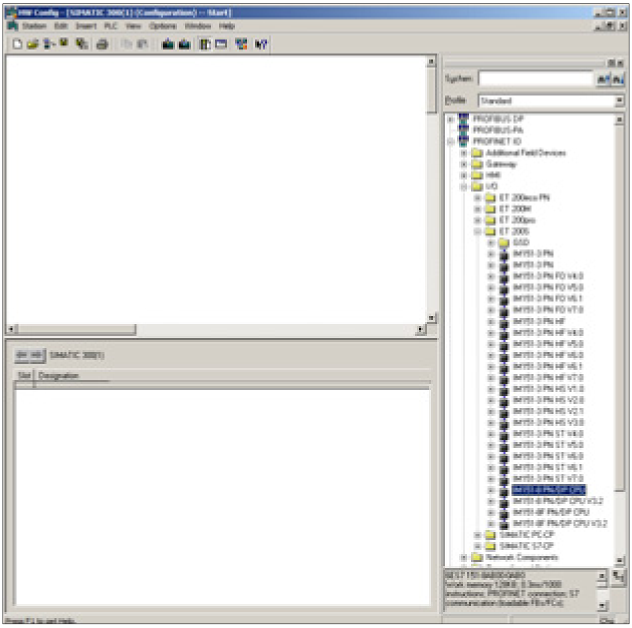
After the station has been added, right-click and add **Industrial Ethernet**.



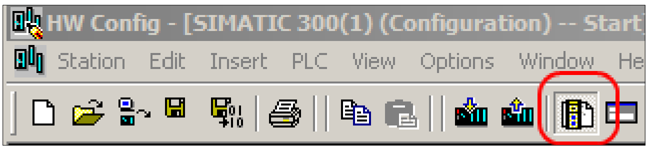

Expand the Start tree node on the left pane and click the station. On the right pane you will see a Hardware icon.



Double-click the **Hardware** icon to launch the **HW Config** dialog.



Make sure the **Catalog** is selected in the ribbon bar on the top. This will add a tree view on the right pane with all the available hardware devices.

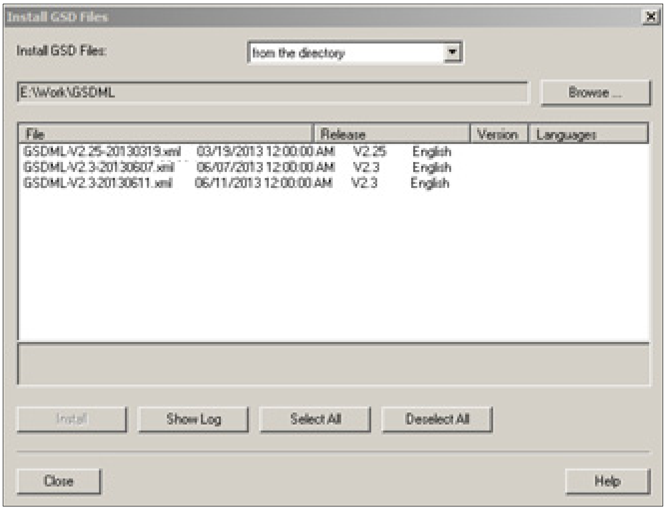



If the latest GSDML file hasn't been imported, follow the next steps to import.

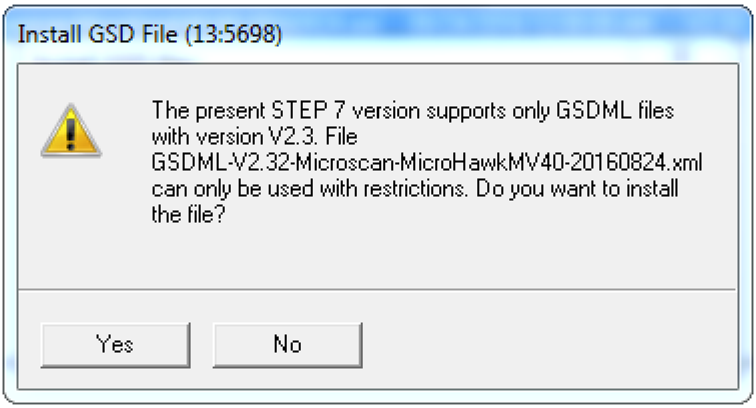
## GSDML

- Install the GSD file from the menu item **Options > Install GSD File**.
- Click the Browse button to locate the GSDML file **\Microscan\Vscape\Firmware\gsd\MicroHAWK**.

- The center pane will list the available GSDML files.

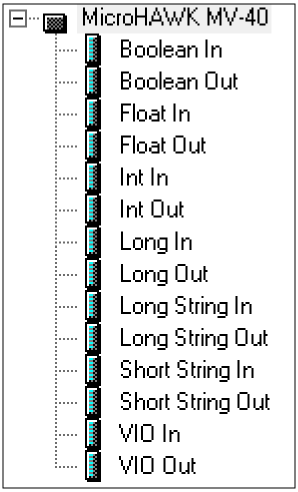



- Select the file and click the Install button.
- You will receive the following error message:

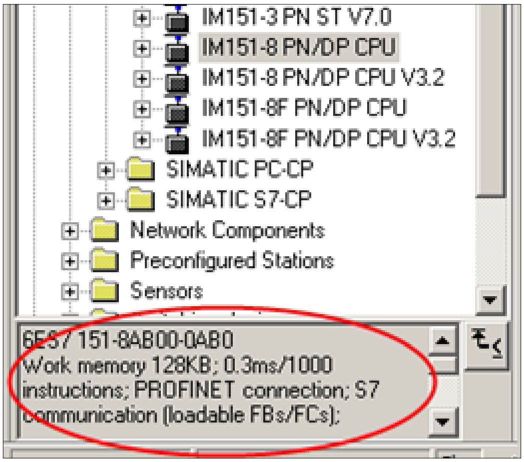



- Click **Yes** to install the file.
- When finished, close the dialog.

- The camera should now be present in the PROFINET I/O section as shown below.



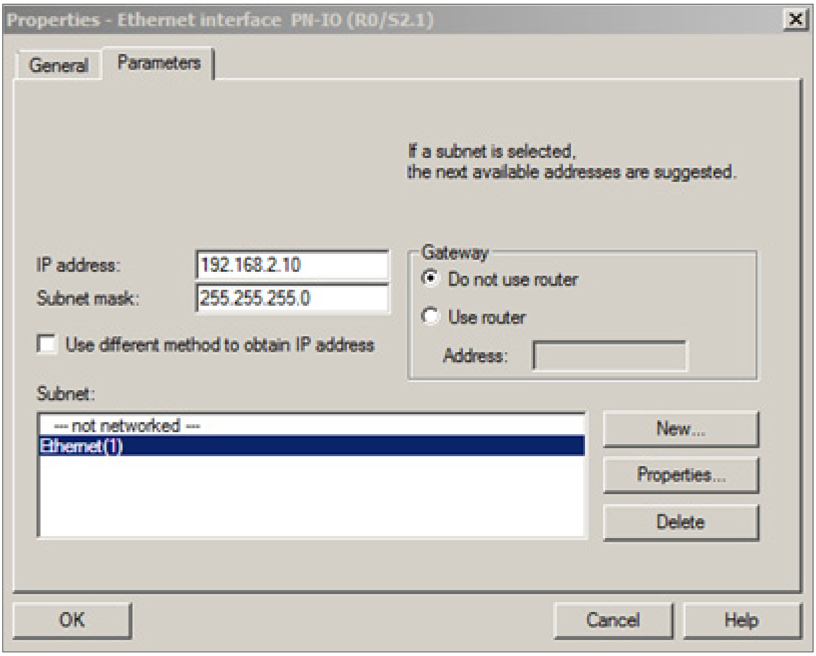

Add the CPU from the catalog view by drag and drop or double-click. Make sure the catalog number and version matches the PLC exactly. The catalog number will be displayed on the bottom of the view.




Some CPUs are modules that will require a generic rack to be added prior. If your PLC requires a rack, you will be prompted to add the rack prior to being allowed to insert the CPU module.

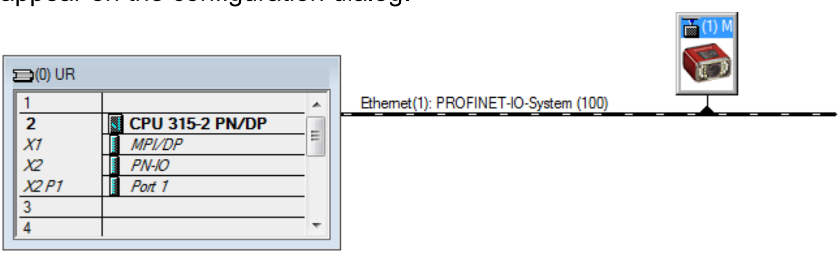
After the CPU is added, a popup dialog will prompt the Properties relating to the IP information.

Select **Ethernet(1)** on the bottom list box and enter the correct IP address of the CPU.

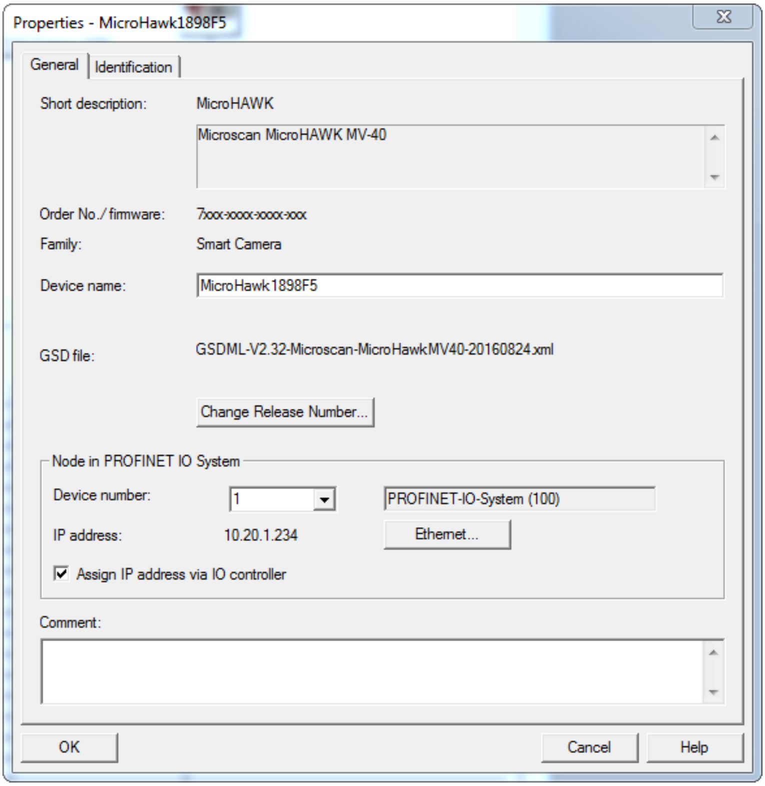



Once the CPU has been added, add any additional expansion modules, if installed. Click the CPU on the left pane and the bottom pane will list the available ports and slots.?

Add the camera in the location (**PROFINET I/O > Additional Field Devices > General > Microscan > Smart Camera\MicroHAWK MV-40**). The camera can be added by drag and drop to **Ethernet(1)**: **PROFINET-I/O System (100)** or by selecting the **Ethernet(1)** line first and double-clicking the camera. Once the camera has been added the icon will appear on the configuration dialog.



Once the camera has been added, double-click the icon to open the properties dialog. Under the device name, enter the existing name of the camera or a unique name.



If a unique name is used, the device has to be manually updated. View Updating camera name section.

Click the camera icon and the data slot address mapping will be displayed below. Take note of the address values since they will be needed in the demo application. Since there is an infinite combination of modules and slot configurations, the addressing is unique to every setup.

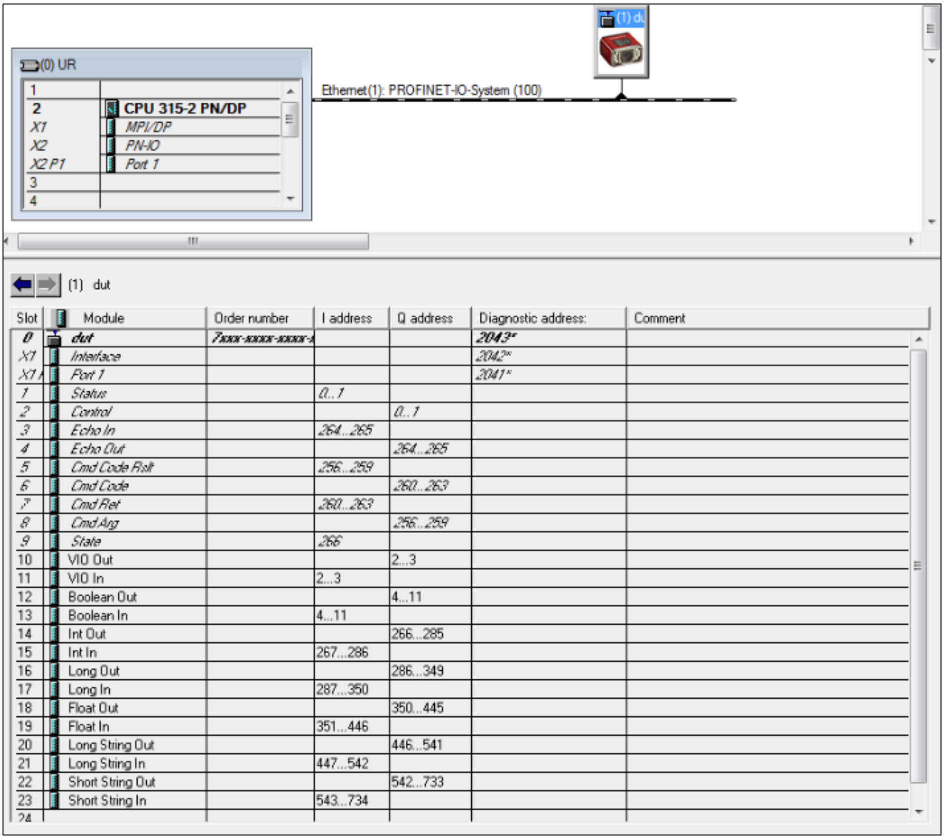
(2) MicroHAWK

| Slot | Module | Order number | I address | Q address | Diagnostic address: |
|---|---|---|---|---|---|
| 0 | MicroHAWK | 7xxx-xxxx-xxxx-; | | | 2040* |
| X1 | Interface | | | | 2039* |
| X1 / | Port 1 | | | | 2038* |
| 1 | Status | | 12...13 | | |
| 2 | Control | | | 12...13 | |
| 3 | Echo In | | 743...744 | | |
| 4 | Echo Out | | | 742...743 | |
| 5 | Cmd Code Rslt | | 735...738 | | |
| 6 | Cmd Code | | | 738...741 | |
| 7 | Cmd Ret | | 739...742 | | |
| 8 | Cmd Arg | | | 734...737 | |
| 9 | State | | 745 | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |
| 17 | | | | | |
| 18 | | | | | |
| 19 | | | | | |
| 20 | | | | | |
| 21 | | | | | |
| 22 | | | | | |
| 23 | | | | | |

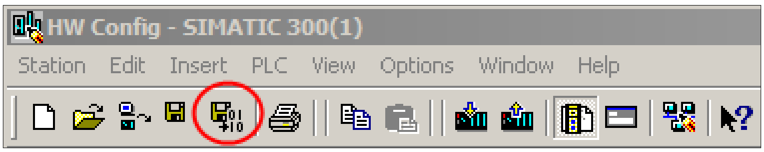
You must manually add the remaining slots into the display shown on the previous page from the database to the right.

- MicroHAWK MV-40
  - Boolean In
  - Boolean Out
  - Float In
  - Float Out
  - Int In
  - Int Out
  - Long In
  - Long Out
  - Long String In
  - Long String Out
  - Short String In
  - Short String Out
  - VIO In
  - VIO Out

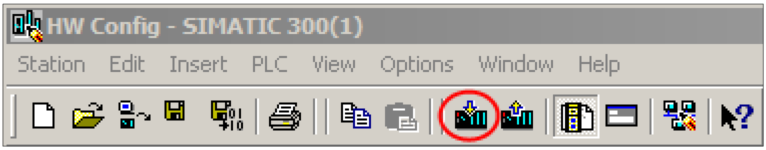
Drag and drop all these data elements into the HW display. It will only allow you to drop them in the appropriate slot. When finished, your HW display should resemble the following:



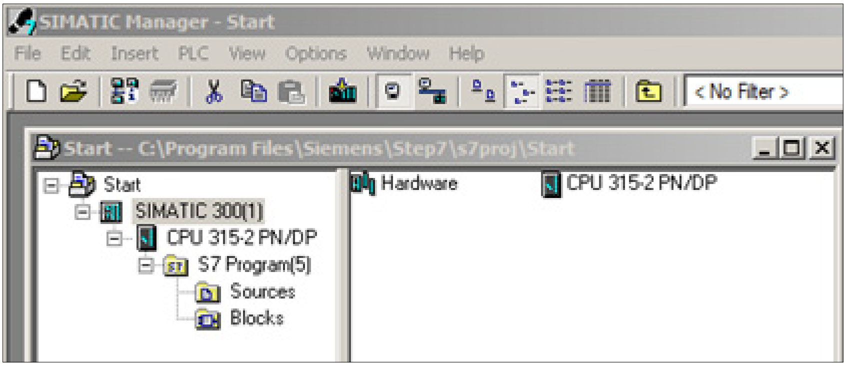

Once the hardware configuration has been completed, it's time to compile and download. Click the compile and save icon on the ribbon. If there are any configuration conflicts, the application will prompt a warning at this point.



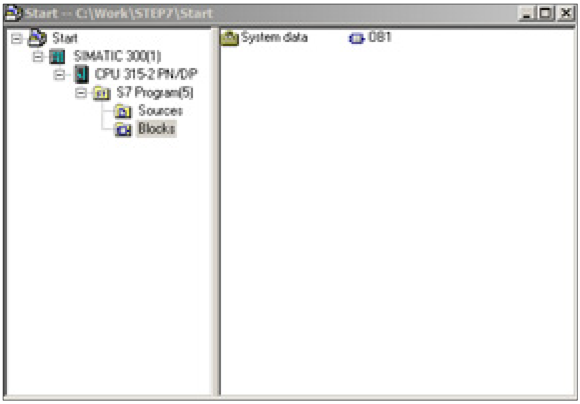
Once compiled, click the download icon on the ribbon to send the information to the PLC.




At this point close or minimize the **HW Config** dialog and re-visit the **SIMATIC Manager** dialog. The CPU should be added next to the Hardware icon and in the tree view in the left pane. Expand the CPU tree item and remaining child items below it.



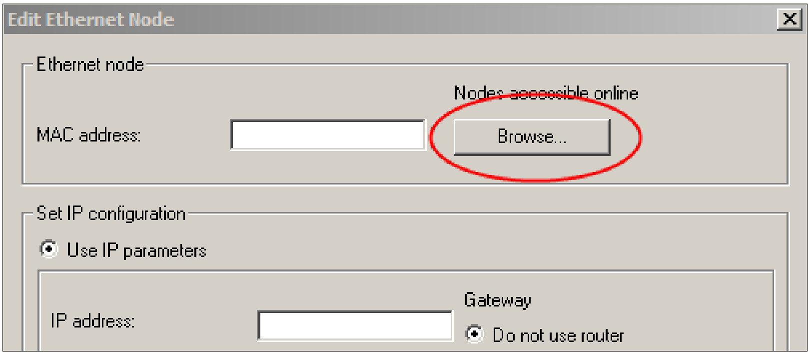

Click the Blocks node to view the program objects.



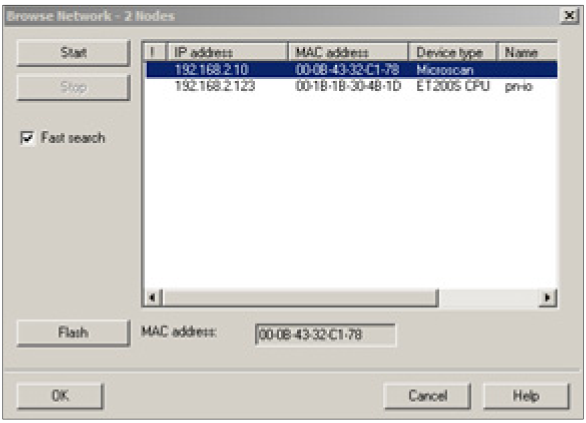
## Updating Camera Name

From the HW Config dialog open **Edit Ethernet Node** from the menu item **PLC > Ethernet > Edit Ethernet Node**. From the SIMATIC Manager dialog open Edit Ethernet Node from the menu item **PLC > Edit Ethernet Node**.

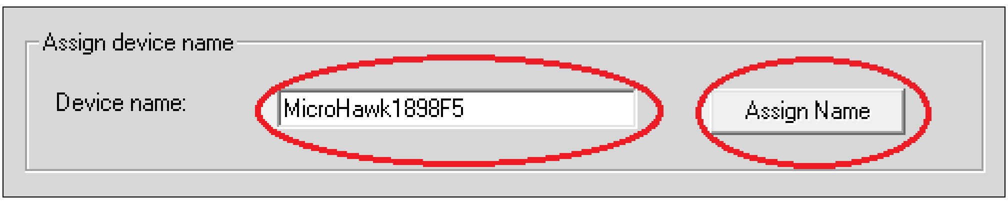
Click the **Browse** button to open the node selection dialog.




Select the camera and click the **OK** button.

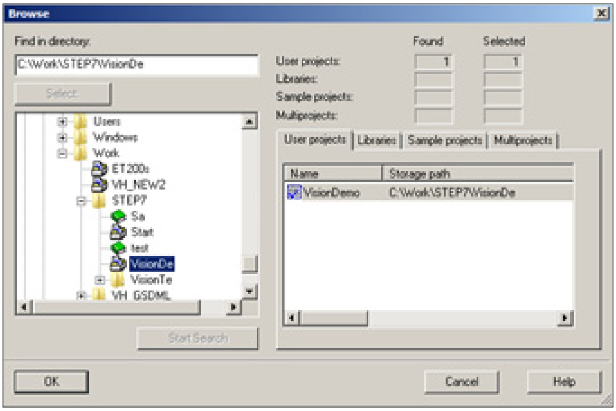



Edit the **Device Name**, if necessary, to match the name entered previously in the MicroHAWK properties dialog, and click the **Assign Name** button.

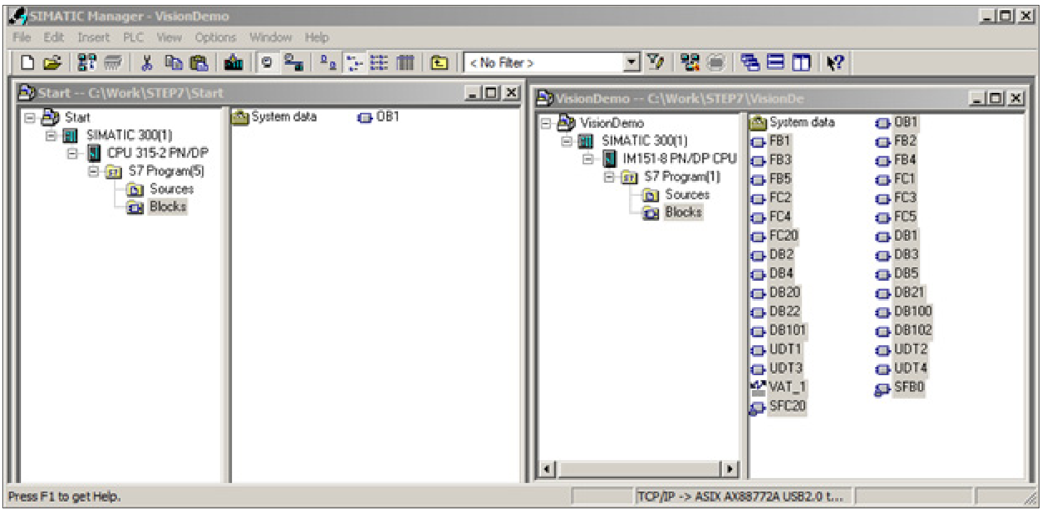


## Importing Example Program

From the SIMATIC Manager, open the example program from the menu **File > Open**. Click the **Browse** button to locate the (AVDemo) program. The AVDemo is located in **\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\STEP7_Demos\Legacy_AVDemo\300PLC_w.s7p**.

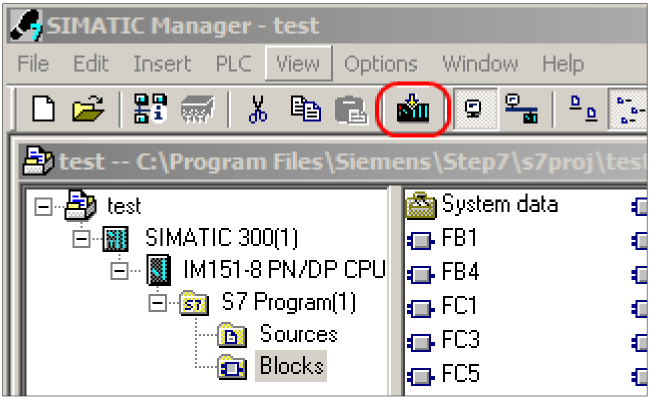



Split both example and current projects in the dialog as shown.

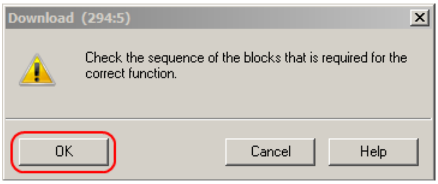


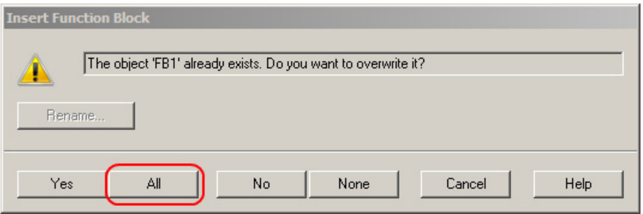

Select the objects minus the **System** data object and copy with right-click > copy or CTRL+C. Paste into current project with right-click > past or CTRL+V. When prompted that **OB1** already exists, click the **Yes** button to overwrite it. Now the example program is imported to the current project.

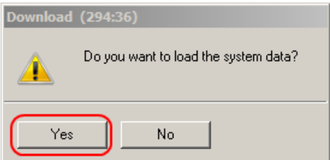
Select the **Blocks** icon on the left pane. Then click the download button on the top ribbon. This will download the new copied functions and system data to the PLC. Now it's time to update the new addressing from the hardware installation prior. Double-click the **OB1** block to open the **LAD/STL/FBD** editor. OB1 is the main routine of the PLC program.

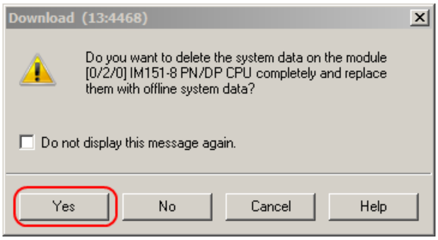


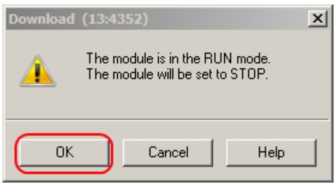
Scroll down to **Network 4**. This is where the data is mapped from the camera to the local data structure. **FC3** is a function that pushes the input data from the camera to the program structure. Click the numbers to match the address on the hardware as shown. Click the following buttons on the popup dialogs:

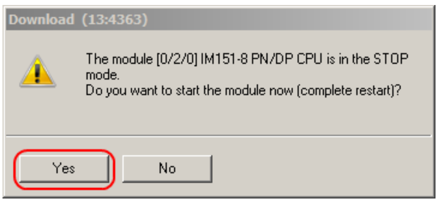


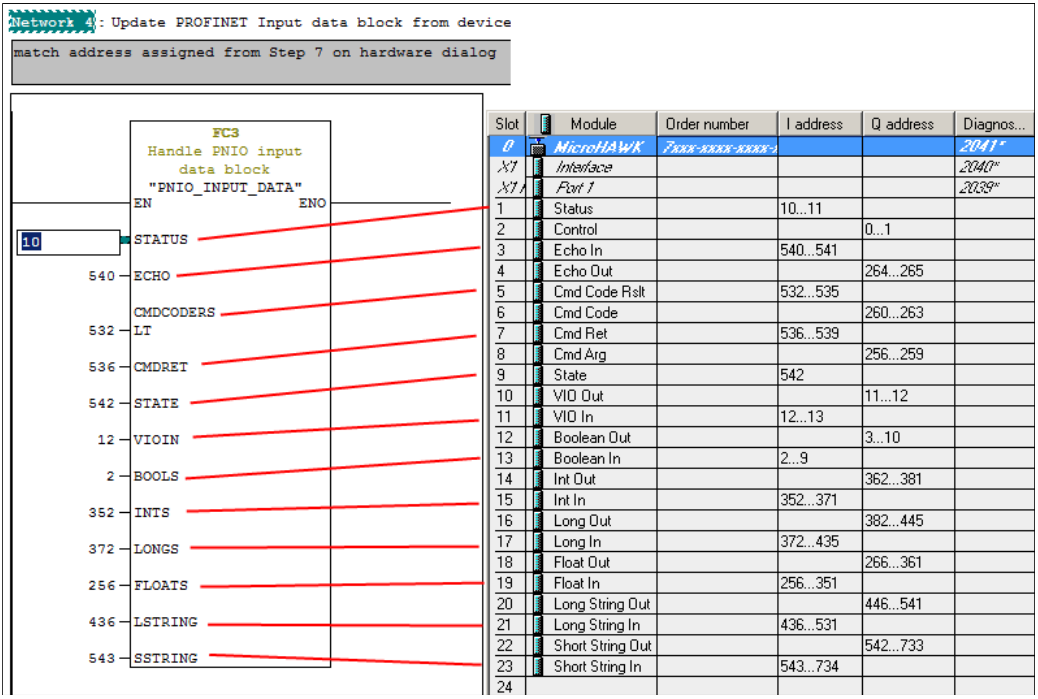



**Note:** This is the address mapping view of the MicroHAWK module which is derived from the hardware config view.



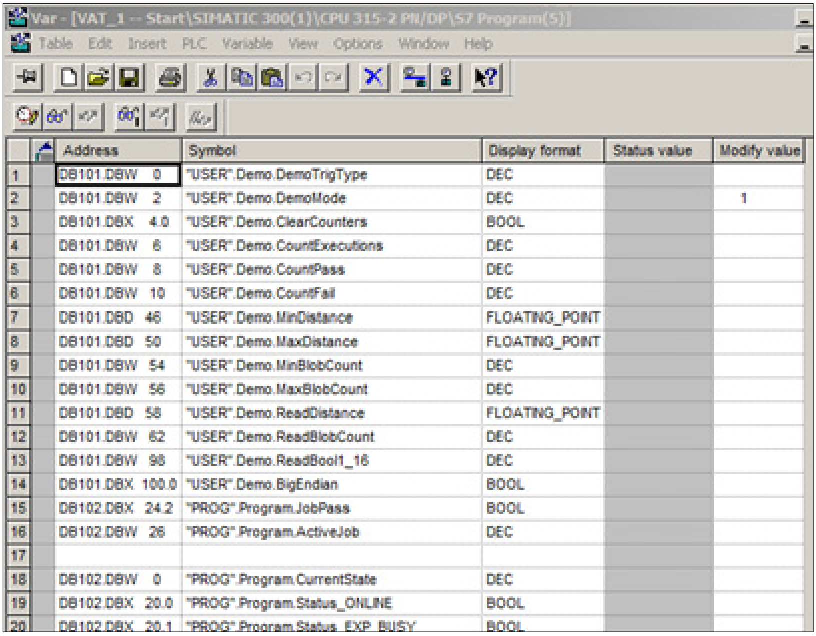

| Slot | Module | Order number | I address | Q address | Diagnos... |
|---|---|---|---|---|---|
| 0 | MicroHAWK | 7xxx-xxxx-xxxx-x | | | 2041* |
| X1 | Interface | | | | 2040* |
| X1 P | Port 1 | | | | 2039* |
| 1 | Status | | 10...11 | | |
| 2 | Control | | | 0...1 | |
| 3 | Echo In | | 540...541 | | |
| 4 | Echo Out | | | 264...265 | |
| 5 | Cmd Code Rslt | | 532...535 | | |
| 6 | Cmd Code | | | 260...263 | |
| 7 | Cmd Ret | | 536...539 | | |
| 8 | Cmd Arg | | | 256...259 | |
| 9 | State | | 542 | | |
| 10 | VIO Out | | | 11...12 | |
| 11 | VIO In | | 12...13 | | |
| 12 | Boolean Out | | | 3...10 | |
| 13 | Boolean In | | 2...9 | | |
| 14 | Int Out | | | 362...381 | |
| 15 | Int In | | 352...371 | | |
| 16 | Long Out | | | 382...445 | |
| 17 | Long In | | 372...435 | | |
| 18 | Float Out | | | 266...361 | |
| 19 | Float In | | 256...351 | | |
| 20 | Long String Out | | | 446...541 | |
| 21 | Long String In | | 436...531 | | |
| 22 | Short String Out | | | 542...733 | |
| 23 | Short String In | | 543...734 | | |
| 24 | | | | | |

Repeat the same steps for **Network 5** to update the output data mapping. Keep in mind that all input addresses are under the **(I address)** column and the output addresses are under the **(Q address)** column. Save to the PC and download to the PLC.

Do a master reset and set the PLC to RUN mode. Make sure all LEDs indicate **good**.

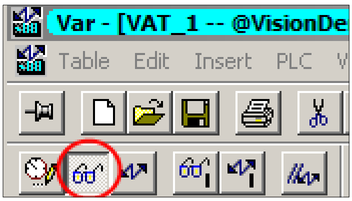
## Running the Demo

In the **SIMATIC Manager** dialog, double-click the **VAT_1** icon. This will open the variable table dialog for the data type demo. Maximize the internal dialog to increase the viewing area.




| | Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|---|
| 1 | DB101.DBW 0 | "USER".Demo.DemoTrigType | DEC | | |
| 2 | DB101.DBW 2 | "USER".Demo.DemoMode | DEC | | 1 |
| 3 | DB101.DBX 4.0 | "USER".Demo.ClearCounters | BOOL | | |
| 4 | DB101.DBW 6 | "USER".Demo.CountExecutions | DEC | | |
| 5 | DB101.DBW 8 | "USER".Demo.CountPass | DEC | | |
| 6 | DB101.DBW 10 | "USER".Demo.CountFail | DEC | | |
| 7 | DB101.DBD 46 | "USER".Demo.MinDistance | FLOATING_POINT | | |
| 8 | DB101.DBD 50 | "USER".Demo.MaxDistance | FLOATING_POINT | | |
| 9 | DB101.DBW 54 | "USER".Demo.MinBlobCount | DEC | | |
| 10 | DB101.DBW 56 | "USER".Demo.MaxBlobCount | DEC | | |
| 11 | DB101.DBD 58 | "USER".Demo.ReadDistance | FLOATING_POINT | | |
| 12 | DB101.DBW 62 | "USER".Demo.ReadBlobCount | DEC | | |
| 13 | DB101.DBW 98 | "USER".Demo.ReadBool1_16 | DEC | | |
| 14 | DB101.DBX 100.0 | "USER".Demo.BigEndian | BOOL | | |
| 15 | DB102.DBX 24.2 | "PROG".Program.JobPass | BOOL | | |
| 16 | DB102.DBW 26 | "PROG".Program.ActiveJob | DEC | | |
| 17 | | | | | |
| 18 | DB102.DBW 0 | "PROG".Program.CurrentState | DEC | | |
| 19 | DB102.DBX 20.0 | "PROG".Program.Status_ONLINE | BOOL | | |
| 20 | DB102.DBX 20.1 | "PROG".Program.Status_EXP_BUSY | BOOL | | |

To establish a live connection to the PLC, click the **Monitor Variable** button on the ribbon. This will update data from the PLC to the dialog. The top title bar will go blue and the bottom status will show run with a green progress bar.



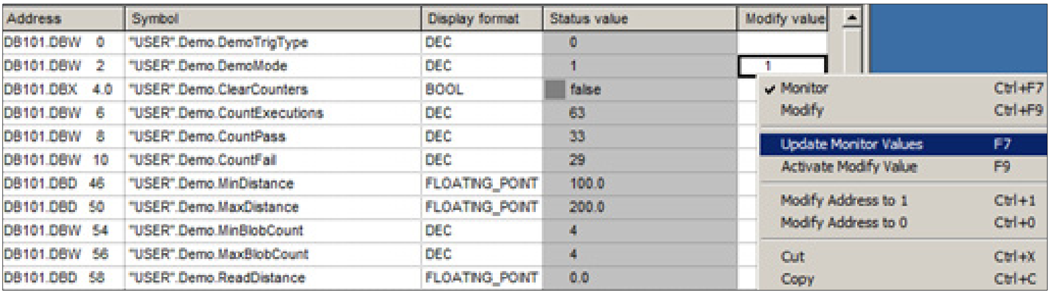
There are two demo modes in this example. The modes are set in **“USER”.Demo.DemoMode (DB101.DBW 2)**.

**0 = AutoVISION Test**

**1 = Job Change Test**

The **AutoVISION Test** will demonstrate communications between the PLC and the current job loaded from AutoVISION.

If **0** doesn't appear under the **Status Value** column, right-click in the **"USER".Demo.DemoMode** row under the **Modify Value** column. Select **Modify** or type **CTRL+F9** to change the value.



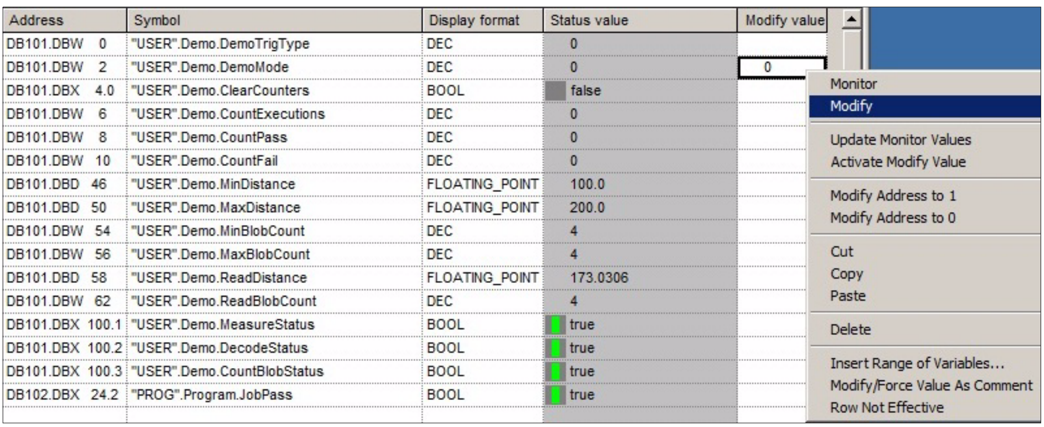
| Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|
| DB101.DBW 0 | "USER".Demo.DemoTrigType | DEC | 0 | |
| DB101.DBW 2 | "USER".Demo.DemoMode | DEC | 1 | 1 |
| DB101.DBX 4.0 | "USER".Demo.ClearCounters | BOOL | false | |
| DB101.DBW 6 | "USER".Demo.CountExecutions | DEC | 63 | |
| DB101.DBW 8 | "USER".Demo.CountPass | DEC | 33 | |
| DB101.DBW 10 | "USER".Demo.CountFail | DEC | 29 | |
| DB101.DBD 46 | "USER".Demo.MinDistance | FLOATING_POINT | 100.0 | |
| DB101.DBD 50 | "USER".Demo.MaxDistance | FLOATING_POINT | 200.0 | |
| DB101.DBW 54 | "USER".Demo.MinBlobCount | DEC | 4 | |
| DB101.DBW 56 | "USER".Demo.MaxBlobCount | DEC | 4 | |
| DB101.DBD 58 | "USER".Demo.ReadDistance | FLOATING_POINT | 0.0 | |



If **0** is not entered in the **Status Value** column of the **“USER.Demo.DemoTrigType”** row, enter **0**.

To start triggering the MicroHAWK program, right-click inside the **Status Value** column of the **“USER.Demo.DemoTrigType” (DB101.DBW 0)** row.

Change the value from **0** to **1** to begin triggering and running the job on the camera.



| Address | Symbol | Display format | Status value | Modify value |
|---|---|---|---|---|
| DB101.DBW 0 | "USER".Demo.DemoTrigType | DEC | 0 | |
| DB101.DBW 2 | "USER".Demo.DemoMode | DEC | 0 | 0 |
| DB101.DBX 4.0 | "USER".Demo.ClearCounters | BOOL | false | |
| DB101.DBW 6 | "USER".Demo.CountExecutions | DEC | 0 | |
| DB101.DBW 8 | "USER".Demo.CountPass | DEC | 0 | |
| DB101.DBW 10 | "USER".Demo.CountFail | DEC | 0 | |
| DB101.DBD 46 | "USER".Demo.MinDistance | FLOATING_POINT | 100.0 | |
| DB101.DBD 50 | "USER".Demo.MaxDistance | FLOATING_POINT | 200.0 | |
| DB101.DBW 54 | "USER".Demo.MinBlobCount | DEC | 4 | |
| DB101.DBW 56 | "USER".Demo.MaxBlobCount | DEC | 4 | |
| DB101.DBD 58 | "USER".Demo.ReadDistance | FLOATING_POINT | 173.0306 | |
| DB101.DBW 62 | "USER".Demo.ReadBlobCount | DEC | 4 | |
| DB101.DBX 100.1 | "USER".Demo.MeasureStatus | BOOL | true | |
| DB101.DBX 100.2 | "USER".Demo.DecodeStatus | BOOL | true | |
| DB101.DBX 100.3 | "USER".Demo.CountBlobStatus | BOOL | true | |
| DB102.DBX 24.2 | "PROG".Program.JobPass | BOOL | true | |

When the program is triggering the camera, each cycle will produce either a pass or fail all state. Each state is counted in **“USER”.Demo.CountPass**:

| | | | |
|---|---|---|---|
| DB101.DBD 58 | "USER".Demo.ReadDistance | FLOATING_POINT | 173.0306 |
| DB101.DBW 62 | "USER".Demo.ReadBlobCount | DEC | 4 |
| DB101.DBX 100.1 | "USER".Demo.MeasureStatus | BOOL | true |
| DB101.DBX 100.2 | "USER".Demo.DecodeStatus | BOOL | true |
| DB101.DBX 100.3 | "USER".Demo.CountBlobStatus | BOOL | true |
| DB102.DBX 24.2 | "PROG".Program.JobPass | BOOL | true |

and **“USER”.Demo.CountFail**:

| | | | |
|---|---|---|---|
| DB101.DBD 58 | "USER".Demo.ReadDistance | FLOATING_POINT | 59.407 |
| DB101.DBW 62 | "USER".Demo.ReadBlobCount | DEC | 6 |
| DB101.DBX 100.1 | "USER".Demo.MeasureStatus | BOOL | false |
| DB101.DBX 100.2 | "USER".Demo.DecodeStatus | BOOL | false |
| DB101.DBX 100.3 | "USER".Demo.CountBlobStatus | BOOL | false |
| DB102.DBX 24.2 | "PROG".Program.JobPass | BOOL | false |

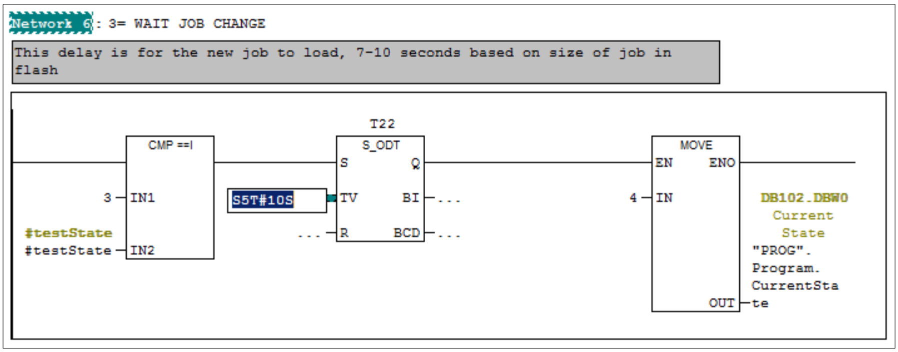
These counters can be cleared by right-clicking the row **“USER”.Demo.ClearCounters** and selecting **Modify Address to 1**.

| | | | |
|---|---|---|---|
| DB101.DBX 4.0 | "USER".Demo.ClearCounters | BOOL | false |
| DB101.DBW 6 | "USER".Demo.CountExecutions | DEC | 104 |
| DB101.DBW 8 | "USER".Demo.CountPass | DEC | 52 |
| DB101.DBW 10 | "USER".Demo.CountFail | DEC | 49 |

The tolerances are below the counters. The inspection values and status bits are below the tolerances.

| | | | |
|---|---|---|---|
| DB101.DBD 46 | "USER".Demo.MinDistance | FLOATING_POINT | 100.0 |
| DB101.DBD 50 | "USER".Demo.MaxDistance | FLOATING_POINT | 200.0 |
| DB101.DBW 54 | "USER".Demo.MinBlobCount | DEC | 4 |
| DB101.DBW 56 | "USER".Demo.MaxBlobCount | DEC | 4 |

In some instances a large job could take longer to load. Increment the delay time to adjust for job size. The image below shows **Function Block 4**, **Network 6**, where the delay is located.

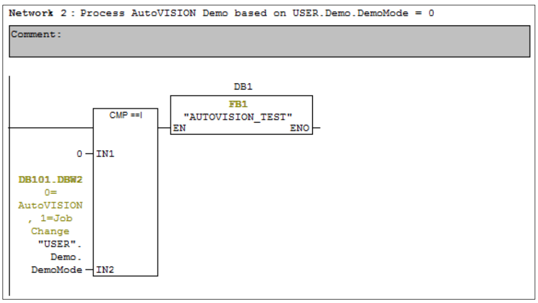


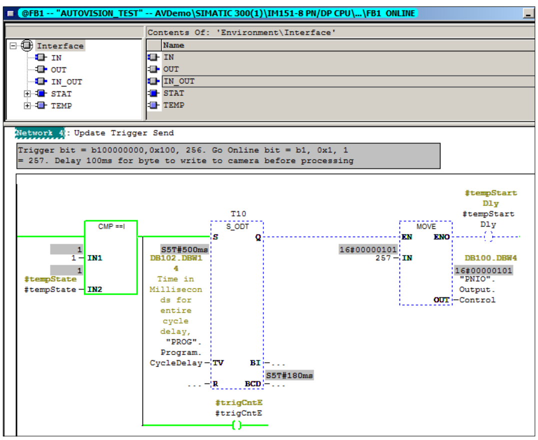
To run the **Job Change Test** you first need to use AutoVISION to download another job to slot 2 on the camera. In AutoVISION, connect to the camera and select the **Connect** icon at the top of the user interface. Click the **Load a Job** icon. Locate **Legacy_AVDemo.avp** and click the **Open** button. Once the job has loaded into AutoVISION, click the **Edit** button at the top of the user interface. Then click **Save the Job to a Slot** on the camera icon in the menu at the top of the user interface. Select **New Slot**, which should be **2**. When switching between tests, disable the current routine by setting the **"USER".Demo.DemoTrigType** to **0**.

Now there should be two jobs loaded in the camera's flash memory. Using the SIMATIC Manager, open **VAT_1**. Click the **Monitor Variable** icon. On the row **"USER".Demo.DemoMode**, right-click and modify to one. Right-click the row **"USER".Demo.DemoTrigType** and **Modify Address** to **1** to start the demo. AutoVISION should be closed for this demo to work.

This demo will cycle through loading jobs from slots **1**, **2**, and **3**. Each cycle will be counted in the variable table **"USER".Demo.CountExecutions**. The job load success is determined by the camera's status register **"Legacy_AVDemo.Input.CmdCodeRslt" = 0x0**. If the job load is successful the counter **"USER".Demo.CountPass** will increment. If the job load fails when **"Legacy_AVDemo".Input.CmdCodeRslt ! = 0x0**, then the counter **"USER".Demo.CountFail** will increment.

**OB1** calls **FB1** to process the AutoVISION test. To view the ladder logic, select **FB1**, right-click, and select **Called Block > Open**. Then select the main menu item **Open ONLINE** to view processes.



# TIA Portal V13 Overview

In this demonstration, you will learn how to load a saved job into the camera, establish connectivity via PROFINET I/O to a Siemens ET200SP CPU PLC, and run some example programs that interface with the camera. While evaluating PROFINET I/O capabilities, you will:

- **Begin with AutoVISION.**

Open a sample demo vision job in AutoVISION and use the Try Out feature to learn what to expect from the camera before it is connected to the PLC.

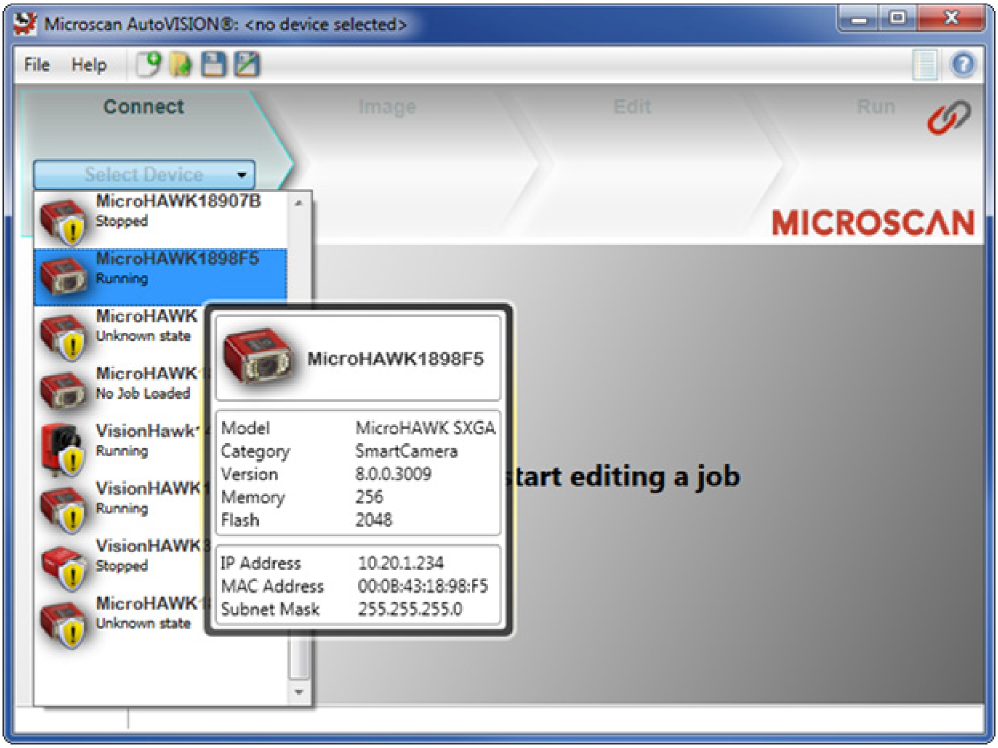
- **Prepare the PLC.**

Integrate the camera into the PLC environment with TIA Portal software and the GSD file.

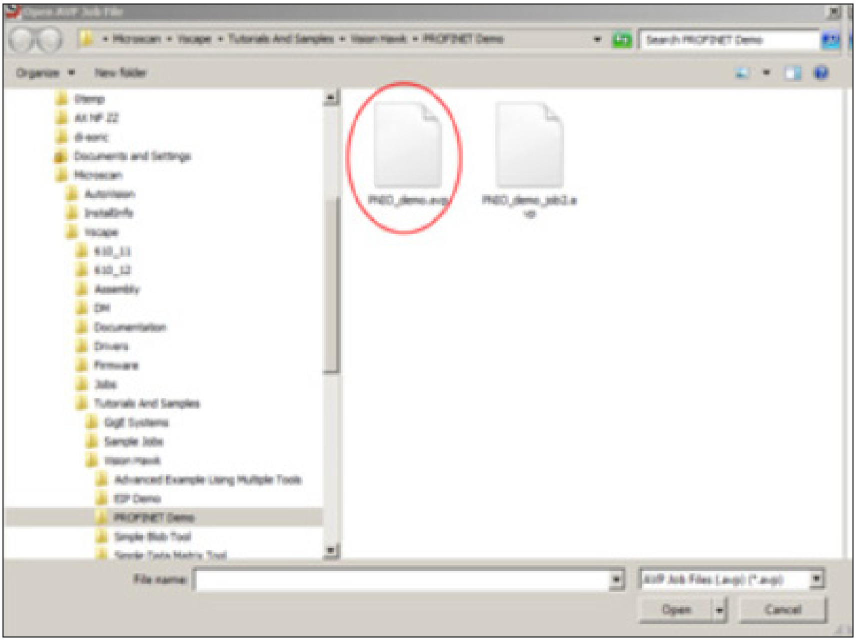
- **Run the camera.**

Trigger the camera while it is online with the PLC and observe changes in the Inspection status as the PLC reconfigures the vision job's parameters. Prior to starting AutoVISION, make sure the camera is either connected to the PLC or both PLC and camera are on the same physical network. Ensure that the PC, PLC, and camera have the same network class and corresponding subnet addresses.

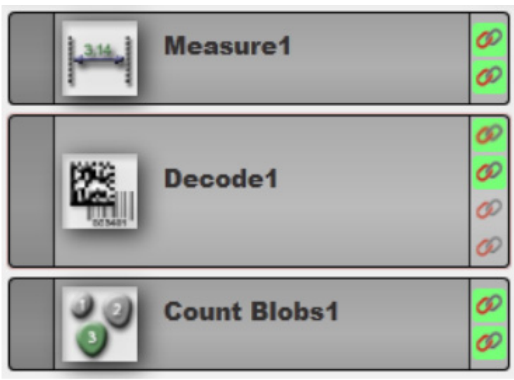
Open AutoVISION and select the camera.



From the Image view, click the **Load a Job** button. Then navigate to **C:\Microscan\Vscape\Tutorials And Samples\MicroHAWK\PROFINET demo\TIAPortal_Demos\Legacy_AVDemo**. Select **Legacy_AVDemo.avp**.




The demo job will include three tools: **Measure**, **Decode**, and **Count Blobs**.



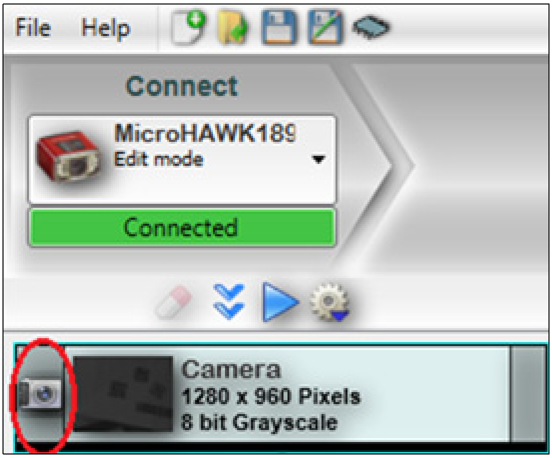
Each tool has data items linked to the PROFINET I/O structure as shown here:

| Tool Result | PLC/PNIO IN tag |
|---|---|
| Measurement Status | "USER".Demo.MeasureStatus (DB101.DBX 1001) |
| Measurement | "USER".Demo.ReadDistance (DB101.DBD 58) |
| Decode Status (matchcode) | "USER".Demo.DecodeStatus (DB101.DBX 100.2) |
| Decode | "USER".Demo.ReadString[32] (DB101.DBB 64-96) |
| Count Blob Status | "USER".Demo.CountBlobStatus (DB101.DBX 100.3) |
| Count Blob Count | "USER".Demo.ReadBlobCount (DB101.DBW 62) |

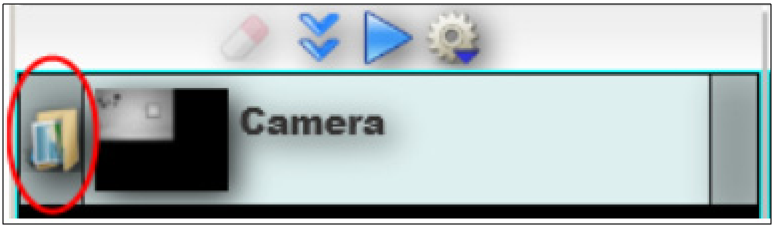
| Tool Parameter | PLC/PNIO OUT tag |
|---|---|
| Measurement Tolerance Low | "USER".Demo.MinDistance (DB101.DBD 46) |
| Measurement Tolerance High | "USER".Demo.MaxDistance (DB101.DBD 50) |
| Decode Matchcode | "USER".Demo.MatchCode [32] (DB101.DBB 12-44) |
| Count Blob Lower Tolerance | "USER".Demo.MinBlobCount (DB101.DBW 54) |
| Count Blob Upper Tolerance | "USER".Demo.MaxBlobCount (DB101.DBW 56) |

This data is transferred cyclically between the camera and PLC.

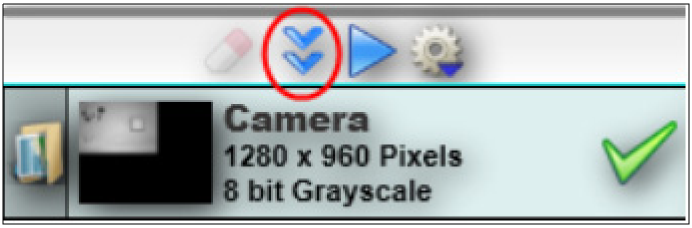
Once the job has loaded, the next step is to link the pre-saved images on the local PC. On the **Camera** button, click the far left icon to select and load an image.



A file browser will open. Then navigate to the same folder where the demo job was loaded PROFINET I/O Demo. If the images are located, the icon will change from a camera to a folder.




By clicking the **Try Job Once** icon, the application will cycle through the entire job with the loaded image.




There are two images on file. One will pass all three inspections and one will fail all three.

Now click the **Run** button on the top ribbon. This will download the job to the camera. At this point the job is ready to run and can be tested. However, in order to run the job change demo this job needs to be loaded into **Slot 1**. Click back to the **Edit** view from the top ribbon bar.

Click the slot icon and select Slot 1 or **New Slot** if no slots are currently in use.

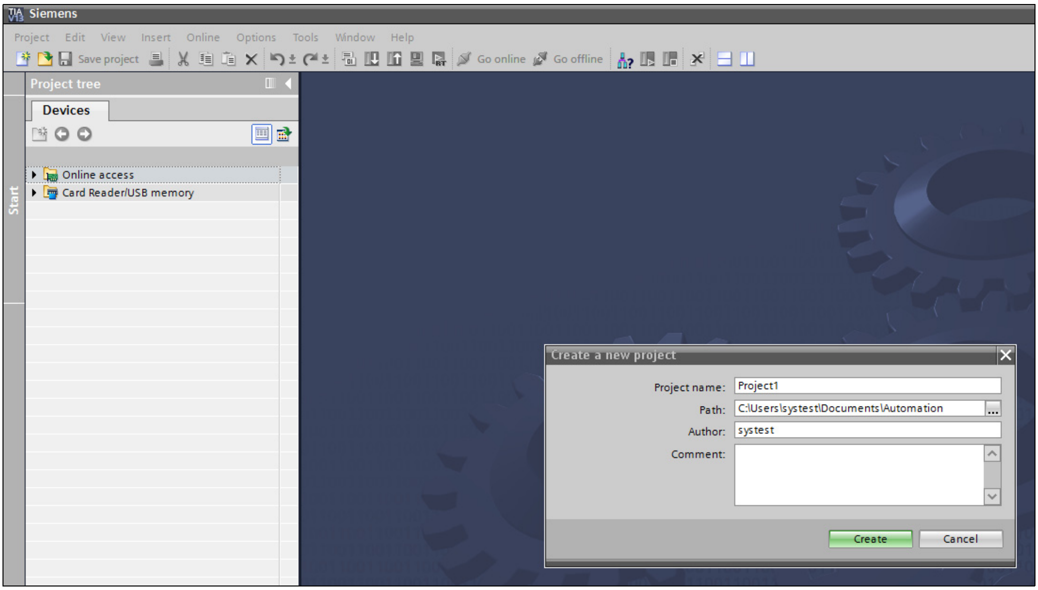

Now the job and images will be saved to the flash memory of the camera.

## TIA Portal V13 Setup

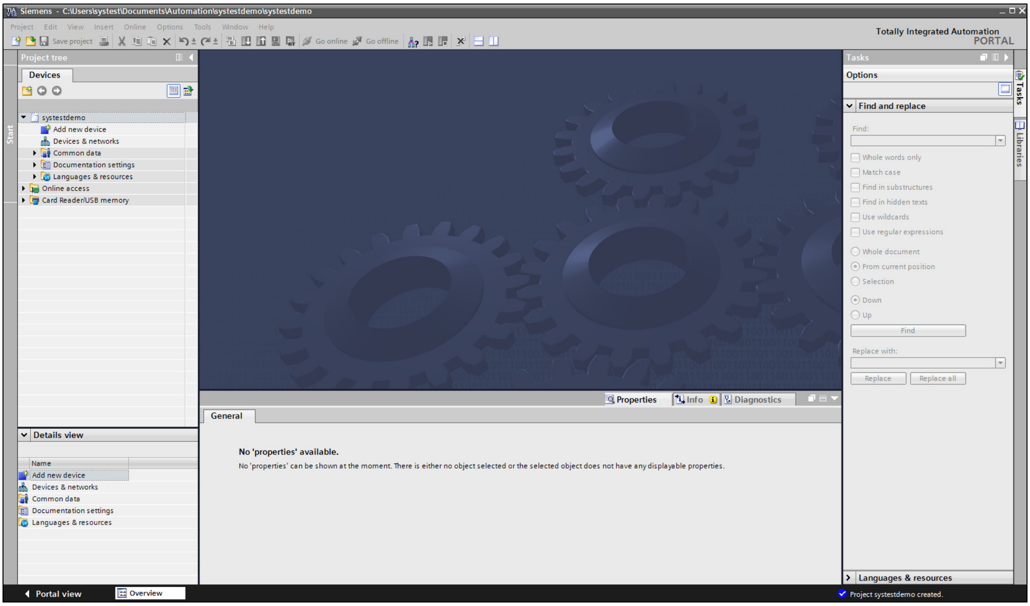
This section was created using Siemens TIA Portal software version 13 and an ET200SP CPU catalog number 6ES7 510-1DJ01-0AB0, HW version 2, FW version 1.8.2. This example shows the MicroHAWK device. However, the steps are similar for any device.

First launch the TIA Portal from the desktop.

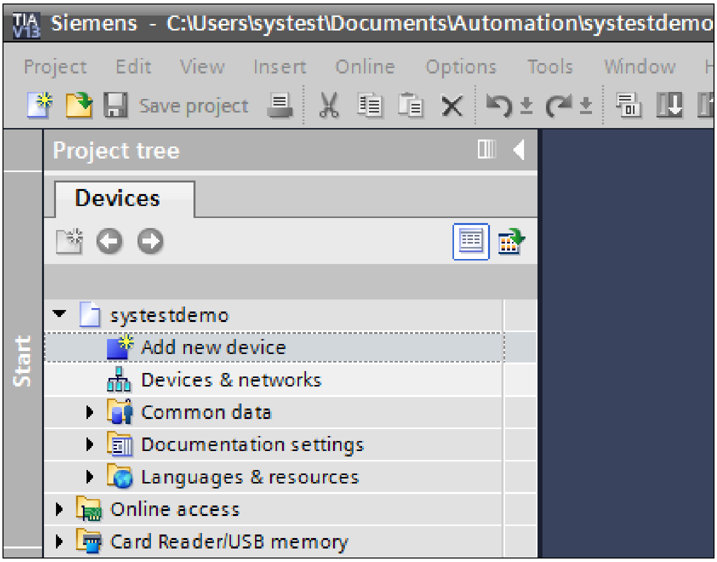

Create a new project from the menu item **Project->New**. The default name is Project1 as seen below. Fill out the required information, then click **Create**.



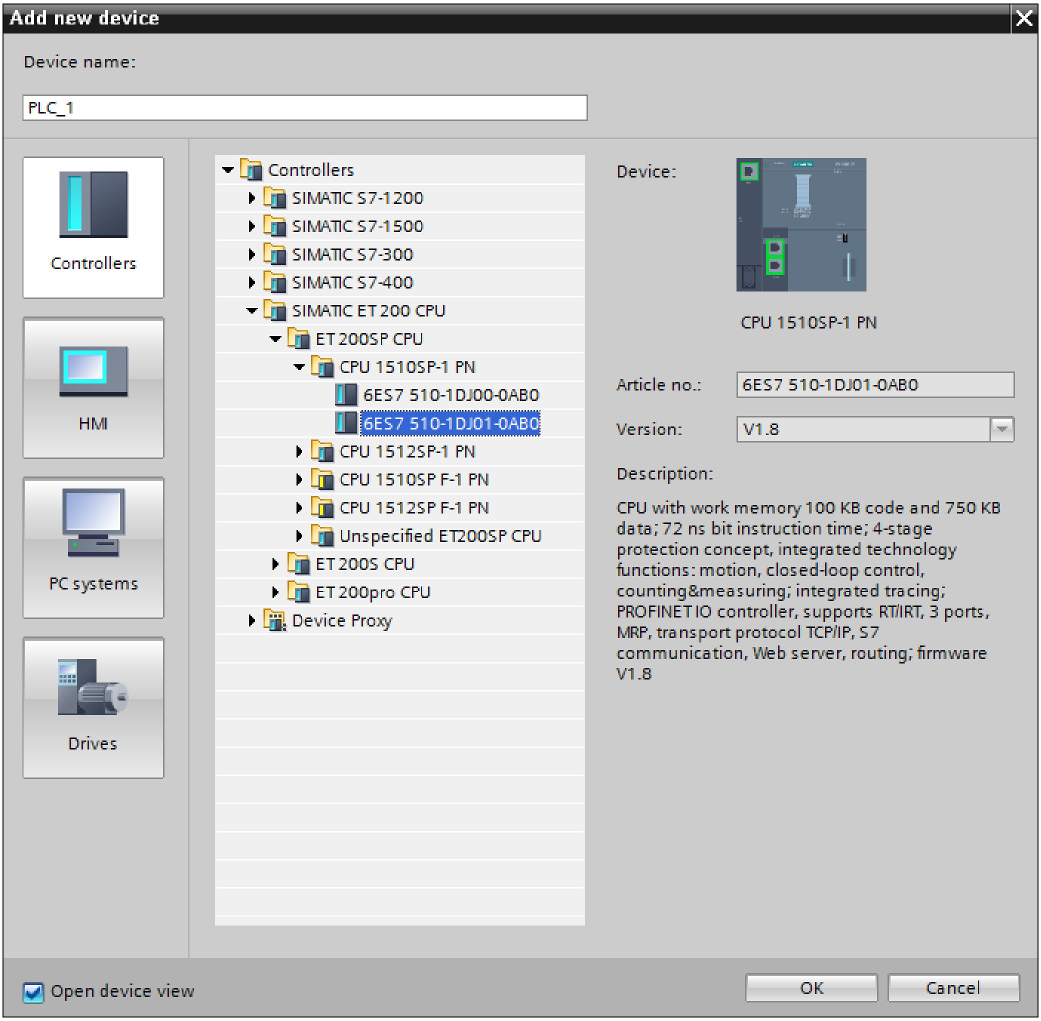
The screen below shows the main screen you will see when a project is opened.



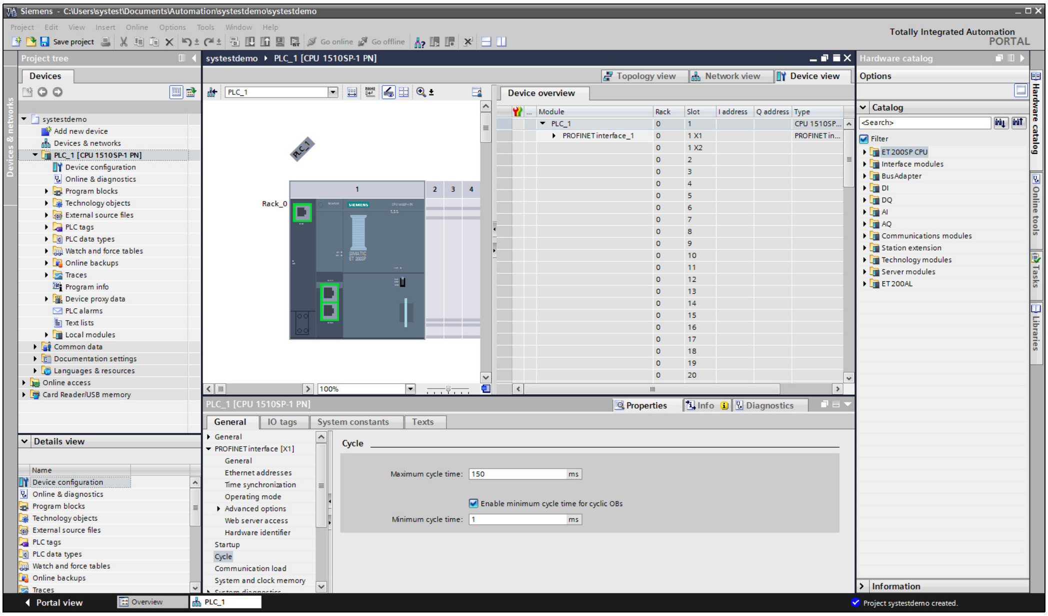
Double-click **Add new device**.



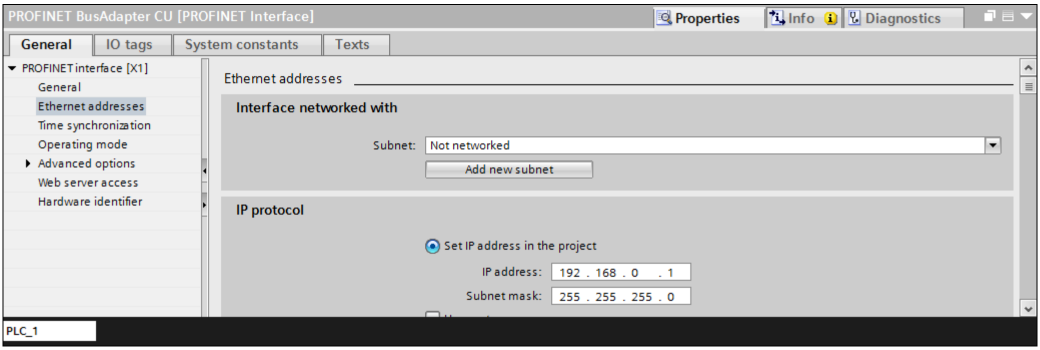
Open the **SIMATIC ET200 CPU** arrow, then open the **ET200SP CPU** arrow, then open the **CPU 1510SP-1 PN**, then click on **6ES7 510-1DJ01-0AB0**. You will see the screen below. Click **OK**.



You will see the following screen.

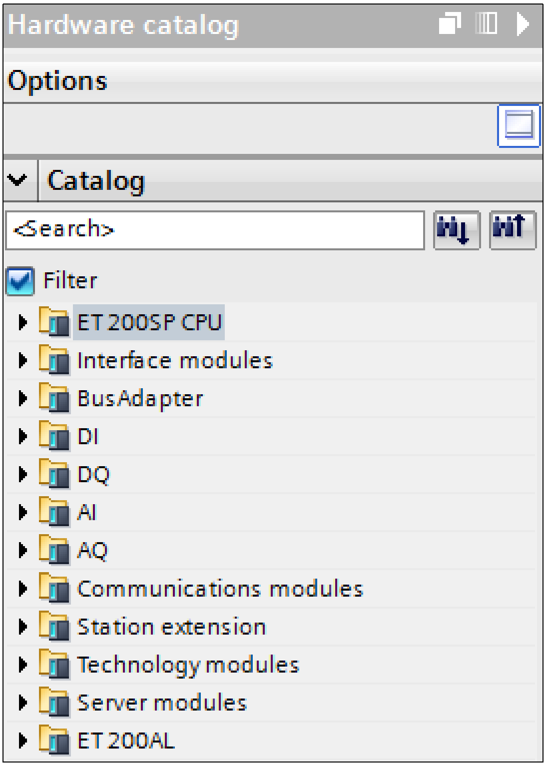



Click the lower green box on the PLC picture. The following will be displayed in the lower pane.




Click **Add new subnet**. The Subnet field will show PN/IE_1. Now go to the IP address field, and type the IP address you wish to use for this device. Click **Save Project** often to save your work.

To the right of the screen, you will see the following:

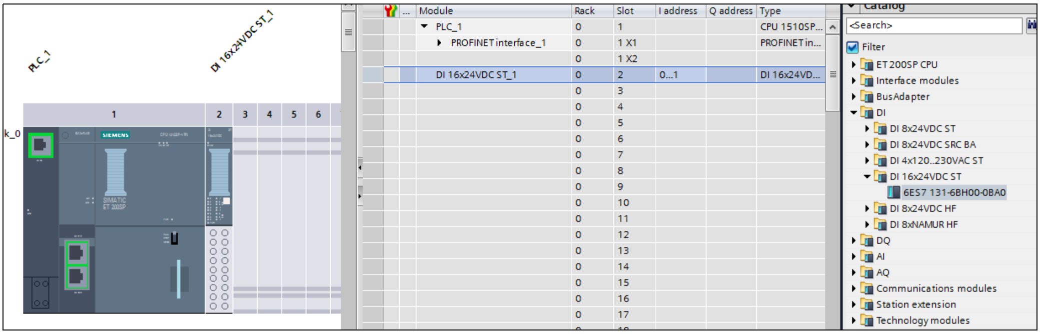



The modules in our test device are as follows:

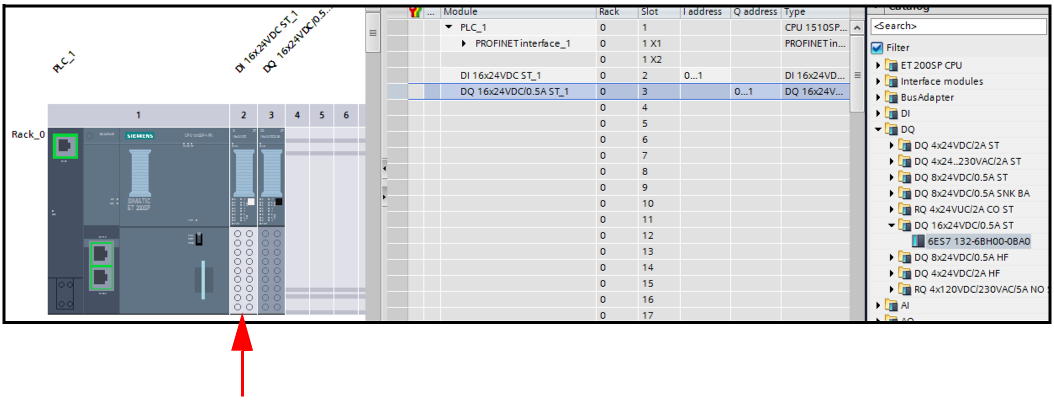
1. DI 16x24VDC ST order number 6ES7 131-6BH00-0BA0
2. DQ 16x24VDC ST order number 6ES7 132-6BH00-0BA0

To add these modules, first click the arrow next to **DI**, keep clicking on arrows until you find the order number for device 1. Click and hold down the left mouse key. Drag it to slot 2 just to the left.

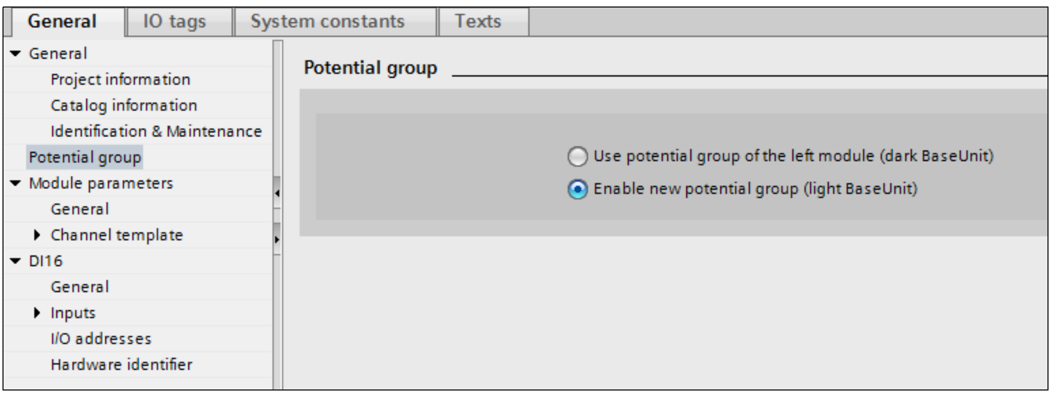
Your display should look like this:




Do the same thing for device 2, looking under the **DQ** arrow. Your display should now look like this:

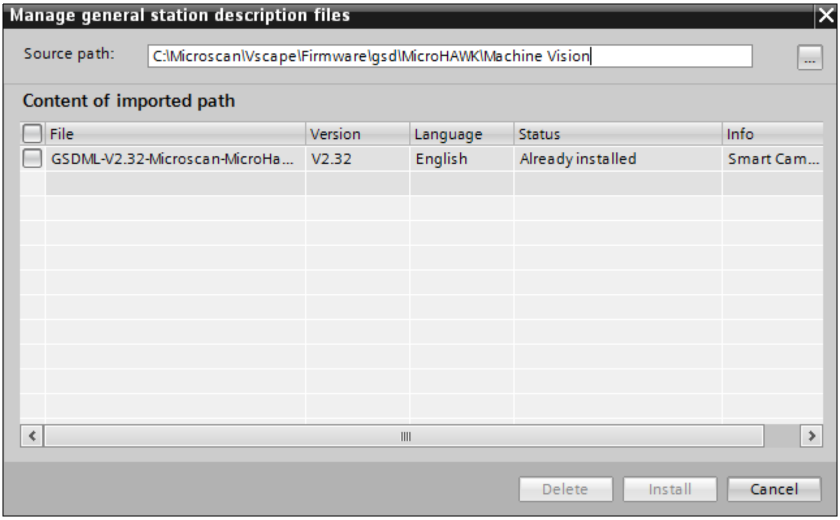



Now click on the PLC picture on slot 2. You will see the following screen:



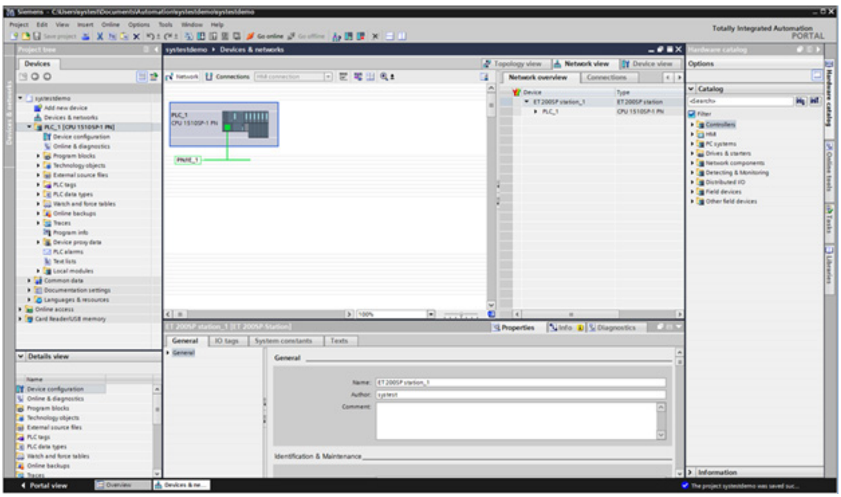
Be sure the **Enable new potential group (light Base Unit)** is selected. Do the same for Slot 3. Save the project.

Now click on **Options->Manage general station description files (GSD)**.

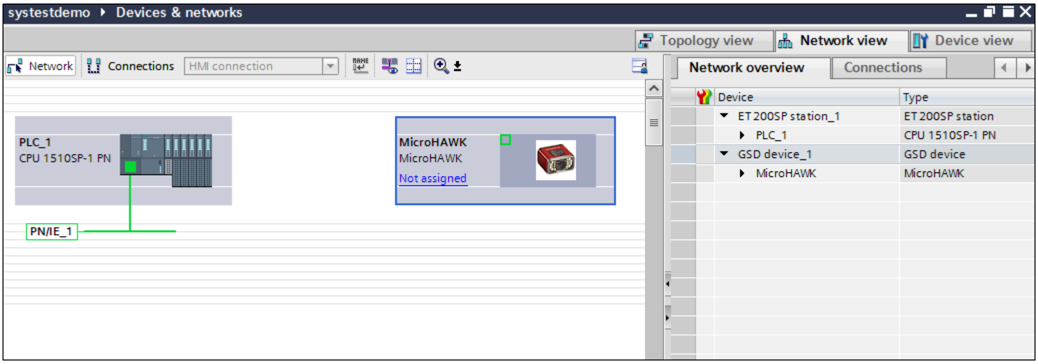



Browse to the following path: **C:\Microscan\Vscape\Firmware\gsd\MicroHAWK**. Select the appropriate GSD file by clicking the box next to the name, and click **Install**.

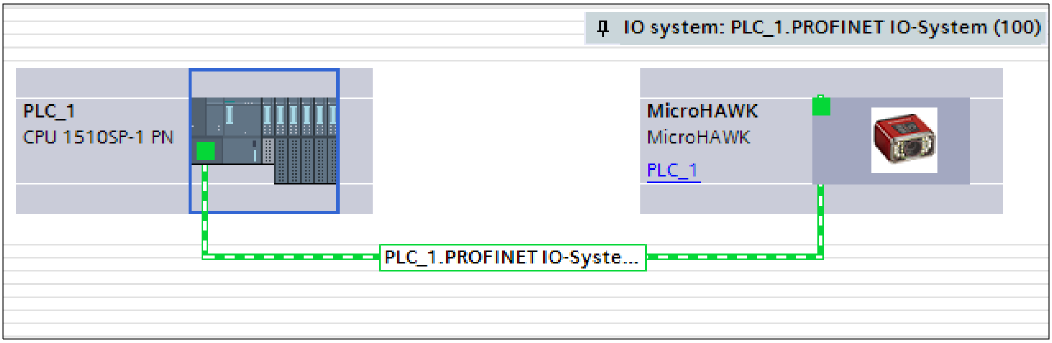
Click the tab that says **Network View**:



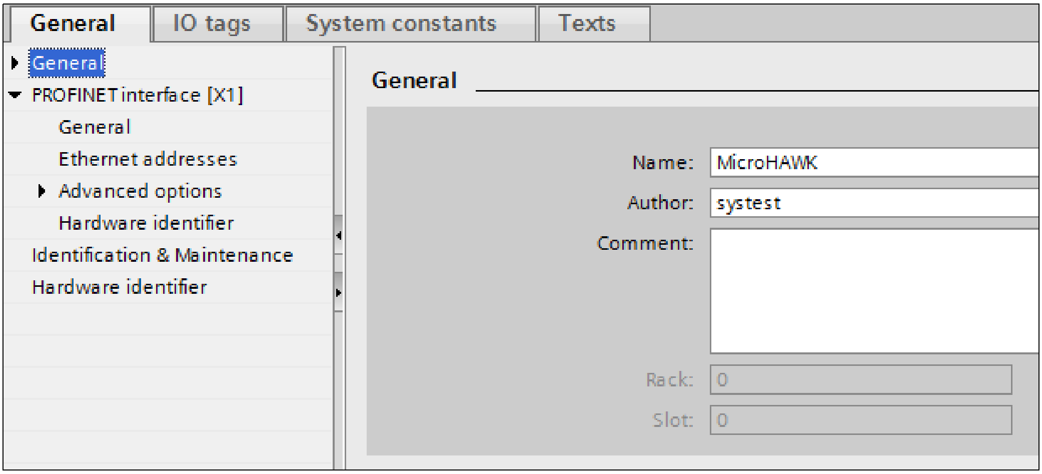
Look at the right panel and click **Hardware Catalog**. Use the arrows to open up the following area. **Other field devices->PROFINET IO ->General->MICROSCAN->Smart Camera->MicroHAWK MV-40**. Left-click on MicroHAWK MV-40 and hold it. Drag it to the left into the Devices and networks panel. Your screen will look like this:




On the MicroHAWK picture, click the **Not assigned** link, then click on **PLC_1.PROFINET IO-System**. This has just connected the MicroHAWK to the PROFINET network.



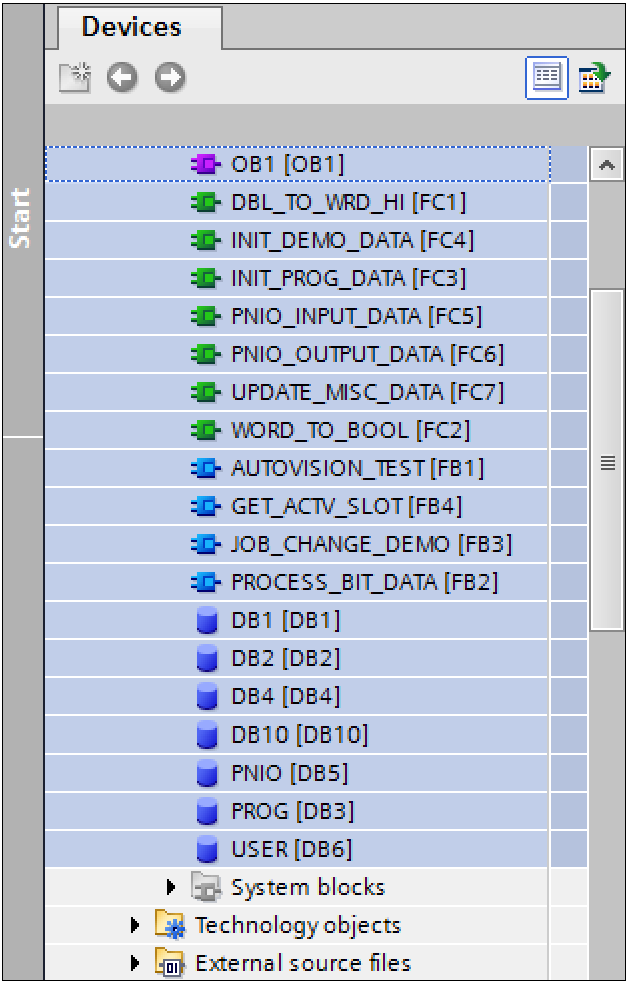
Double-click the picture of the MicroHAWK. In the bottom pane you will see the following:




Enter the name of the device in the Name: field. Then click Ethernet addresses on the left, and type in the IP address of the camera. This will allow the PLC to set the camera IP.

## Importing an Example Program

From TIA Portal, open the example program from the menu **Project > Open**. Click the **Browse** button to locate the (**MicroHAWK231.ap13**) program. The AVDemo is located in **C:\Microscan\Vscape\Tutorials and Samples\MicroHAWK\PROFINET Demo\TIAPortal_Demos\Legacy_AVDemo**. In the left pane in the **Devices** tab, open the PLC_1[CPU 1510SP-1 PN] arrow. Open the Program blocks arrow. Select all the blocks as shown below.

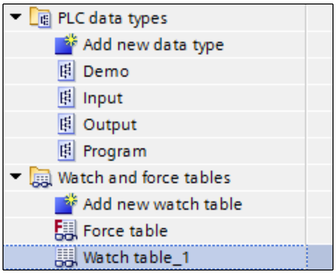


Right click and copy. Go to your new project and open the same arrows, right click on Program blocks, then click paste. Select Main[OB1] right click, and hit delete. Be sure to save your work.

Now back in the left pane of the AVDemo project, click on PLC data types, right-click, copy, and paste them into the new project PLC data types.

In the left pane of the AVDemo project, click on Watch and force tables. Right click on Watch table_1, then copy. Paste into new project Watch and force tables. Be sure to save the project.

Your data in the new project should look like this:




In the left pane on the AVDemo project open the PLC tags arrow. Double-click on Default tag table. You should see the following:

**Default tag table**

| | Name | Data type | Address | Retain | Visibl... | Acces... | Comment |
|---|---|---|---|---|---|---|---|
| 1 | Tag__635939309667999050 | Timer | %T10 | ☐ | ☑ | ☑ | |
| 2 | Tag__635939309669449133 | Timer | %T11 | ☐ | ☑ | ☑ | |
| 3 | Tag__635939309669679146 | Timer | %T12 | ☐ | ☑ | ☑ | |
| 4 | Tag__635939309669899159 | Timer | %T1 | ☐ | ☑ | ☑ | |
| 5 | Tag__635939309670609199 | Timer | %T3 | ☐ | ☑ | ☑ | |
| 6 | Tag__635939309670909216 | Timer | %T4 | ☐ | ☑ | ☑ | |
| 7 | Tag__635939309671339241 | Timer | %T5 | ☐ | ☑ | ☑ | |
| 8 | Tag__635939309671749264 | Timer | %T6 | ☐ | ☑ | ☑ | |
| 9 | Tag__635939309689560283 | Timer | %T20 | ☐ | ☑ | ☑ | |
| 10 | Tag__635939309689790296 | Timer | %T21 | ☐ | ☑ | ☑ | |
| 11 | Tag__635939309690120315 | Timer | %T22 | ☐ | ☑ | ☑ | |
| 12 | Tag__635939309690890359 | Timer | %T23 | ☐ | ☑ | ☑ | |
| 13 | Tag__635939309691150374 | Timer | %T29 | ☐ | ☑ | ☑ | |
| 14 | <Add new> | | | ☐ | ☑ | ☑ | |

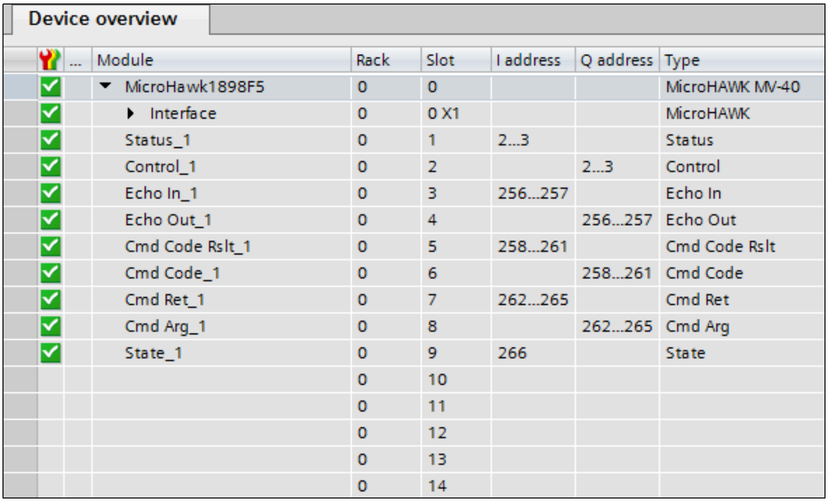
Type Ctrl-A to select all, and right-click copy. Open the new project PLC tags in the same way, and paste these in.

## Checking the Mapping of IO

Depending on the order the PLC can load things, we may need to change the IO addresses in our block to match the Device Overview in the Device view screen.

Double-click on the MicroHAWK picture, and the Device overview will be displayed to the right.

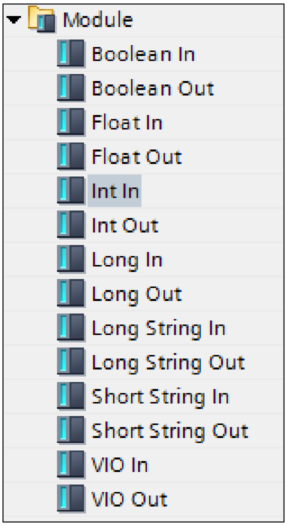
The image below is intended as an example only and may not exactly resemble your Device Overview.



Device overview

| Module | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|
| MicroHawk1898F5 | 0 | 0 | | | MicroHAWK MV-40 |
| Interface | 0 | 0 X1 | | | MicroHAWK |
| Status_1 | 0 | 1 | 2...3 | | Status |
| Control_1 | 0 | 2 | | 2...3 | Control |
| Echo In_1 | 0 | 3 | 256...257 | | Echo In |
| Echo Out_1 | 0 | 4 | | 256...257 | Echo Out |
| Cmd Code Rslt_1 | 0 | 5 | 258...261 | | Cmd Code Rslt |
| Cmd Code_1 | 0 | 6 | | 258...261 | Cmd Code |
| Cmd Ret_1 | 0 | 7 | 262...265 | | Cmd Ret |
| Cmd Arg_1 | 0 | 8 | | 262...265 | Cmd Arg |
| State_1 | 0 | 9 | 266 | | State |
| | 0 | 10 | | | |
| | 0 | 11 | | | |
| | 0 | 12 | | | |
| | 0 | 13 | | | |
| | 0 | 14 | | | |

From the Hardware catalog display to the far right, open the **Module** selection. You will see the following:



You must manually drag and drop each one of these in the Device Overview pane. Click on one of the entries and TIA will highlight the appropriate slot to drag it to. TIA will only allow you to insert entries into their proper slot position.

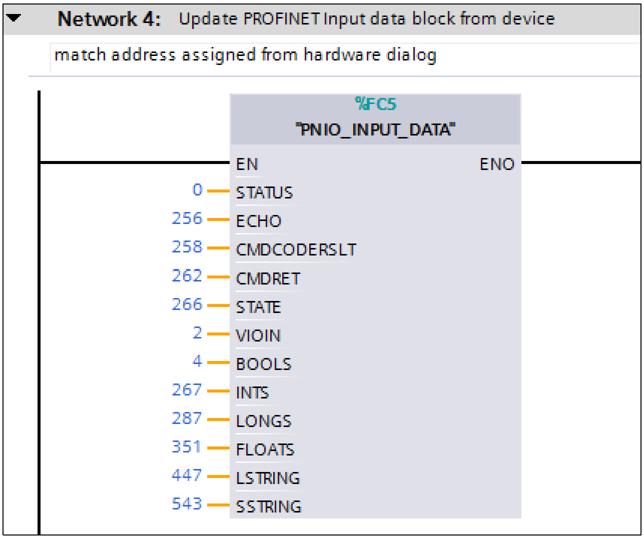
Your device view should now look as follows. The example addresses shown will not be the same for your device.

Device overview

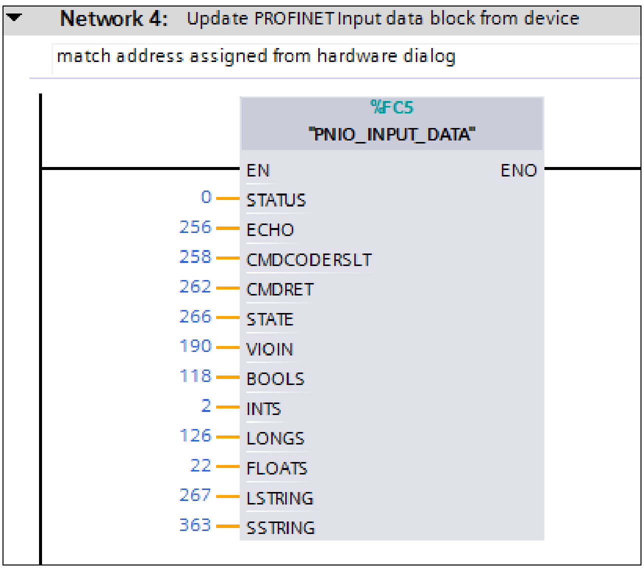
| Module | Rack | Slot | I address | Q address | Type |
|---|---|---|---|---|---|
| MicroHAWK | 0 | 0 | | | MicroHAWK MV-40 |
| Interface | 0 | 0 X1 | | | MicroHAWK |
| Status_1 | 0 | 1 | 0...1 | | Status |
| Control_1 | 0 | 2 | | 0...1 | Control |
| Echo In_1 | 0 | 3 | 256...257 | | Echo In |
| Echo Out_1 | 0 | 4 | | 256...257 | Echo Out |
| Cmd Code Rslt_1 | 0 | 5 | 258...261 | | Cmd Code Rslt |
| Cmd Code_1 | 0 | 6 | | 258...261 | Cmd Code |
| Cmd Ret_1 | 0 | 7 | 262...265 | | Cmd Ret |
| Cmd Arg_1 | 0 | 8 | | 262...265 | Cmd Arg |
| State_1 | 0 | 9 | 266 | | State |
| VIO Out_1 | 0 | 10 | | 190...191 | VIO Out |
| VIO In_1 | 0 | 11 | 190...191 | | VIO In |
| Boolean Out_1 | 0 | 12 | | 118...125 | Boolean Out |
| Boolean In_1 | 0 | 13 | 118...125 | | Boolean In |
| Int Out_1 | 0 | 14 | | 2...21 | Int Out |
| Int In_1 | 0 | 15 | 2...21 | | Int In |
| Long Out_1 | 0 | 16 | | 126...189 | Long Out |
| Long In_1 | 0 | 17 | 126...189 | | Long In |
| Float Out_1 | 0 | 18 | | 22...117 | Float Out |
| Float In_1 | 0 | 19 | 22...117 | | Float In |
| Long String Out_1 | 0 | 20 | | 266...361 | Long String Out |
| Long String In_1 | 0 | 21 | 267...362 | | Long String In |
| Short String Out_1 | 0 | 22 | | 362...553 | Short String Out |
| Short String In_1 | 0 | 23 | 363...554 | | Short String In |
| | 0 | 24 | | | |

You need to modify the PLC program blocks to agree with this display. On the left under the Devices pane, open the Program blocks arrow, and double-click on OB1 [OB1]. This will open a screen in the middle pane. Scroll this middle pane down until you see Network 4.

### Original Mapping

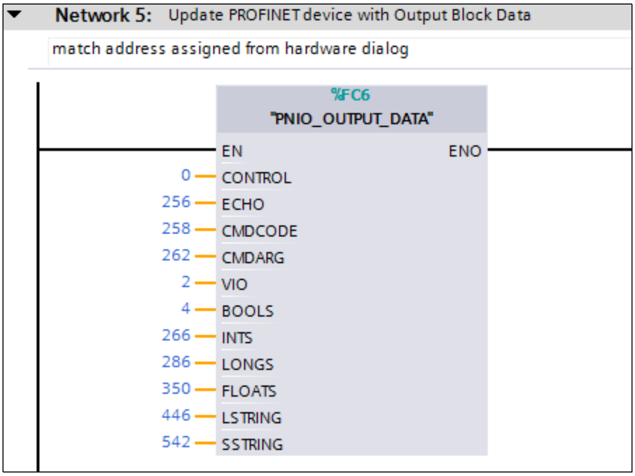



Change the numbers here to match with the Device overview display, as shown in the example below.

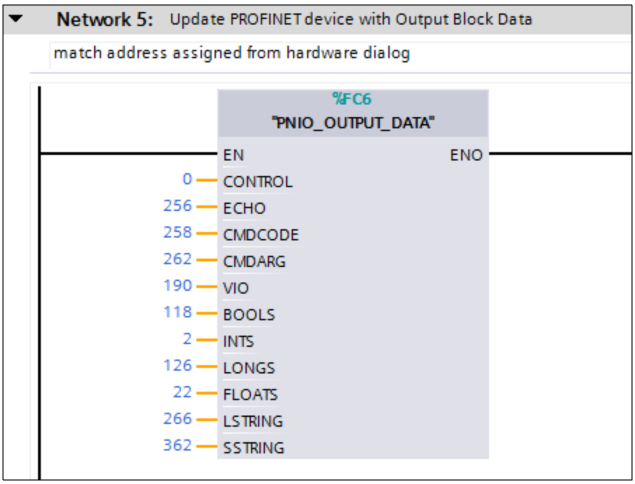


**Repeat this for Network 5.**

## Original Mapping

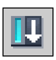



## Updated Mapping

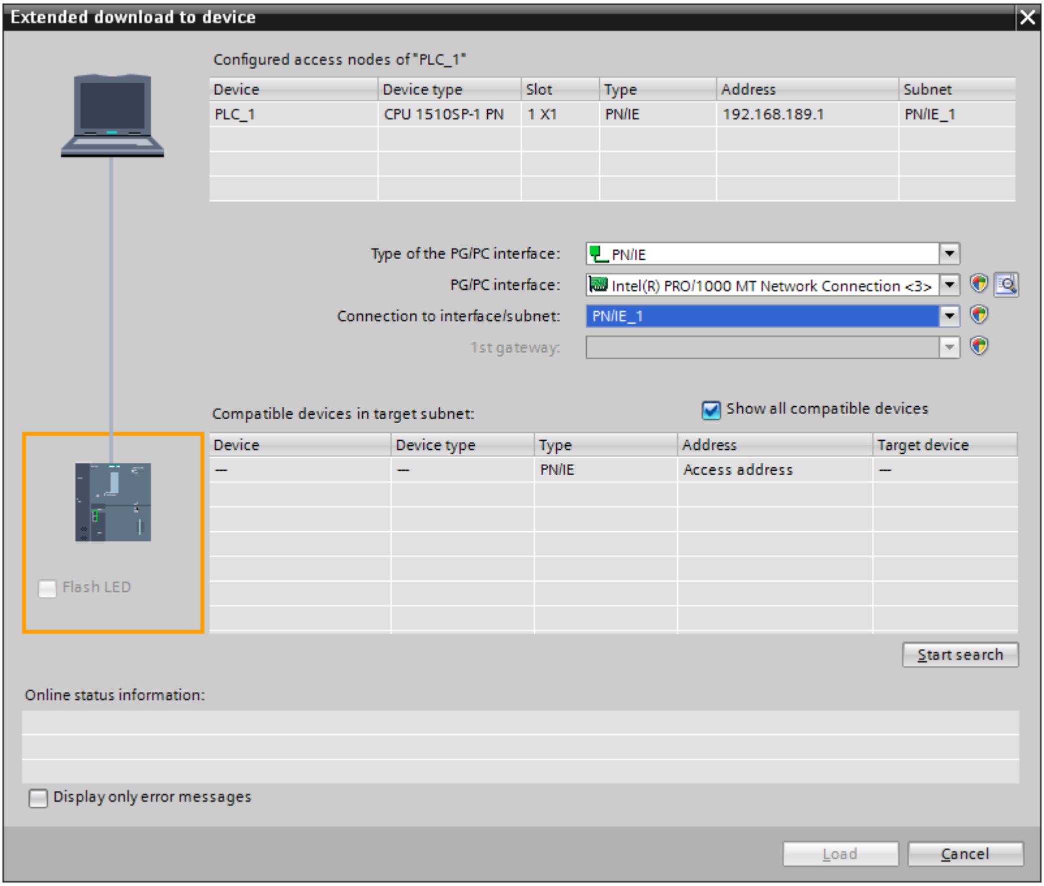



Be sure to save the project. Click the **PLC_1 {CPU 1510SP-1 PN]** top level, then click the compile icon.

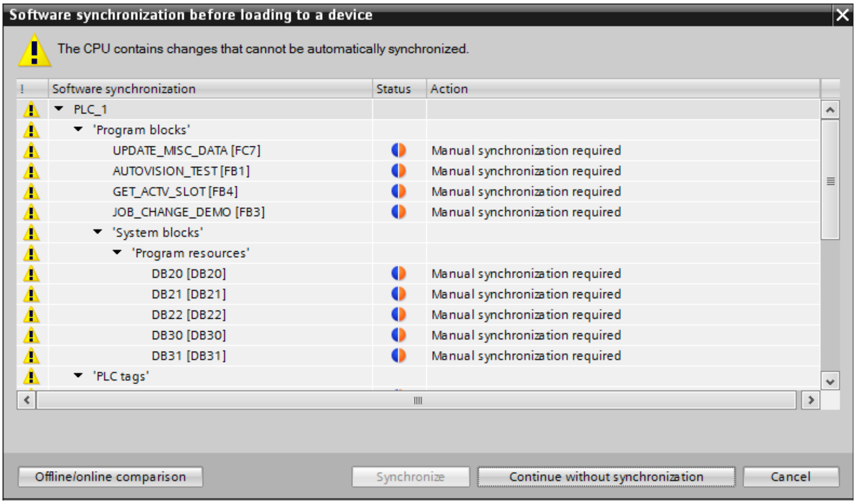
There should be NO compile errors. If there are no errors, click the download icon.



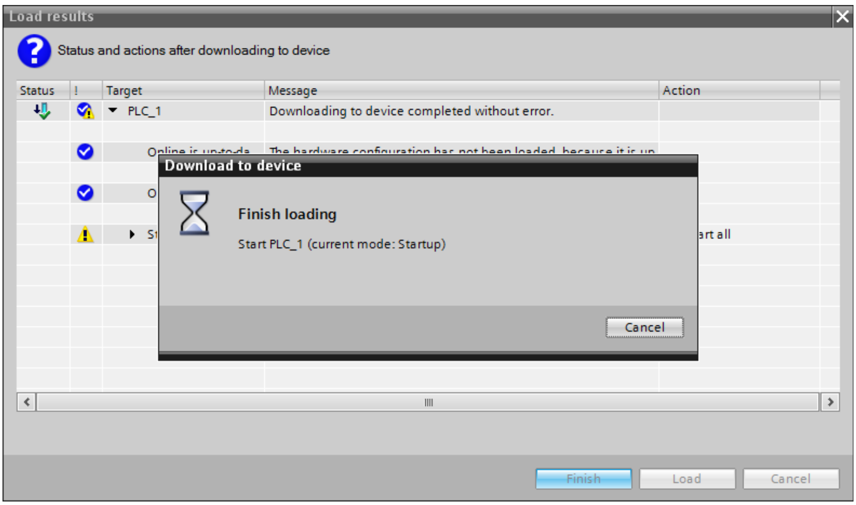
You will see the following screen:



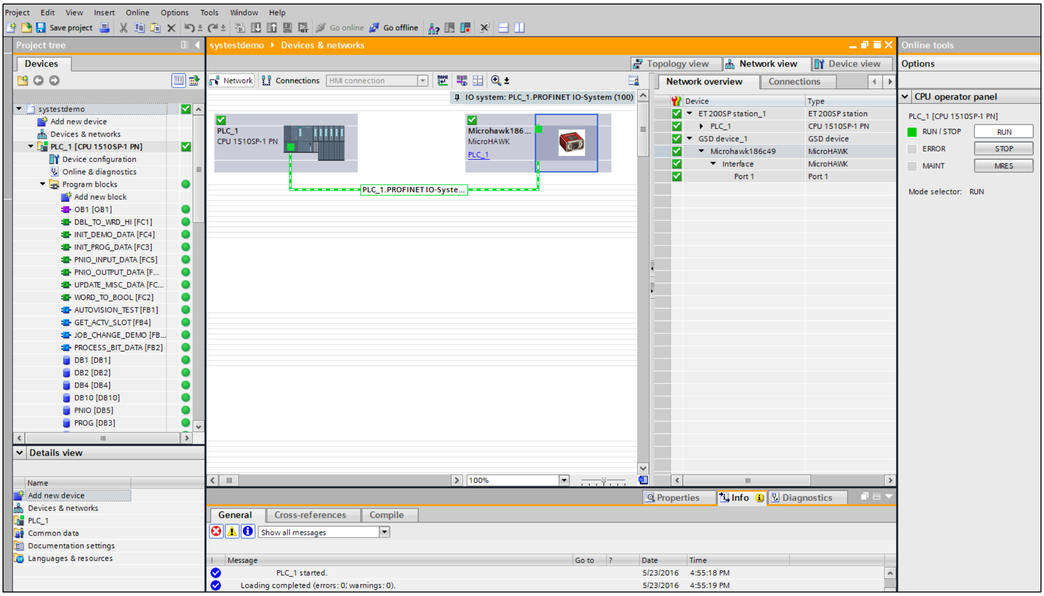
Select your PC network interface, and change the Connection to interface/subnet to **PN/IE_1**. Click **Start Search**. PLC_1 should be displayed with the address you gave it. Click **Load**. If you see a screen similar to below, do a memory reset (MRES) on the PLC, and try to load again. If this continues to be an issue, click **Continue without synchronization** and continue.




When downloading is working correctly you will see the following:



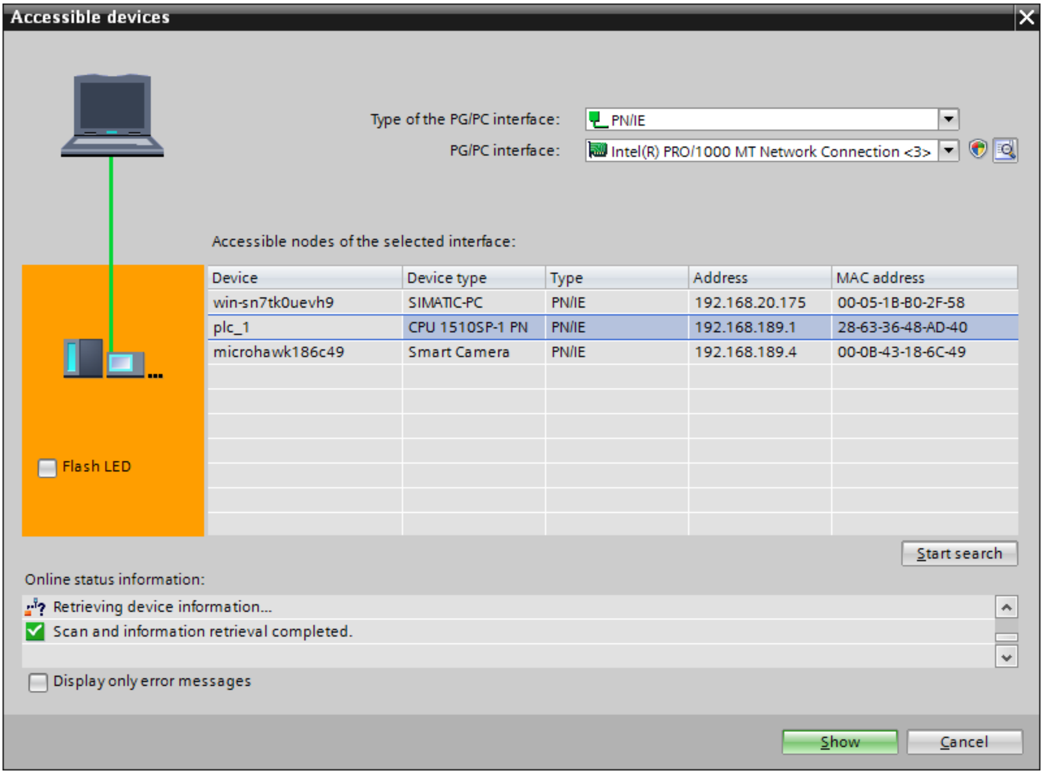
All LEDS on the PLC should be **GREEN**. Click **Go online**. You should see the following:



# Troubleshooting

If all LEDs are not green, go offline and click the Accessible devices icon.

Click **Start search**. The PLC and the MicroHAWK should be displayed here.




If the name of the MicroHAWK is not correct, you will not connect. The default name is microhawk with the last 3 of the MAC address. Ex: microhawk186c49. Go back to the general settings of the MicroHAWK in the device view, and change the name if required. Re-compile, and re-load. Try running again. This is the most common reason for not connecting.

## Running the Demo

In the left pane click the Watch and force tables arrow. Double-click Watch table_1. The watch table will open in the center pane. Click on the monitor variables icon.

| | | | | |
|---|---|---|---|---|
| 1 | "USER".Demo.DemoTrigType | %DB6.DBW0 | DEC | 0 |
| 2 | "USER".Demo.DemoMode | %DB6.DBW2 | DEC | 0 |
| 3 | "USER".Demo.ClearCounters | %DB6.DBX4.0 | Bool | FALSE |
| 4 | "USER".Demo.CountExecutions | %DB6.DBW6 | DEC | |
| 5 | "USER".Demo.CountPass | %DB6.DBW8 | DEC | |
| 6 | "USER".Demo.CountFail | %DB6.DBW10 | DEC | |
| 7 | "USER".Demo.MinDistance | %DB6.DBD46 | Floating-point nu... | 100.0 |
| 8 | "USER".Demo.MaxDistance | %DB6.DBD50 | Floating-point nu... | |
| 9 | "USER".Demo.MinBlobCount | %DB6.DBW54 | DEC | |
| 10 | "USER".Demo.MaxBlobCount | %DB6.DBW56 | DEC | |
| 11 | "USER".Demo.ReadDistance | %DB6.DBD58 | Floating-point nu... | |
| 12 | "USER".Demo.ReadBlobCount | %DB6.DBW62 | DEC | |
| 13 | "USER".Demo.MeasureStatus | %DB6.DBX100.1 | Bool | |
| 14 | "USER".Demo.DecodeStatus | %DB6.DBX100.2 | Bool | |
| 15 | "USER".Demo.CountBlobStatus | %DB6.DBX100.3 | Bool | |
| 16 | "PROG".Program.JobPass | %DB3.DBX24.2 | DEC | |
| 17 | "PROG".Program.ActiveJob | %DB3.DBW26 | DEC | |
| 18 | "PROG".Program.JobStatus | %DB3.DBW28 | DEC | |
| 19 | "PROG".Program.CurrentState | %DB3.DBW0 | DEC | |
| 20 | | | | |
| 21 | "PNIO".Input.CmdCodeRslt | %DB5.DBD4 | Hex | |
| 22 | "PNIO".Input.CmdRet | %DB5.DBD8 | Hex | |
| 23 | "PNIO".Output.CmdCode | %DB5.DBD464 | Hex | |
| 24 | "PNIO".Output.CMDArg | %DB5.DBD468 | Hex | |
| 25 | "PNIO".Output.Control | %DB5.DBW460 | Bin | |
| 26 | "PROG".Program.Status_ONLINE | %DB3.DBX20.0 | Bool | |
| 27 | "PROG".Program.Status_EXP_BUSY | %DB3.DBX20.1 | Bool | |
| 28 | "PROG".Program.Status_ACQ_BUSY | %DB3.DBX20.2 | Bool | |
| 29 | "PROG".Program.Status_TRIGGER_READY | %DB3.DBX20.3 | Bool | |

There are two demo modes in this example. The modes are set in:

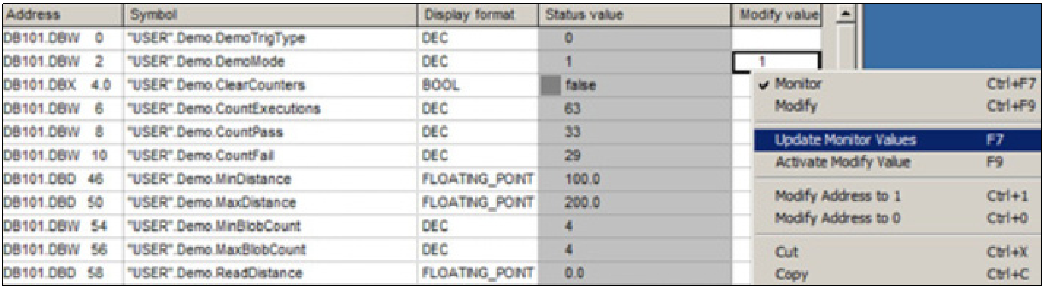
“USER”.Demo.DemoMode (DB101.DBW 2)

0 = AutoVISION Test

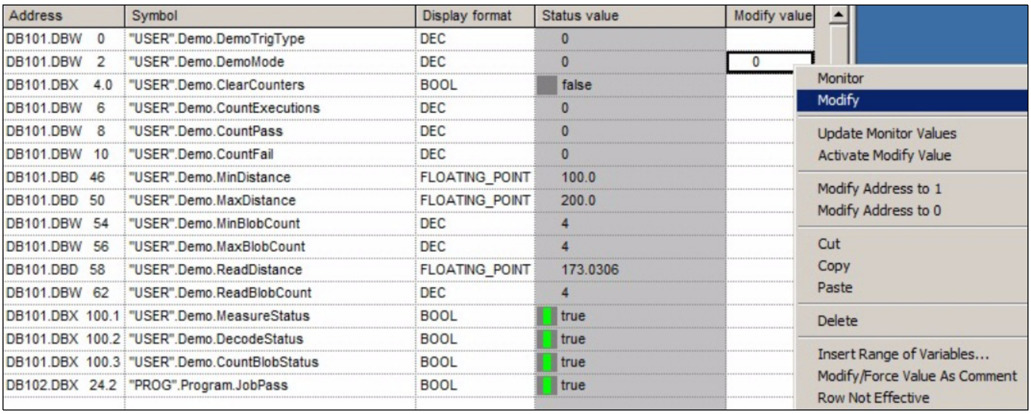
1 = Job Change Test

The AutoVISION Test will demonstrate communications between the PLC and the current job loaded from AutoVISION.

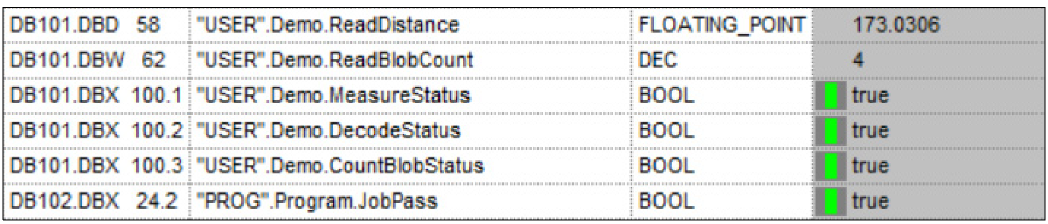
If **0** doesn't appear under the **Status Value** column, right-click in the **"USER".Demo.DemoMode** row under the **Modify Value** column. Select **Modify** or type **CTRL+F9** to change the value.




If **0** is not entered in the **Status Value** column of the **USER.Demo.DemoTrigType** row, enter **0**.To start triggering the program, right-click inside the **Status Value** column of the **USER.Demo.DemoTrigType (DB101.DBW 0)** row. Change the value from **0** to **1** to begin triggering and running the job on the camera.




When the program is triggering the camera, each cycle will produce either a pass or fail all state. Each state is counted in **"USER".Demo.CountPass**:



...and **"USER".Demo.CountFail**:

| | | | |
|---|---|---|---|
| DB101.DBD 58 | "USER".Demo.ReadDistance | FLOATING_POINT | 59.407 |
| DB101.DBW 62 | "USER".Demo.ReadBlobCount | DEC | 6 |
| DB101.DBX 100.1 | "USER".Demo.MeasureStatus | BOOL | false |
| DB101.DBX 100.2 | "USER".Demo.DecodeStatus | BOOL | false |
| DB101.DBX 100.3 | "USER".Demo.CountBlobStatus | BOOL | false |
| DB102.DBX 24.2 | "PROG".Program.JobPass | BOOL | false |

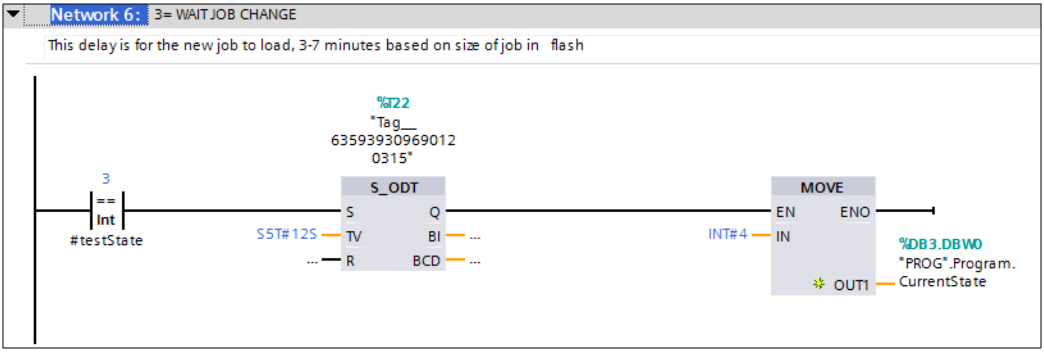
These counters can be cleared by right-clicking the row **"USER".Demo.ClearCounters** and selecting **Modify Address to 1**.

| | | | |
|---|---|---|---|
| DB101.DBX 4.0 | "USER".Demo.ClearCounters | BOOL | false |
| DB101.DBW 6 | "USER".Demo.CountExecutions | DEC | 104 |
| DB101.DBW 8 | "USER".Demo.CountPass | DEC | 52 |
| DB101.DBW 10 | "USER".Demo.CountFail | DEC | 49 |

The tolerances are below the counters. The inspection values and status bits are below the tolerances.

| | | | |
|---|---|---|---|
| DB101.DBD 46 | "USER".Demo.MinDistance | FLOATING_POINT | 100.0 |
| DB101.DBD 50 | "USER".Demo.MaxDistance | FLOATING_POINT | 200.0 |
| DB101.DBW 54 | "USER".Demo.MinBlobCount | DEC | 4 |
| DB101.DBW 56 | "USER".Demo.MaxBlobCount | DEC | 4 |

In some instances a large job could take longer to load. Increment the delay time to adjust for job size. The image below shows **Function Block 3**, **Network 6**, where the delay is located.



To run the **Job Change Test** you first need to use AutoVISION to download another job to slot 2 on the camera. In AutoVISION, connect to the camera and select the **Connect** icon at the top of the user interface. Click the **Load a Job** icon. Locate **Legacy_AVDemo.avp** and click the **Open** button. Once the job has loaded into AutoVISION, click the **Edit** button at the top of the user interface. Then click **Save the Job to a Slot** on the camera icon in the menu at the top of the user interface. Select **New Slot**, which should be **2**. When switching between tests, disable the current routine by setting the **"USER".Demo.DemoTrigType** to **0**. Now there should be two jobs loaded in the camera's flash memory.

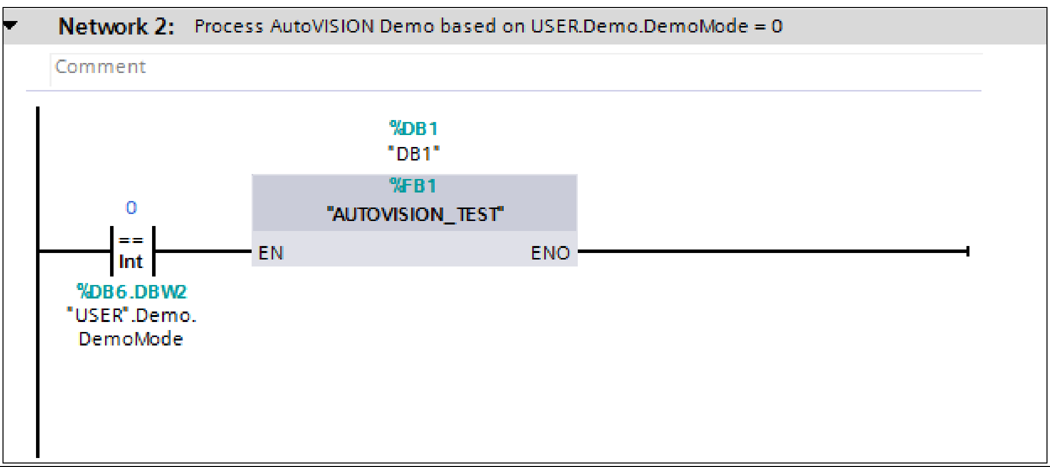
In the left pane of TIA Portal, click on the Watch and force tables arrow. Double-click on Watch table_1. The watch table will open in the center pane. Click on the monitor variables icon.

**AutoVISION must be closed for this demo to work.**

On the row **"USER".Demo.DemoMode**, right-click and modify to one. Right-click the row **"USER".Demo.DemoTrigType** and **Modify Address** to **1** to start the demo.

This demo will cycle through loading jobs from slots **1** and **2**. Each cycle will be counted in the variable table **"USER".Demo.CountExecutions**. The job load success is determined by the camera's status register **"Legacy_AVDemo.Input.CmdCodeRslt" = 0x0**. If the job load is successful the counter **"USER".Demo.CountPass** will increment. If the job load fails when **"Legacy_AVDemo".Input.CmdCodeRslt ! = 0x0**, then the counter **"USER".Demo.CountFail** will increment. **OB1** calls **FB1** to process the AutoVISION test. To view the ladder logic, double-click **AUTOVISION_TEST[ FB1]**. This will open the block in the center pane.

**OB1 Network 2**, showing **0** to run **AUTOVISION_TEST**.



A

APPENDIX A

# Serial Commands

**Serial commands** can be sent via **TCP port**, **AutoVISION Terminal**, or **HyperTerminal**.

**Important:** The HAWK MV-4000 Smart Camera does not support any focus commands.

# Serial Commands

## Serial Command Syntax

**< >** = Required argument. Replace appropriately.

For example:

**-u <DB_User_name>** becomes **-u av** where **av** replaces **DB_User_name**.

**|** = Mutually exclusive arguments. Choose one from the list.

**{ }** = Used with **|** to specify a list of choices for an argument.

**[ ]** = Optional parameter.

**Important:** Unless otherwise stated, commands will respond with **!OK** on success and **!ERROR** on failure.

## AUTOCAL [-exp={0|1}] [-expval={60-100000}] [-gain={0|1}] [-gainval ={0-100}] [-focus={0|1}] [-focval{0-9999}]

Initiates camera calibration of gain, exposure, and focus. Each parameter is independent. Ranges are device-dependent.

**-exp enable=1** or **disable=0** autocalibrate exposure.

**-expval** value of exposure in µs.

**-gain enable=1** or **disable=0** autocalibrate gain.

**-gainval** value of gain in percentage.

**-focus enable=1** or **disable=0** autocalibrate focus

**-focval** value of focus in mm.

***Example 1:***

**Command:** AUTOCAL

**Response:** 0;4632;134;50;300 (gain=0, exposure=4632 µs, focus=134, min allowable focus=50, max allowable focus=300)

***Example 2:***

**Command:** OFFLINE

**Response:** !OK

**Command:** QUERYAUTOCAL

**Response:** 0;4632;134;50;300 (Gain was 0.)

**Command:** AUTOCAL -exp=1 -gain=0 -gainval=18 (Fixed gain at 18%.)

**Response:** 18;3308;128;50;300 (Gain did not change and exposure has changed from 4632 µs to 3308 µs.)

***Example 3:***

**Command:** OFFLINE

**Response:** !OK

**Command:** QUERYAUTOCAL

**Response:** 0;3478;226;50;300 (Exposure was 3478 µs.)

**Command:** AUTOCAL -exp=0 -expval=1000 (Fixed exposure at 1000 µs.)

**Response:** 31;1000;98;50;300 (Exposure stayed at 1000 µs and gain has changed from 0% to 31%.)

**Important:** AUTOCAL only functions when the camera is OFFLINE.

## GET {tagname|service|service.tagname}

Gets value of a global tag.

The tagname must correspond to one of the supported tags within the device. Use the **INFO** command to get a full list of tags and services, as well as attributes of the tag and list of subtags.

The command is terminated by a carriage return and/or line feed character.

Include an index to get a single value from an array such as **GET int1**. If the index is omitted, the full array of values will be returned in a comma-separated list of values.

Send **Get {tagname|service.tagname|service}** to get the value of a tag within the global data service. To get the value of a tag within another service, prefix the tagname with the service name. For example, a **GET <service.tagname>** command such as **GET eip.input** for the EIP input assembly.

The AVP service allows retrieval of step and datum information from the job tree using forward slash '/' in the symbolic name path. **GET avp/insp1/snapshot1/status** paths are not case-sensitive and do not need to be fully qualified if unique.

**GET avp/snapshot1/status** will return the same result if there is only one inspection.

When issued against a step, **GET avp/snapshot1** will return the values for all datums.

**Success Return:** On success will return the value stored in the tag.

For example: ABCD

**Fail Return:** On failure will return !ERROR followed by the reason for the failure.

For example: !ERROR Tag matchstring66 not found

**Important:** This command only functions when the camera is ONLINE.

## GETIMAGE <-transfer=ymodem> [-format={jpg|png|tif|raw|bmp}] [-quality ={0-100}] [-woi=left,top,right,bottom] [-inspection=n]

Initiates serial transfer of inspection image.

**Note:** This command always returns the last (most recent) image.

**-transfer=ymodem** uses Ymodem protocol over the serial port. If the -transfer option is omitted completely, the transfer mode is over the TCP and Ethernet port.

**Important:** YModem transfer option is not supported on the HAWK MV-4000.

**-format={jpg|png|raw|tif|bmp}** specifies the format of the image. If omitted, the image format is JPG.

**Note:** For monochrome cameras, the only formats available are TIF, PNG, and JPG. For color cameras, the only formats available are RAW, PNG, and BMP.

**Note:** If BMP is selected, the system will return a PNG file.

**Note:** All image file types return complete file information that can be saved directly to disk except the RAW file type, which requires explicit conversion.

**-quality=*n*** specifies a JPG compression quality of *n* less than or equal to 100. The default quality is 80 if not specified. This setting is only supported for the JPG file type.

**Note: PNG**, **RAW**, **TIF**, and **BMP** formats provide lossless image compression. If **format** is set to **PNG**, **RAW**, **TIF**, or **BMP**, the **quality** setting does not apply.

**woi=left,top,right,bottom** specifies a rectangular area of the image to be included in the output image. If omitted, the full image buffer is returned.

**-inspection=*n*** specifies the inspection from which to retrieve an image. The image will be from the first snapshot within that inspection. If not specified, the image will be from the first inspection that contains a snapshot.

The following example will retrieve an image from the camera with these settings: **Protocol:** ymodem; **Format:** png; **Quality:** N/A; **Inspection:** second inspection.

**GETIMAGE –transfer=ymodem –format=png –inspection=2**

The following example will retrieve an image from the camera with these settings: **Protocol:** ymodem; **Format:** jpg (default); **Quality:** 50; **Inspection:** first inspection (default).

**GETIMAGE –transfer=ymodem –quality=50**

**Important:** This command only functions when the camera is ONLINE.

## HELP

Returns a list of all serial commands showing correct syntax and functionality descriptions.

## INFO [tagname|service]

Gets information about a tag or service.

**INFO** with no arguments gets a list of services.

**INFO <service>** gets a list of tags in that service.

**INFO <service.tagname>** gets attributes of the tag as well as a list of subtags.

The AVP service allows retrieval of step and datum information from the job tree using forward slash '/' in the symbolic name path. **INFO avp/insp1/snapshot1/status** paths are not case-sensitive and do not need to be fully qualified if unique.

**INFO avp/snapshot1/status** will return the same result if there is only one inspection.

When issued against a step, **INFO avp/snapshot1** returns properties of the step, a list of child datums, and a list of child steps. Child steps are indicated by a trailing forward slash.

## JOBBOOT [-slot=]<n>

Sets bootup job slot ***n*** (RS-232 only).

## JOBDELETE {[-slot=]n|-all}

Deletes job in slot ***n***, or all jobs if **-all**.

**Important:** Does not delete the current job loaded in camera memory.

## JOBDOWNLOAD <-transfer={ymodem|ftp}> [-size=value] [-c]

**Important:** JOBDOWNLOAD only supports FTP on the HAWK MV-4000.

Downloads a **.avz** job file via the specified transfer method (ymodem supported only over RS-232; FTP supported only over network connection).

The **ymodem transfer method** only requires that the user send the .avz file via the ymodem protocol over RS-232, and the job will load automatically after the transfer is complete.

The **FTP transfer method** requires the user to perform the following steps to load the job:

- **JOBDOWNLOAD: -transfer=ftp [-size=avpsizeinbytes]**

Pre-creates a fixed-size /streamd0 RAMdisk to receive the .avz over FTP. If size is omitted, the default RAM disk size is used to create /streamd0. The size of /streamd0 is limited to (available contiguous RAM – minimum target contiguous RAM) / 2.

- User FTPs the job to /streamd0
- **JOBLOAD: -mem -r**

Loads .avz from /streamd0 into RAM, deletes the RAMDisk /streamd0, and optionally starts the job (if -r is specified).

## JOBINFO [[-slot=]n] [-v]

Gets job summary or info about slot ***n***.

**JOBINFO** with no arguments returns a list of all jobs on the device.

**-v** = Verbose ***n***. This option shows the amount of space that would be freed if the job were deleted. It also lists the total disk space and free disk space.

## JOBLOAD {[-slot=]<n>|-mem} [-r]

Loads a job from slot *n* or from memory when used with the JOBDOWNLOAD command via FTP.

**-r** = Start inspections.

## JOBSAVE [-slot=]<n>

Saves current job to slot ***n.***

## MEMAVAIL [-cp]

Returns available memory for device or coprocessor.

## MEMCONTIG [-cp]

Returns maximum memory block for device or coprocessor.

## MEMFRAGS [-cp]

Returns memory fragments for device or coprocessor.

**Important:** MEMFRAGS is not supported by the HAWK MV-4000. It will return **!ERROR**.

## MEMINFO [-cp] [-v]

Returns memory summary “avail/contig/frags” for device or coprocessor. Verbose.

## OFFLINE

Stops all inspections.

## ONLINE

Starts all inspections.

## ONLINE?

Queries the camera if any job is online. If the camera is running in a multi-inspection job, this command will return **1** if all inspections are online and **0** otherwise.

## QUERYAUTOCAL

Returns photometry settings: Gain, Exposure, and Focus.

## QUERYFOCUSUNITS

Queries the units being used for autofocus, mm (0) or inches (1).

## QUERYWHITEBAL

Returns white balance settings: RED gain, BLUE gain, and GREEN gain.

## QUICKFOCUS [x] [y]

Performs an autofocus by analyzing the area around the point specified by **x** and **y**.

The response is in the format of the camera’s current focus, min. allowable focus on the camera, max. allowable focus on the camera.

***Example:***

Perform a quick focus on point (640,480) in the image.

**Command:** QUICKFOCUS 640 480

**Response:** 124;50;300 (Current focus is set to 124 mm with an allowable focus range of 50 – 300 mm on the current camera.)

**Important:** This command only functions when the camera is OFFLINE.

## READY? [-insp=n]

Queries if inspection is waiting for a trigger. **!1** if all inspections are ready or **!0** if not all inspections are ready.

**-insp=n** specifies the inspection to query if it is ready.

## REBOOT [-noload]

Reboots the device.

-noload = do not load BOOT job.

## RESTOREWBAL

Restores preset white balance parameters:  RED gain, BLUE gain, and GREEN gain.

## SET <tagname> <value>

Sets value of a global tag.

The tagname must correspond to one of the supported tags within the device. Use the **INFO** command to get a full list of tags and services, as well as attributes of the tag and list of subtags.

The value can contain spaces.

The command is terminated by a carriage return and/or line feed character.

The value can be a list of comma-separated items to set a sequence of tags:

Send **SET int1 1, 2, 3** to set int1 = 1, int2 = 2, int3 = 3.

The AVP service allows setting of step and datum information from the job tree using forward slash '/' in the symbolic name path. **SET avp/insp1/snapshot1/acq1/gain 2.0** paths are not case-sensitive and do not need to be fully qualified if unique.

**SET avp/acq1/gain 2.0** will set the same gain value if there is only one acquire.

Control tags in the AVP service such as **START**, **STOP**, and **TRIGGER** act as momentary switches. **SET avp.start 1** is equivalent to the **ONLINE** command. **avp.start** will reset immediately and always read as **0**.

**Success Return:** On success will return **!OK** followed by an echo of the command.

For example:

!OK SET matchstring1

**Fail Return:** On failure will return **!ERROR** followed by the reason for the failure.

For example:

!ERROR Tag matchstring66 not found

## SETFOCUSUNITS

Sets units used for autofocus, **mm (0)** or **inches (1)**.

**Important:** The MicroHAWK MV-40 only supports mm so SETFOCUSUNITS will only accept **0** and anything else will respond with **!ERROR**.

## TARGET {0|1|off|on}

Turns targeting LEDs On or Off.

**target 1** = Turn Target On

**target 0** = Turn Target Off

## TRIGGER

Triggers an inspection.

## VERSION

Returns Visionscape software version.

## vt [n]

Triggers an inspection by pulsing a Virtual I/O point.

For example: **vt 1**

will return pulse **VIO1**. The inspection will run if it is configured to use **VIO 1** as a trigger.

If specified, the VIO index must be in the allowed range for Virtual I/O points within Visionscape. The virtual I/O line will be set high then low.

If VIO Index is not specified, VIO1 is assumed.

**Fail Return:** Return **!ERROR** followed by the reason for the failure.

For example: **!ERROR No such trigger** when the index specified ‘**n**’ is out of range of virtual triggers.

## WHITEBAL

Performs automatic calibration of white balance settings: RED gain, BLUE gain, and GREEN gain.

**Important:** This command only functions when the camera is OFFLINE.