BANNER ENGINEERING

# PresencePLUS

## Industrial Ethernet User's Guide

Volume 1 (OLD firmware)

**4/29/2015**

**NOTE:** This guide is for use with PresencePLUS vision sensors with firmware older than 2.7.0 (found in the 2011R1 Software/Firmware release). This includes the all of the original PresencePLUS Pro models (PPCTL and PPCAM). Please see "PresencePLUS Industrial Ethernet User's Guide: Volume 2" for information relevant to vision sensors with firmware equal to 2.7.5 or later (found in the 2014R1B Software/Firmware release).

*An aid for use in establishing Ethernet communications between PresencePLUS sensors and PLCs or HMI.*

# Table of Contents

# PresencePLUS: Volume 1

**NOTE:** This guide is for use with PresencePLUS vision sensors with firmware older than 2.7.0 (found in the 2011R1 Software/Firmware release). This includes the all of the original PresencePLUS Pro models (PPCTL and PPCAM). Please see "PresencePLUS Industrial Ethernet User's Guide: Volume 2" for information relevant to vision sensors with firmware equal to 2.7.5 or later (found in the 2014R1B Software/Firmware release).

## Chapter 1: Basics of Operation

The PresencePLUS vision sensor is controlled via EtherNet/IP or Modbus/TCP through the input and output data it makes available as a slave device for these protocols. Nothing need be done on the vision sensor side to make inspection-independent data and basic control available to the PLC or HMI.

Using these input and output values, the following sensor operations can be performed:

- Trigger an inspection
- Remote Teach (to learn new Pass/Fail requirements for an inspection)
- Product Change (to switch between stored inspection files on the sensor)
- Monitor Status Indicators (pass, fail, ready, error)
- Check Counters (pass, fail, system error, missed trigger, frame count, iteration count)
- Change Compare String (in the case of BCR or OCR/OCV a new matching ASCII string can be sent to the vision sensor for 'on-the-fly' changes to the inspection pass/fail requirements)
- Inspection Specific Data (see Chapter 2 for more information)

### 1.1 PresencePLUS Input Data Values

The following values are used as inputs by the PresencePLUS vision sensor, and can be set from the PLC or HMI to control the vision sensor.

| INPUT VALUE | USE |
|---|---|
| Trigger bit | Transitions from 0 to 1 cause the system to trigger an inspection. Trigger Divide and Trigger Width settings in the PresencePLUS software are ignored. |
| Remote Teach bit | Transitions from 0 to 1 cause a Remote Teach to occur on the next Trigger. |
| Product Change bit | Transitions from 0 to 1 cause a Product Change to occur. |
| Product Select value | The desired inspection number to change to when utilizing Product Change. |
| BCR String Change bit | Transitions from 0 to 1 cause the system to update the ASCII compare string used by the Barcode Test tool. Only applicable on vision sensors with BCR tool. |
| Tool String Change bit | Transitions from 0 to 1 cause the system to update the ASCII compare string used by the String tool or OCV tool. Only applicable on vision sensors with OCR/OCV tool. |

## 1.2 PresencePLUS Output Data Values

The following values are output by the PresencePLUS vision sensor, and can be read by the PLC or HMI.

| OUTPUT VALUE | USE |
|---|---|
| Pass bit | If set, the last inspection iteration passed. |
| Fail bit | If set, the last inspection iteration failed. |
| Error bit | If set, a system error has occurred. This must be checked out, and cleared, using the PresencePLUS software. |
| Ready bit | If set, the vision sensor is ready to accept a trigger and run an inspection. |
| I/O 1 to I/O 6 bits | Displays the current state of the I/O pins. Output Delay and Output Duration settings from the PresencePLUS software apply. |
| Inspection Number value | The number of the currently running inspection. Will be set to -1 (0xFFFF) if the sensor is not in Run mode (online). |
| System Error Count value | Displays the number of system errors that have occurred. Specific error codes can be seen using the PresencePLUS software. |
| Frame Count value | Number of images that have been captured since the vision sensor was last powered up. Used as an ID number in Log entries. |
| Pass Count value | Number of inspection iterations that have passed for this inspection number. May be reset using the PresencePLUS software. |
| Fail Count value | Number of inspection iterations that have failed for this inspection number. May be reset using the PresencePLUS software. |
| Missed Trigger Count value | Number of missed trigger incidents. Missed Triggers result from Trigger signals sent to a vision sensor that was not Ready. Missed Triggers do not result in an inspection being performed. |
| Iteration Count value | Number of total inspection iterations that have been performed. Defined as the sum of Pass, Fail, and Missed Trigger. This is a record of all Trigger signals that have arrived at the vision sensor, even those that did not cause an inspection to occur.<br>When this number increments, it signals that an inspection iteration is complete. |
| Trigger ACK bit | The Trigger acknowledge flag. |
| Remote Teach ACK bit | The Remote Teach acknowledge flag. |
| Product Change ACK bit | The Product Change acknowledge flag. |
| BCR String Change ACK bit | The BCR String Change acknowledge flag. Valid for sensors with BCR tool. |
| Tool String Change ACK bit | The Tool String Change acknowledge flag. Valid for sensors with OCR/OCV tools. |
| Inspection Specific values | See Chapter 2 for more information. |

### *1.2.1 ACK Flags*

For each of the defined Input Flags (Trigger, Remote Teach, Product Change, BCR String Change, Tool String Change) there is a corresponding acknowledgement or ACK flag found in the Output Flags register. The PresencePLUS vision sensor uses these ACK flags to echo the input flag back to the controlling PLC or HMI, providing a method for the PLC programmer to verify that the input flag transition was detected by the PresencePLUS sensor.

As an example, to use the Trigger ACK flag, the programming steps for triggering an inspection would be:

- Wait for the vision sensor to be Ready (i.e. Ready bit = 1)
- Set the Trigger bit to 1
- Wait for the Trigger ACK bit to go to 1 (signaling that the sensor heard you)
- Set Trigger bit back to 0 (as the 0→1 transition is what constitutes a valid trigger)

Waiting for the Trigger ACK bit to go to 1 assures that the Trigger flag was received by the PresencePLUS vision sensor. This is useful as most Industrial Ethernet configurations are asynchronous, meaning the PLC/HMI logic execution time, the PLC/HMI polling time, and the vision sensor operation are not all synched together on a standardized clock.

## 1.3 Sensor Operation

PLCs and HMIs can be used to Trigger inspections, Remote Teach new pass/fail conditions, and perform Product Changes to new inspection files, among other things. The programming instructions that follow assume that the vision sensor has been correctly set up and configured using the PresencePLUS software.

**General Note:** All Input Flag bits (like Trigger and Remote Teach) cause their respective actions to occur on the low-to-high transition of that flag. The Trigger bit changing from a 0 to a 1 is what makes the camera trigger and run an inspection. Keeping the Trigger bit at a value of 1, for example, will only result in a single inspection being performed. That is where the ACK flags come in; once you get the corresponding ACK flag for the input bit you are asserting, you can drive the input bit value back to 0 to be ready for next time.

### 1.3.1 Trigger Inspection

To trigger an inspection, perform the following steps:

- Wait for the Ready bit to be 1
- Toggle the Trigger bit from 0 to 1
- Wait for the Trigger ACK bit to go from 0 to 1
- After the Trigger ACK bit is seen by the PLC, toggle the Trigger bit back to 0
- Wait for the Iteration Count to increment. This will signal that the inspection is complete and you can now evaluate the rest of the output data.
- If the Missed Trigger count has incremented, the inspection did not occur.
- Check Pass, Fail, and Error Flags for inspection results as needed.

Optional steps that may be used when triggering an inspection:

- Before triggering, verify the Inspection Number is as expected. If not, make use of the Product Change functionality to switch inspections.
- The Ready bit can be used to determine if the inspection is complete in place of the incrementing of the Iteration Count number.

### 1.3.2 Remote Teach

Remote Teach occurs when the Remote Teach bit is toggled from 0 to 1, then is followed by a Trigger. In this case, the trigger sent to the camera results in a Remote Teach episode rather than the running of normal inspection iteration. To perform a Remote Teach, do the following steps:

- Wait for the Ready bit to be 1
- Toggle the Remote Teach bit from 0 to 1
- Wait for the Remote Teach ACK bit to go from 0 to 1
- After the Remote Teach ACK bit is seen by the PLC, toggle the Remote Teach bit back to 0
- Toggle the Trigger bit from 0 to 1
- Wait for the Trigger ACK bit to go from 0 to 1
- After the Trigger ACK bit is seen by the PLC, you can drive the Trigger bit back to 0
- Wait for the Iteration Count to increment. This will signal that the Remote Teach episode is complete.
- If the Missed Trigger Count has incremented, then the Remote Teach episode did not occur.
- Check the Pass and Fail bits to learn whether the Remote Teach episode was successful in modifying the inspection. A pass following the Remote Teach iteration signifies a successful Remote Teach. A fail indicates the Remote Teach attempt failed to change the inspection parameters.

**Note:** Remote Teach via Industrial Ethernet is only enabled when the **Product Select** option is set to 'Hardware Input' (found on the Select tab of the Run Screen in the PresencePLUS software)

### *1.3.3 Product Change*

Product Change is used to change the active inspection to another inspection stored in the PresencePLUS vision sensor's permanent flash memory. To perform a Product Change, do the following steps:

- Wait for the Ready bit to be 1
- Write the desired inspection number into the Product Select register
- Toggle the Product Change bit from 0 to 1
- Wait for the Product Change ACK bit to go from 0 to 1
- After the Product Change ACK bit is seen by the PLC, toggle the Product Change bit back to 0
- Check the Inspection Number register to verify whether the change-over occurred as desired. The Inspection Number register will either contain the desired inspection number (if the Product Change attempt was successful) or it will contain a "-1" (0xFFFF), indicating that a change to an invalid inspection number was attempted.

**Note:** If the Product Select register is set to a number for which there is no corresponding inspection stored in the PresencePLUS sensor's permanent flash memory, a System Error will occur and the sensor will be kicked out of Run mode. The Inspection Number register will show a "-1" (0xFFFF). If this happens, change the Product Select register to a valid value and try the Product Change attempt again. If the subsequent attempt is successful, the sensor will be back in the Run mode with the new inspection number loaded. The error bit will still be present until cleared using the PresencePLUS software.

# Chapter 2: Exporting Inspection-Specific Data

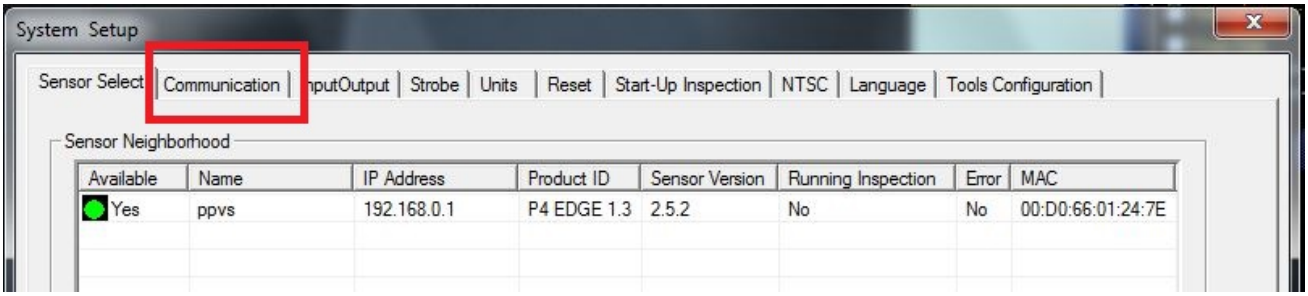
By default, a PresencePLUS sensor will only make inspection-independent data available to the PLC or HMI via an Industrial Ethernet connection. This data includes basic camera output variables discussed in Chapter 1 like the Pass and Fail count, the current Inspection Number, and the state of the I/O. The default data set also includes basic camera inputs like Trigger and Product Change, allowing the PLC or HMI to control the operation of the vision sensor.

## 2.1 Adding a Communication Tool

In order to make inspection-specific data available to the PLC/HMI, we need to add a Communication tool to the inspection file and then arrange for the Communication tool to send the desired information to the camera's array of output data.

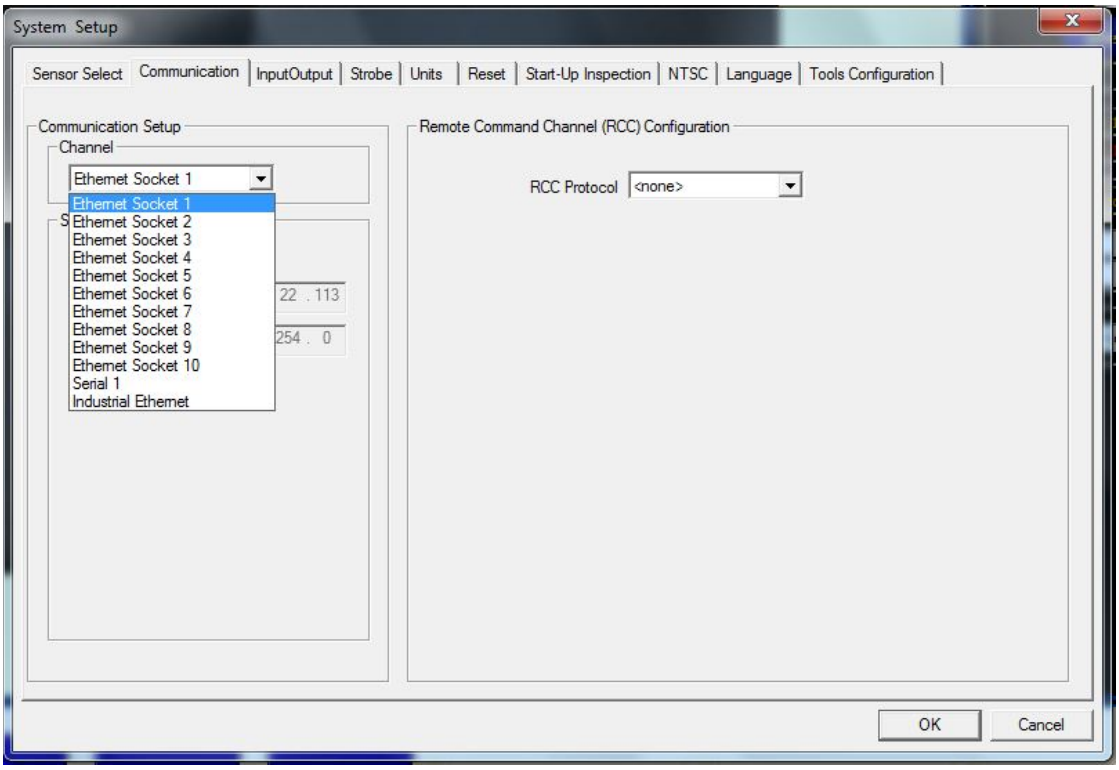
Before adding the Communication tool, it is suggested that we first tell the PresencePLUS vision sensor which Industrial Ethernet Protocol we'll be using. The sensor can 'speak' EtherNet/IP, Modbus/TCP, and PCCC. Telling the sensor ahead of time which protocol we're interested in allows the camera to package the registers in the correct format so we'll not get confused when working with the Communication tool.

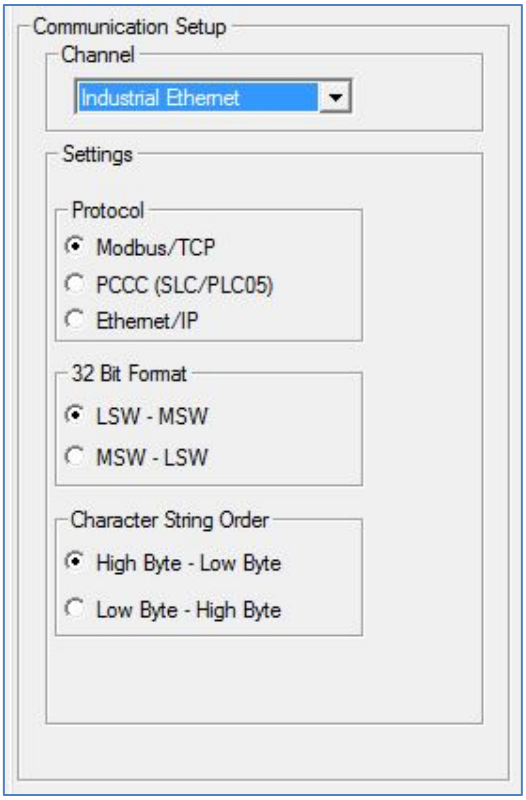
Connect the PresencePLUS PC software to the vision sensor in question. Click Stop (if needed) to stop the camera from inspecting and then click on the System button in the upper right corner of the PC Software. When the **System Setup** window opens, click on the **Communication** tab.




Here is the System Setup window (click the System button to get here). Now choose the "Communication" tab (in red box).

After opening up the Communication tab, click on the drop-down menu seen under **Communication Setup** and **Channel** (on the left hand side of the screen). A list of the various communication pathways available to the vision sensor appears. Select "Industrial Ethernet".




The picture at left shows the **Communication Setup** window after choosing "Industrial Ethernet" as the **Channel**.

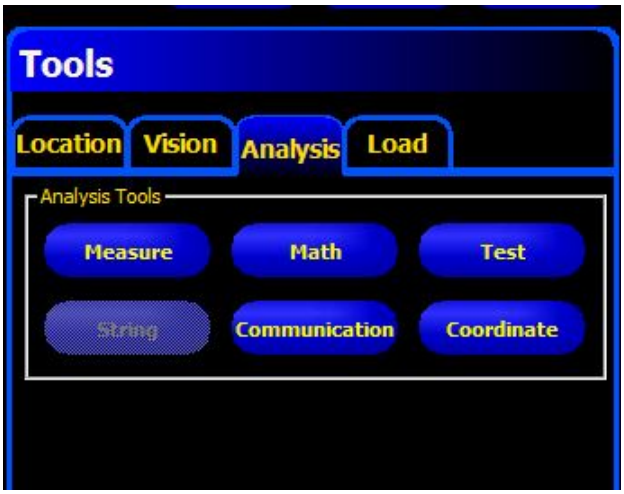
Now we can select the appropriate Protocol from the three radio button choices shown: Modbus/TCP, PCCC, or EtherNet/IP.

This setting choice is mostly cosmetic: we can still communicate with a CompactLogix PLC over EtherNet/IP even if we have this radio button set to Modbus/TCP, for instance.

Mainly, this is the place for us to tell the PC Software how we'd like the memory map to look while we're working with the Communication tool (e.g. whether we mention 'Assembly Instances' or not, which register number we start with: 0 or 1, etc.). Once you've made the correct choice, click the OK button in the lower right corner of the System Setup window.

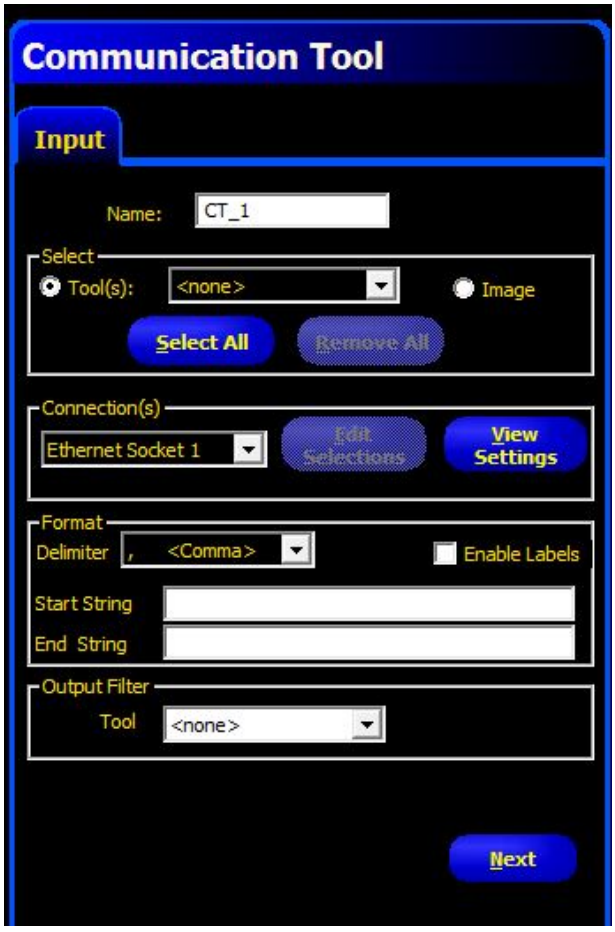
Now we can add the Communication tool to our existing vision inspection. We need to add the Communication tool last (or nearly so), as it can only work with information supplied by tools to its left (i.e. tools that execute earlier in the sequence of operation).

We need to go to the Tools screen and load the inspection we'd like to modify. Then go to the Analysis tab on the Tools screen (if necessary). This is where we'll find the Communication tool. Click on the blue button labeled "Communication" to add one of these tools to your inspection.





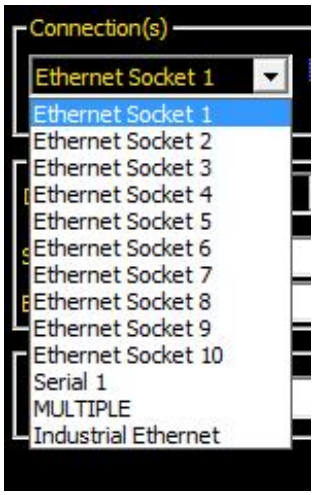
Here is a sample inspection as shown along the bottom of the tools screen. Note how the Communication tool comes last.



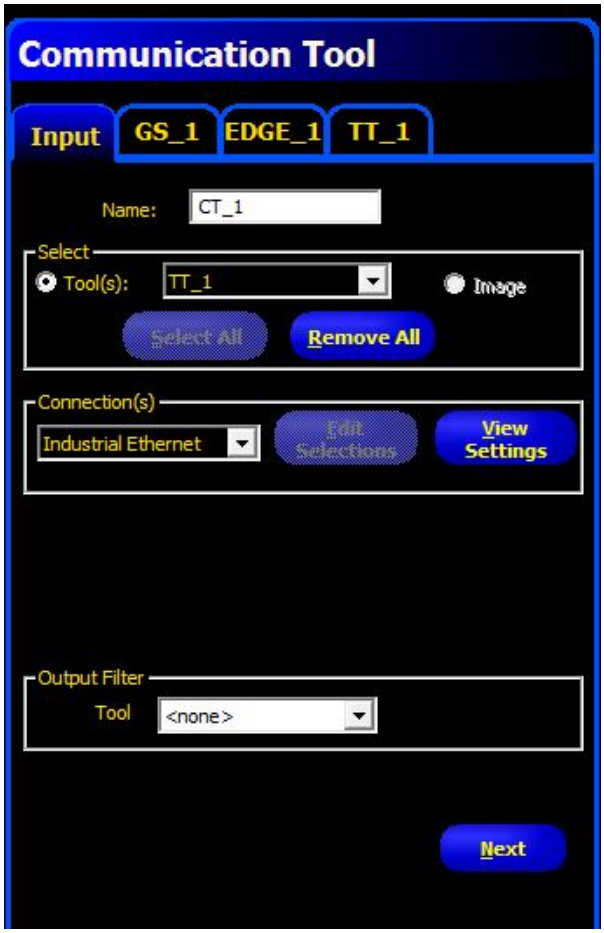

The picture at left shows the Communication tool. Near the top of the tool there is a box called **Select** with two radio buttons, one is labeled 'Tool(s)' and the other is labeled 'Image'. We want to keep this selection set to Tool(s), so the Communication tool will be working to export vision tool data. From the drop-down menu in the middle of the Select box, choose the vision tool or tools that you'd like the PLC or HMI to know more about.

You can use a single Communication tool to send many vision tools' worth of inspection specific-data to the Industrial Ethernet memory map.

Next, we will go to the **Connection(s)** portion of the Communication tool, found about midway down the column. From this drop-down menu, we'll choose Industrial Ethernet from the bottom of the list.



In the example shown at right, we have a Communication tool set to export some tool data to the Industrial Ethernet connection.

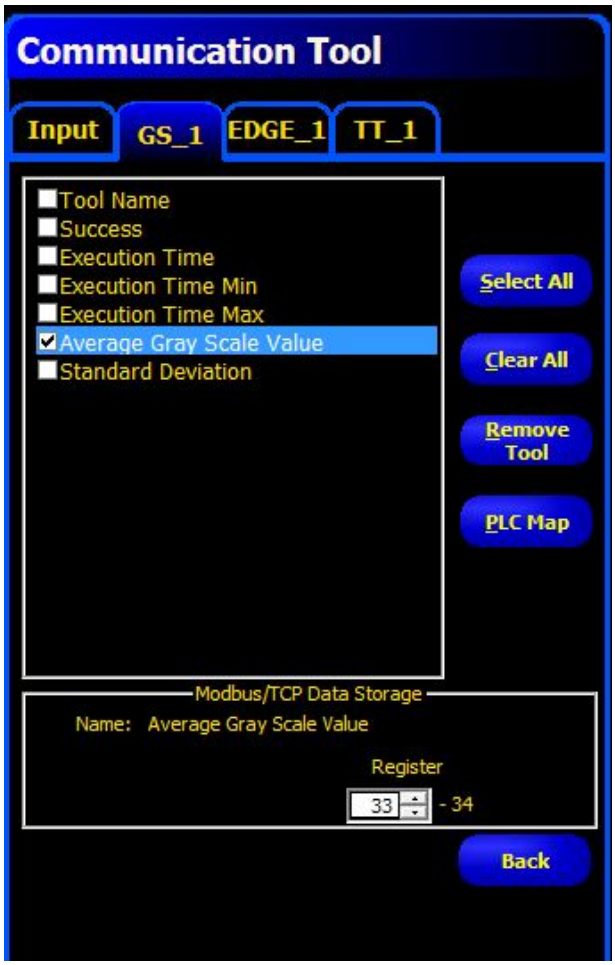



Each of the tools selected from the drop-down menu in the **Select** box now have their own tab along the top of the Communication tool.

We click on each of these tabs to let the vision sensor know which specific pieces of data from each tool we'd like to send to the memory map that will be made available to the PLC or HMI.

An example is shown below, wherein we elect to send the 'Average Gray Scale Value' from the vision tool named GS_1 to the memory map.

In this example, the Protocol selected in the Software System → Communication →Industrial Ethernet (see page 7) was Modbus/TCP.



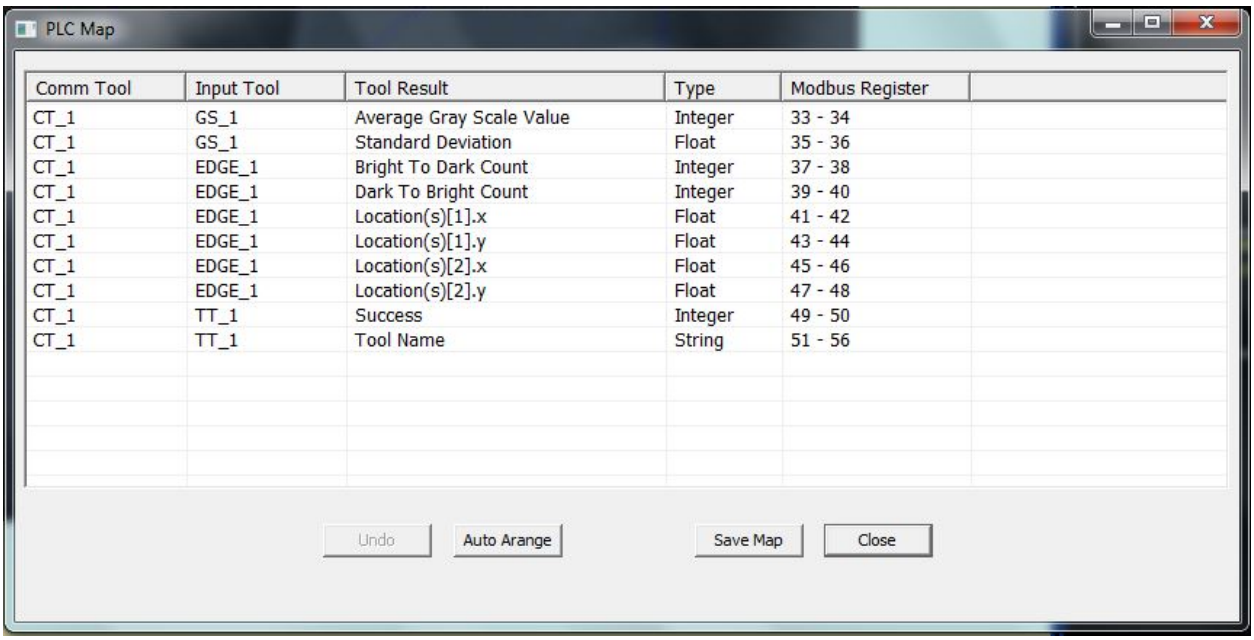

Note how the Communication tool reports, near the bottom of the tool pictured at left, that the Average Gray Scale Value will be written into Modbus/TCP registers 33-34.

We'll see more about the memory maps specific to each protocol in later chapters, but just remember that the inspection-independent data (like Pass/Fail count and other information seen in Chapter 1) is permanently assigned a space in the memory map while inspection-specific data (like the Average Gray Scale Value from this sample inspection) is only sent to the memory map by the direct action of a Communications tool inserted into the inspection.

If you have multiple inspection files saved on a vision sensor, each inspection would have to have its own Communication tool set up in this way in order to send inspection-specific data to the PLC or HMI.

## 2.2 The PLC Map

After using the Communication tool to add data to the memory map, click on the **PLC Map** button found on the subpages of the Communication tool. The PLC map is just that, a record of which data is going to be placed in which registers, for the given inspection file.




| Comm Tool | Input Tool | Tool Result | Type | Modbus Register |
|---|---|---|---|---|
| CT_1 | GS_1 | Average Gray Scale Value | Integer | 33 - 34 |
| CT_1 | GS_1 | Standard Deviation | Float | 35 - 36 |
| CT_1 | EDGE_1 | Bright To Dark Count | Integer | 37 - 38 |
| CT_1 | EDGE_1 | Dark To Bright Count | Integer | 39 - 40 |
| CT_1 | EDGE_1 | Location(s)[1].x | Float | 41 - 42 |
| CT_1 | EDGE_1 | Location(s)[1].y | Float | 43 - 44 |
| CT_1 | EDGE_1 | Location(s)[2].x | Float | 45 - 46 |
| CT_1 | EDGE_1 | Location(s)[2].y | Float | 47 - 48 |
| CT_1 | TT_1 | Success | Integer | 49 - 50 |
| CT_1 | TT_1 | Tool Name | String | 51 - 56 |

Here is an example of a PLC map, generated from our sample inspection. Each entry includes the vision tool generating the data (Input Tool), the data values (Tool Result), and the memory location the data will be stored in (here called Modbus Register).

**Note:** One other piece of information gained from the PLC Map is the data **Type**. Not all the pieces of information that we can export from the PresencePLUS vision sensor are 16-bit integers. Some of the data is comprised of Floating point numbers (also called Real numbers by some PLCs), some may take the form of ASCII strings. Knowing what type of data is stored in each register is as important as knowing which Tool Result the data represents. See Chapter 7 for more information on the various data types used by the PresencePLUS vision sensors.

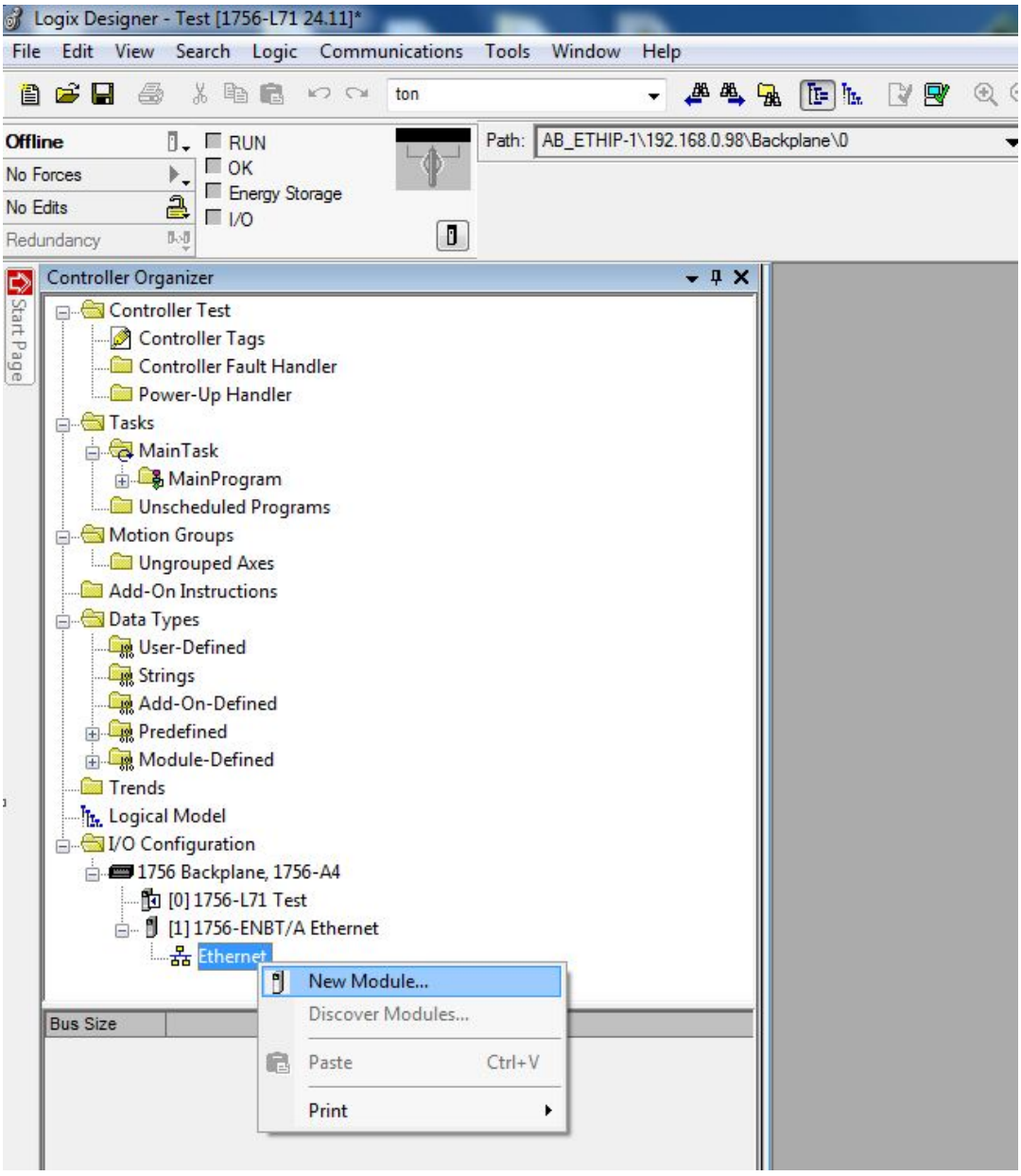
Click on the **Save Map** button to save a copy of the PLC Map, as a text file, to your computer. You can reference this file later on when programming your PLC or HMI.

# Chapter 3: EtherNet/IP

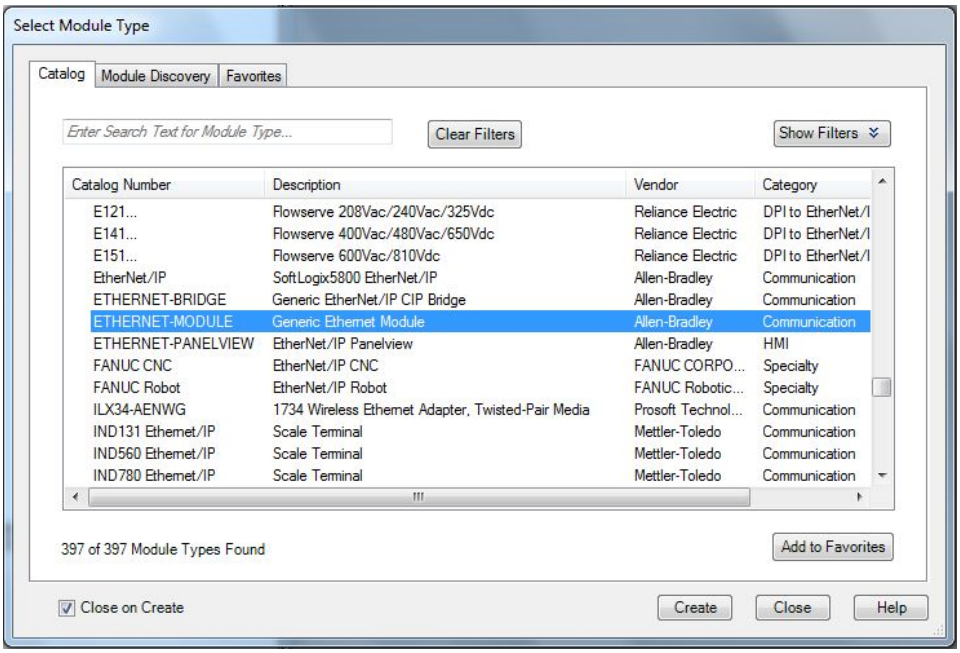
## 3.1 RSLogix5000 Configuration as Generic Ethernet Module

We use a "Generic Ethernet Module" to create an implicit Class 1 configuration between a PresencePLUS vision sensor with old firmware and a ControlLogix family PLC. The following is a sample setup:

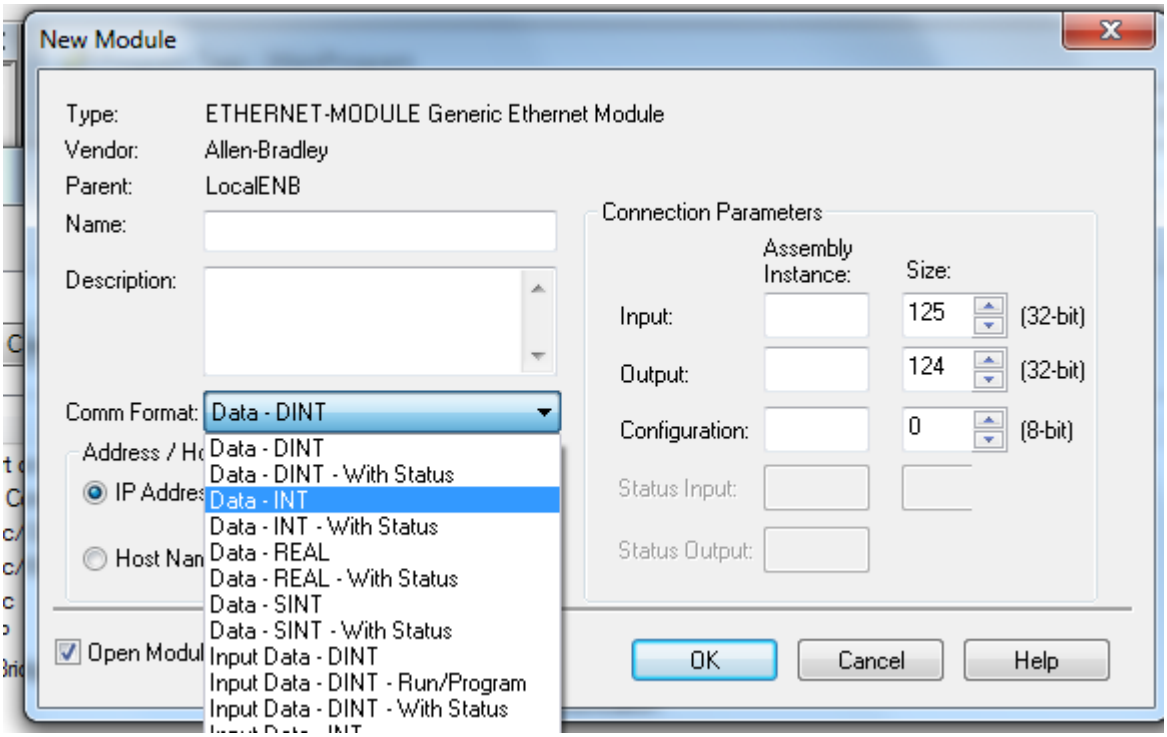
1. Add a module to the PLC's Ethernet card.



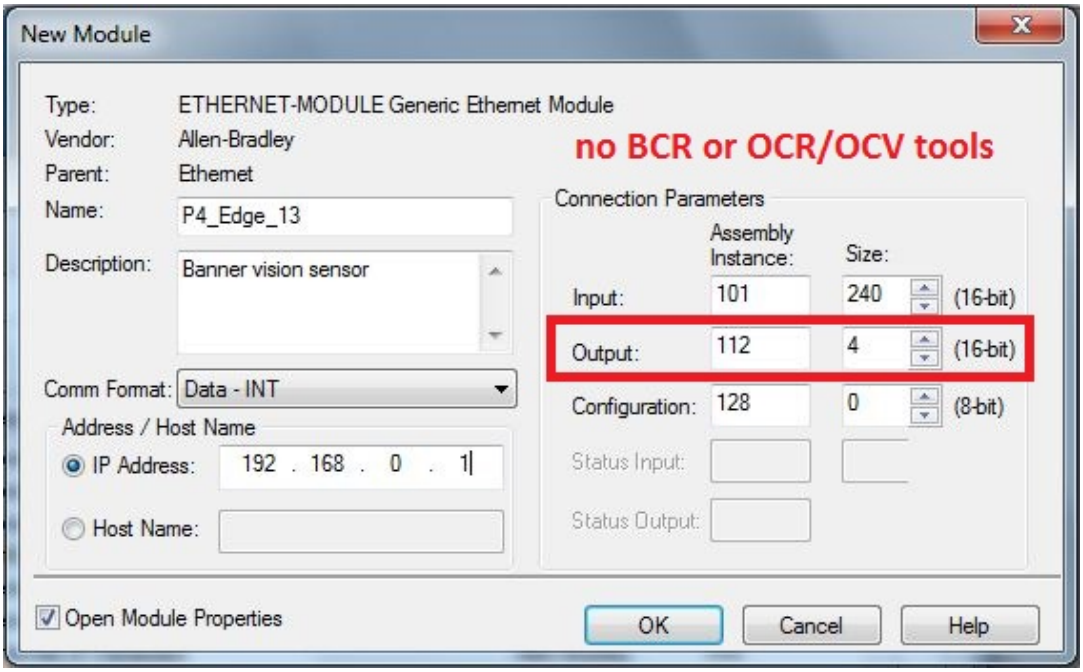
2. Select Module as "Generic Ethernet Module"



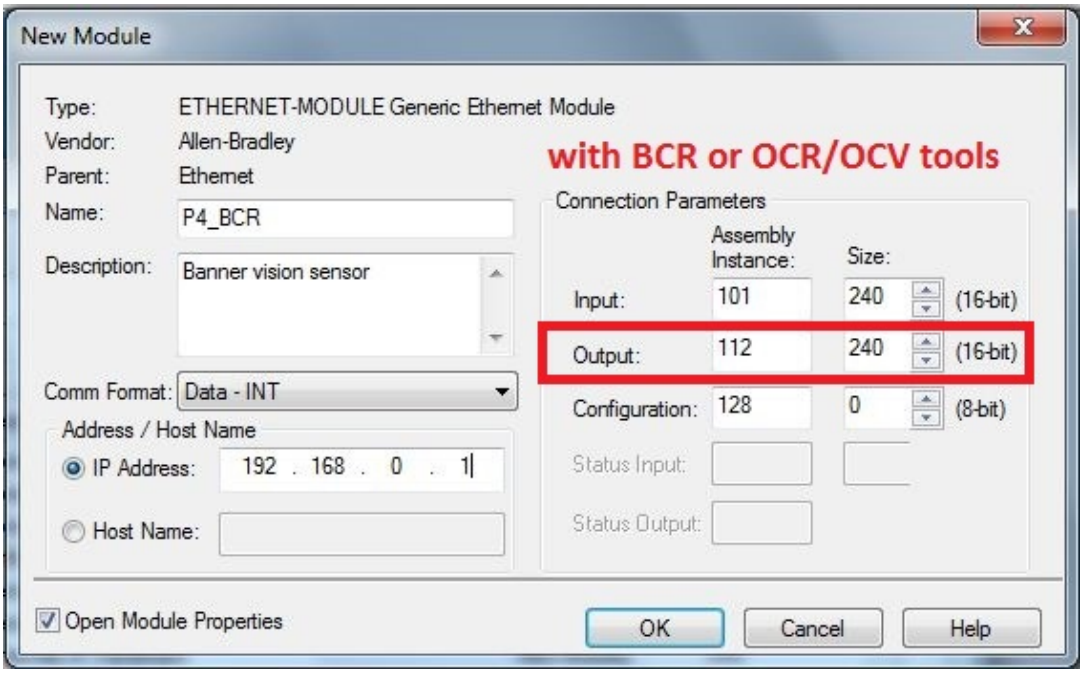

3. Change Comm Format to INT (default is DINT)



4. Add a module name and the IP address of the PresencePLUS vision sensor (default sensor IP address is 192.168.0.1 with a subnet mask of 255.255.255.0).
5. Choose one of these Assembly Object setups, based on whether the PresencePLUS vision sensor has the optional BCR or OCR/OCV tools unlocked. See section 3.3 for more information about the available Assembly Objects.

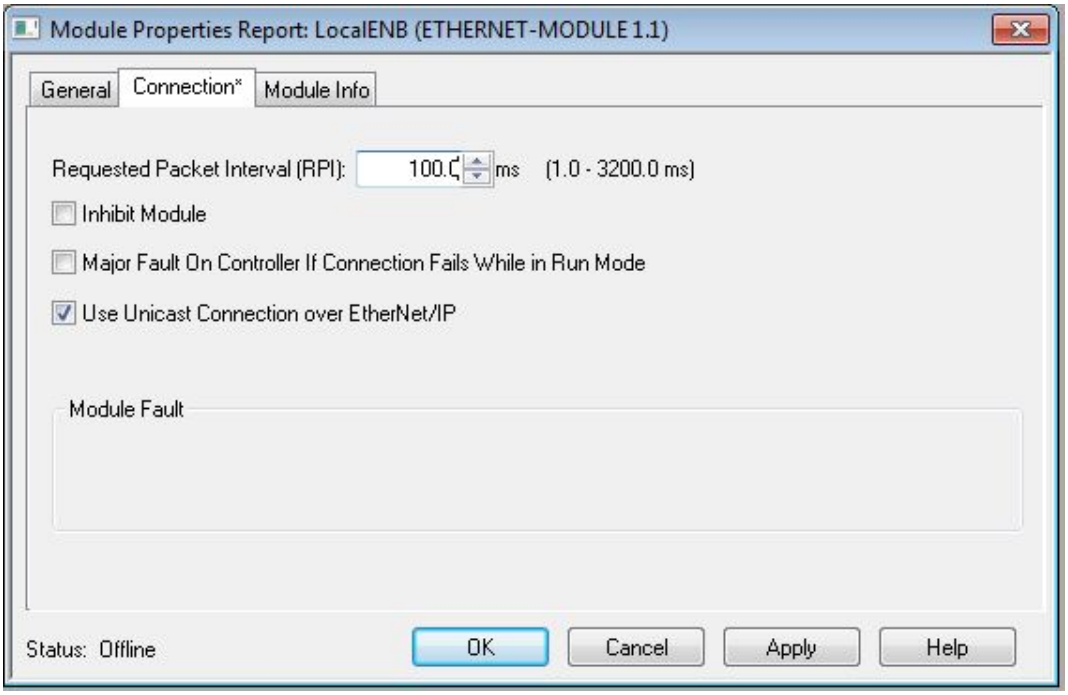



Choose this setup (PLC Output Assembly Instance 112 → size 4) when no BCR or OCR/OCV tools are present

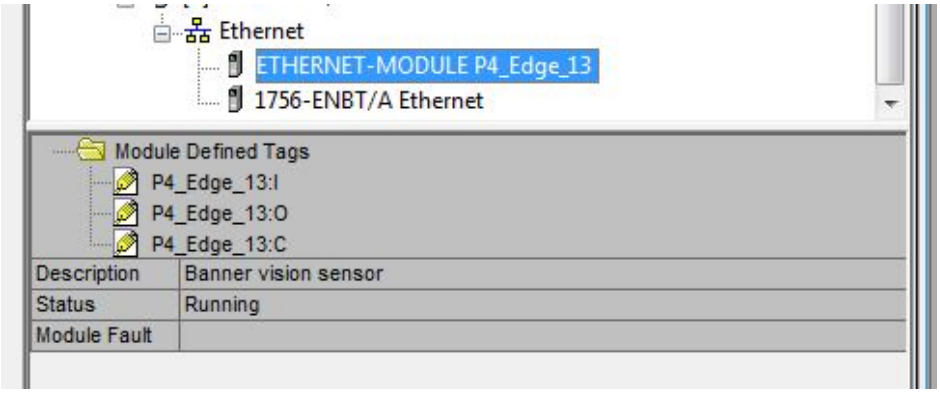



Choose this setup (PLC Assembly Instance 112 → size 240) when BCR and/or OCR/OCV tools are present.

6. Set the Connection parameters: RPI and Unicast yes/no. Note that the recommended minimum RPI is 100msec.




7. If the module configuration was successful, the following information should be displayed:




I = Inputs to PLC (outputs from the vision sensor)
O = Outputs from PLC (inputs to the vision sensor)
C = Configuration (not used)

8. Locate the memory map in the Controller Tags list. The PLC Input (PresencePLUS Output) words from Assembly Instance 100 (0x64) are shown below as an example.

| | | | | | |
|---|---|---|---|---|---|
| ⊞ | P4_Edge_13:C | {...} | {...} | | AB:ETHERNET_MODULE:C:0 |
| ⊟ | P4_Edge_13:I | {...} | {...} | | AB:ETHERNET_MODULE_INT_32Bytes:. |
| ⊟ | P4_Edge_13:I.Data | {...} | {...} | Decimal | INT[16] |
| ⊞ | P4_Edge_13:I.Data[0] | 12288 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[1] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[2] | -1 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[3] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[4] | 10476 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[5] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[6] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[7] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[8] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[9] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[10] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[11] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[12] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[13] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[14] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:I.Data[15] | 0 | | Decimal | INT |
| ⊞ | P4_Edge_13:O | {...} | {...} | | AB:ETHERNET_MODULE_INT_8Bytes:... |

Here we can see some of the data coming back from a PresencePLUS camera. Notice how word [2] reads "-1". This register is used to report the currently running inspection number. A "-1" means the camera is currently offline.

## 3.2 Assembly Objects

### *3.2.1 Inputs to PLC (Outputs from PresencePLUS) T→O*

PresencePLUS vision sensors with firmware older than 2.7.5 have two choices for Vision Sensor Output/PLC Input Assembly Instances. One choice is a small group of registers which provide just the basic inspection-independent data from the vision sensor. The other choice is a larger group of registers that include both the basic data and some space reserved for inspection-specific information as defined by the user (see Chapter 2 for more information).

**PLC Input Assembly Instance 100 (0x64) - 16 Registers**

This Assembly Instance includes only inspection-independent data like Pass Count, Fail Count, and the status of the I/O.

**PLC Input Assembly Instance 100 (0x64) – PresencePLUS Outputs T→O**

| WORD # | WORD NAME | DATA TYPE |
|---|---|---|
| 0 | Output Flags (see Flags, section 3.3) | 16-bit integer |
| 1 | ACK Flags (see Flags, section 3.3) | 16-bit integer |
| 2 | Inspection Number | 16-bit integer |
| 3 | System Error Count | 16-bit integer |
| 4-5 | Frame Count | 32-bit integer |
| 6-7 | Pass Count | 32-bit integer |
| 8-9 | Fail Count | 32-bit integer |
| 10-11 | Missed Triggers | 32-bit integer |
| 12-13 | Iteration Count | 32-bit integer |
| 14 | *reserved* | 16-bit integer |
| 15 | *reserved* | 16-bit integer |

**PLC Input Assembly Instance 101 (0x65) - 240 Registers**

This larger Assembly Instance includes inspection-specific data in addition to the basic information like Pass Count, Fail Count, and the status of the I/O. The registers labeled Location 1, Location 2, etc. represent registers reserved for inspection-specific data. A Communication tool is required to map data to these locations. See Chapter 2 for more information.

**PLC Input Assembly Instance 101 (0x65) – PresencePLUS Outputs T→O**

| WORD # | WORD NAME | DATA TYPE |
|---|---|---|
| 0 | Output Flags (see Flags, section 3.3) | 16-bit integer |
| 1 | ACK Flags (see Flags, section 3.3) | 16-bit integer |
| 2 | Inspection Number | 16-bit integer |
| 3 | System Error Count | 16-bit integer |
| 4-5 | Frame Count | 32-bit integer |
| 6-7 | Pass Count | 32-bit integer |
| 8-9 | Fail Count | 32-bit integer |
| 10-11 | Missed Triggers | 32-bit integer |
| 12-13 | Iteration Count | 32-bit integer |
| 14-31 | *reserved* | 16-bit integer |
| 32-33 | Location 1 | 32-bit integer |
| 34-35 | Location 2 | 32-bit integer |
| 36-37 | Location 3 | 32-bit integer |
| 38-39 | Location 4 | 32-bit integer |
| … | … | |
| 236-237 | Location 103 | 32-bit integer |
| 238-239 | Location 104 | 32-bit integer |

*Word Number = 32 + (Location -1)*2)*

**PLC Input Assembly Instance 102 (0x66) - 240 Registers**

This Assembly Instance only has room for inspection-specific data. None of the registers are pre-mapped with Pass Count, Fail Count, or similar. All 240 words are given over to user-defined data. The registers labeled Location 105, Location 106, etc. represent the locations where inspection-specific data can be stored by a Communication tool. See Chapter 2 for more information.

**PLC Input Assembly Instance 102 (0x66) – PresencePLUS Outputs T→O**

| WORD # | WORD NAME | DATA TYPE |
|---|---|---|
| 0-1 | Location 105 | 32-bit integer |
| 2-3 | Location 106 | 32-bit integer |
| 4-5 | Location 107 | 32-bit integer |
| … | … | |
| 236-237 | Location 223 | 32-bit integer |
| 238-239 | Location 224 | 32-bit integer |

*Word Number = ((Location -105)*2*

**PLC Input Assembly Instance 103 (0x67) and 104 (0x68) - 240 Registers each (PresencePLUS Outputs/PLC Inputs) T->O**

Two more wide open Assembly Instances are also implemented on the PresencePLUS vision sensors. These Instances are functionally equivalent to 0x66 (102) and have only user-defined data in them.

### *3.2.2 Outputs from PLC (PresencePLUS Inputs)* O→T

PresencePLUS vision sensors with firmware older than 2.7.5 have a single, variable sized PLC Output/Vision Sensor Input Assembly Instance. The size of this instance changes based on whether the vision sensor in question has a BCR tool and/or OCR/OCV tools on board. BCR (barcode reader) and OCR/OCV (optical character recognition/optical character verification) tools generate and utilize ASCII strings. The larger sized PLC Output/Vision Sensor Input memory map allows for the PLC or HMI to send a new ASCII compare strings to the vision sensor to change the pass/fail criteria "on the fly".

**PLC Output Assembly Instance 112 (0x70) - 4 Registers**

***Only used in vision sensors without BCR or OCR/OCV tools***

This version of Assembly Instance 112 includes only basic controls like Trigger, Remote Teach, and Product Change. See Flags in section 3.4 for more information.

**PLC Output Assembly Instance 112 (0x70) – PresencePLUS Inputs w/o BCR, OCR/OCV O→T**

| WORD # | WORD NAME | DATA TYPE |
|---|---|---|
| 0 | Input Flags (see Flags, section 3.3) | 16-bit integer |
| 1 | Product Select | 16-bit integer |
| 2 | *reserved* | 16-bit integer |
| 3 | *reserved* | 16-bit integer |

**PLC Output Assembly Instance 112 (0x70) - 240 Registers**

***Only used in vision sensors with BCR or OCR/OCV tools***

This version of Assembly Instance 112 adds to the basic functionality of Trigger, Remote Teach, and Product Change the ability to send new ASCII compare strings to the vision sensor, allowing for the changing of BCR and/or OCR/OCV pass fail criteria. See Chapter 6 for more information on the string regions.

**PLC Output Assembly Instance 112 (0x70) – PresencePLUS Inputs w/ BCR, OCR/OCV O→T**

| WORD # | WORD NAME | DATA TYPE |
|---|---|---|
| 0 | Input Flags (see Flags, section 3.3) | 16-bit integer |
| 1 | Product Select | 16-bit integer |
| 2 | Tool String Update Flags (see Flags, section 3.3) | 16-bit integer |
| 3 | *reserved* | 16-bit integer |
| … | *reserved* | 16-bit integer |
| 39 | String Region 1 (50 or 25 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 64 | (String Region 1 Mask, 25 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 89 | String Region 2 (50 or 25 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 114 | (String Region 2 Mask, 25 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 139 | String Region 3 (25 or 12 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 151 | (String Region 3 Mask, 12 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 164 | String Region 4 (25 or 12 words, see Chapter 6) | 16-bit integers |

| | | |
|---|---|---|
| … | … | |
| 176 | (String Region 4 Mask, 12 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 189 | String Region 5 (25 or 12 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 201 | (String Region 5 Mask, 12 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 214 | String Region 6 (25 or 12 words, see Chapter 6) | 16-bit integers |
| … | … | |
| 226 | (String Region 6 Mask, 12 words, see Chapter 6) | 16-bit integers |
| 239 | … | |

### *3.2.3 PresencePLUS Configuration Assembly Object*

PresencePLUS vision sensors do not use a Configuration Assembly Object. As some EtherNet/IP clients require one, use PLC Configuration Assembly Instance 128 (0x80) with a size of zero 16-bit registers.

## 3.3 Flags

### OUTPUT FLAGS

These bits are outputs from the PresencePLUS vision sensor (inputs to the PLC). They are used to report the basic status of the vision sensor and the last inspection run.

**e.g. Word #0, Assembly Instance 100 (0x64)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *reserved* | *reserved* | I/O 6 | I/O 5 | I/O 4 | I/O 3 | I/O 2 | I/O 1 | *reserved* | *reserved* | *reserved* | Ready | *reserved* | Error | Fail | Pass |

### ACK FLAGS

These bits are outputs from the PresencePLUS vision sensor (inputs to the PLC). They are used to acknowledge each of the input flags sent from the PLC. For example, if the PLC changes the Trigger bit (from Input Flags, above) from a 0 to a 1, the vision sensor will change the Trigger ACK bit from a 0 to a 1 in response.

**e.g. Word #1, Assembly Instance 100 (0x64)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Tool String Change ACK | BCR String Change ACK | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Product Change ACK | Remote Teach ACK | Trigger ACK |

### INPUT FLAGS

These bits are inputs to the PresencePLUS vision sensor (outputs from the PLC). They are used for basic control of the vision sensor.

**e.g. Word #0, Assembly Instance 112 (0x70)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Tool String Change | BCR String Change | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Product Change | Remote Teach | Trigger |

### TOOL STRING UPDATE FLAGS

These bits are inputs to a PresencePLUS vision sensor (outputs from the PLC). These are only used when the vision sensor in question has BCR or OCR/OCV tools (and thus makes use of ASCII string regions). See Chapter 6 for more information on the String Update regions.

**e.g. Word #2, Assembly Instance 112 (0x70) (Barcode/OCR/OCV only)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Region 6 Mask | Region 5 Mask | Region 4 Mask | Region 3 Mask | Region 2 Mask | Region 1 Mask | *reserved* | *reserved* | *reserved* | *reserved* | Region 6 Data | Region 5 Data | Region 4 Data | Region 3 Data | Region 2 Data | Region 1 Data |

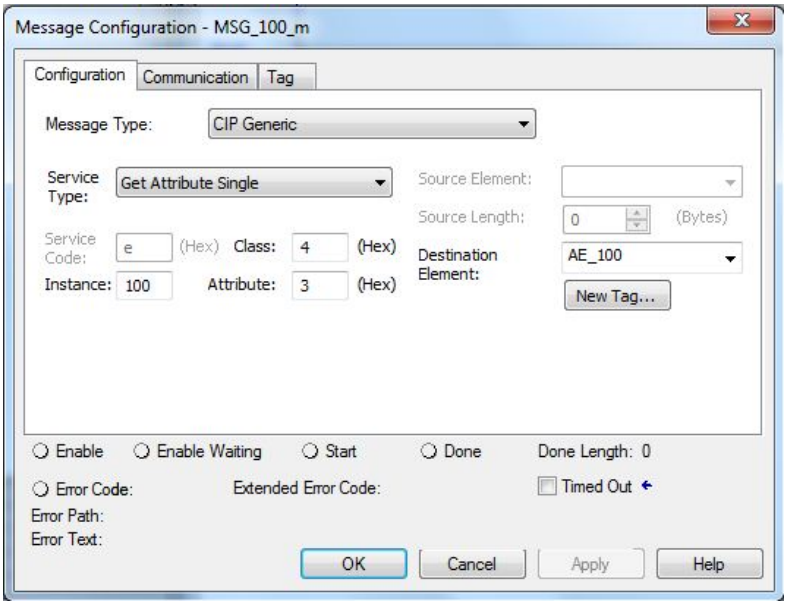
## 3.4 RSLogix5000 Configuration (Explicit Messaging)

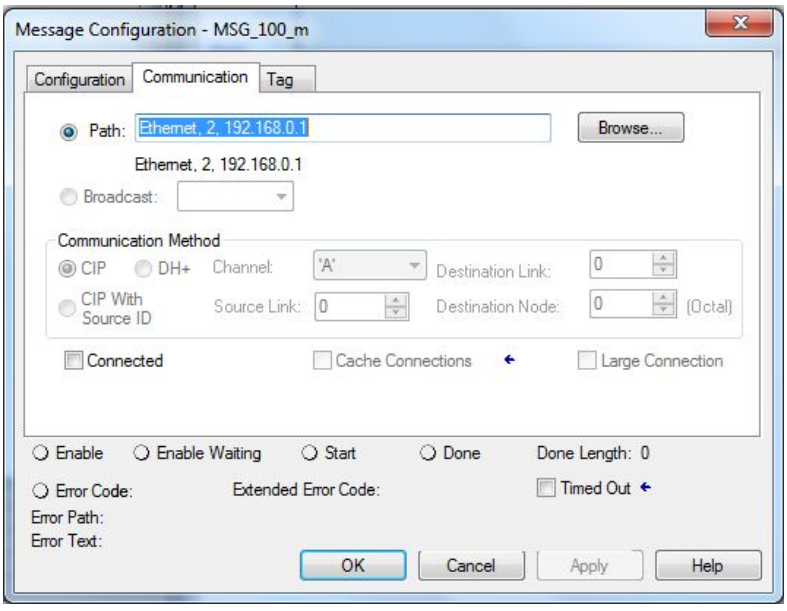
To get a copy of one of the Assembly Instances from section 3.2, use Service Code 14 (Get Attribute Single, hex 0E), Class 4, Instance 100 (0x64) or 101 (0x65) or 102 (0x66), Attribute 3. A successful Explicit Message of this type will return the appropriate Assembly Instance as show in section 3.2.

### 3.4.1 Example of Explicit Message Connection Assembly Instance 0x64 (100)

To get the 100 (0x64) Assembly Instance, use Service Type 14 (Get Attribute Single, hex 0E), Class 4, Instance 100, Attribute 3. A successful Explicit Message of this type will return all 16 registers of the 100 (0x64) Assembly Instance, as defined in section 3.2.1.

Here is the MSG command for this explicit message.



Here is a user-defined array (called **AE_100**) showing all 16 registers of Assembly Instance 100 (0x64).

| AE_100 | {...} | {...} | Decimal | INT[16] |
|---|---|---|---|---|
| AE_100[0] | 12305 | | Decimal | INT |
| AE_100[1] | 0 | | Decimal | INT |
| AE_100[2] | 1 | | Decimal | INT |
| AE_100[3] | 0 | | Decimal | INT |
| AE_100[4] | -27894 | | Decimal | INT |
| AE_100[5] | 0 | | Decimal | INT |
| AE_100[6] | 16 | | Decimal | INT |
| AE_100[7] | 0 | | Decimal | INT |
| AE_100[8] | 0 | | Decimal | INT |
| AE_100[9] | 0 | | Decimal | INT |
| AE_100[10] | 0 | | Decimal | INT |
| AE_100[11] | 0 | | Decimal | INT |
| AE_100[12] | 16 | | Decimal | INT |
| AE_100[13] | 0 | | Decimal | INT |
| AE_100[14] | 0 | | Decimal | INT |
| AE_100[15] | 0 | | Decimal | INT |

In this example we see that the current running inspection is 1, as register [2] is "1".

Making an explicit message connection from scratch in an Allen-Bradley PLC program requires the following steps:

1. Make a new tag with the Message data type
2. Make a new tag to act as a Destination Element (a 16-bit array large enough to hold the data you'll be requesting).
3. Add a MSG command to your ladder logic (using the Message tag from #1 and the Destination Element from #2). The Class, Instance, and Attribute values depend on the data desired.
4. In the Communication tab of the MSG command, type in the Path to the safety controller:
   e.g. Ethernet, 2, 192.168.0.1
   where
   the "2" is the slot number for the EtherNet/IP card in the PLC rack and the IP Address shown is that of the vision sensor

# Chapter 4: Modbus/TCP

The Modbus/TCP protocol provides device information using register and coil banks defined by the slave device. This section defines the register and coil banks. By specification, Modbus/TCP uses TCP port 502. PresencePLUS vision sensors do not support a Unit ID of 0 (sometimes called Slave ID or Device ID).

The following registers are used to send values back and forth from the vision sensor to the PLC. PresencePLUS read-only output data can be read as Input Registers (30000) using Modbus function code 04 (Read Input Registers). Because some devices like the Modicon family of PLCs cannot access data using the 30000 range of registers, the same values can also be seen as Holding Registers (40000) using Modbus function code 03 (Read Holding Registers).

The Input Flags can be set as Coils, using Modbus function code 05 (Force Single Coil). The state of the ACK Flags and Output Flags can be read as Inputs (10000) using Modbus function code 02 (Read Input Status).

*Modbus Function Codes Supported*

01: Read Coil Status
02: Read Input Status
03: Read Holding Registers
04: Read Input Registers
05: Force Single Coil
06: Preset Single Register
07: Read Exception Status
16: Preset Multiple Registers

**Input Flags (Coils 00001-00016)**

*05: Force Single Coil*

| Register | Bit Position | WORD Name |
|---|---|---|
| 00001 | 0 | Trigger |
| 00002 | 1 | Remote Teach |
| 00003 | 2 | Product Change |
| 00004 | 3 | *reserved* |
| 00005 | 4 | *reserved* |
| 00006 | 5 | *reserved* |
| 00007 | 6 | *reserved* |
| 00008 | 7 | *reserved* |
| 00009 | 8 | BCR String Change |
| 00010 | 9 | Tool String Change |
| 00011 | 10 | *reserved* |
| 00012 | 11 | *reserved* |
| 00013 | 12 | *reserved* |
| 00014 | 13 | *reserved* |
| 00015 | 14 | *reserved* |
| 00016 | 15 | *reserved* |

**Output Flags (Inputs 10001-10016)**

*02: Read Input Status*

| Register | Bit Position | WORD Name |
|---|---|---|
| 10001 | 0 | Pass |
| 10002 | 1 | Fail |
| 10003 | 2 | Error |
| 10004 | 3 | *reserved* |
| 10005 | 4 | Ready |
| 10006 | 5 | *reserved* |
| 10007 | 6 | *reserved* |
| 10008 | 7 | *reserved* |
| 10009 | 8 | I/O 1 |
| 10010 | 9 | I/O 2 |
| 10011 | 10 | I/O 3 |
| 10012 | 11 | I/O 4 |
| 10013 | 12 | I/O 5 |
| 10014 | 13 | I/O 6 |
| 10015 | 14 | *reserved* |
| 10016 | 15 | *reserved* |

**ACK Flags (Inputs 10017-10032)**

*02: Read Input Status*

| Register | Bit Position | WORD Name |
|---|---|---|
| 10017 | 0 | Trigger ACK |
| 10018 | 1 | Remote Teach ACK |
| 10019 | 2 | Product Change ACK |
| 10020 | 3 | *reserved* |
| 10021 | 4 | *reserved* |
| 10022 | 5 | *reserved* |
| 10023 | 6 | *reserved* |
| 10024 | 7 | *reserved* |
| 10025 | 8 | BCR String Change ACK |
| 10026 | 9 | Tool String Change ACK |
| 10027 | 10 | *reserved* |
| 10028 | 11 | *reserved* |
| 10029 | 12 | *reserved* |
| 10030 | 13 | *reserved* |
| 10031 | 14 | *reserved* |
| 10032 | 15 | *reserved* |

**PresencePLUS Output Registers (Modbus/TCP Input or Holding Registers)**

*04: Read Input Registers* or *03: Read Holding Registers*

| Input REG # | Holding REG # | WORD NAME | DATA TYPE |
|---|---|---|---|
| 1 | 1001 | Output Flags (see Flags, section 4.1) see also Inputs 10001-16 | 16-bit integer |
| 2 | 1002 | ACK Flags (see Flags, section 4.1) see also Inputs 10017-32 | 16-bit integer |
| 3 | 1003 | Inspection Number | 16-bit integer |
| 4 | 1004 | System Error Count | 16-bit integer |
| 5-6 | 1005-6 | Frame Count | 32-bit integer |
| 7-8 | 1007-8 | Pass Count | 32-bit integer |
| 9-10 | 1009-10 | Fail Count | 32-bit integer |
| 11-12 | 1011-12 | Missed Triggers | 32-bit integer |
| 13-14 | 1013-14 | Iteration Count | 32-bit integer |
| 15-32 | 1015-32 | *reserved* | 16-bit integer |
| 33-34 | 1033-34 | Location 1 | 32-bit integer |
| 35-36 | 1035-36 | Location 2 | 32-bit integer |
| 37-38 | 1037-38 | Location 3 | 32-bit integer |
| … | … | … | … |
| 957-958 | 1957-958 | Location 463 | 32-bit integer |
| 959-960 | 1959-960 | Location 464 | 32-bit integer |

*Register Number = 33 + (Location – 1)*2)*

**PresencePLUS Input Registers (Modbus/TCP Holding Registers)**

*06: Preset Single Register* or *16: Preset Multiple Registers*

| REG # | WORD NAME | DATA TYPE |
|---|---|---|
| 1 | Input Flags (see Flags, section 4.1) see also Coils 00001-16 | 16-bit integer |
| 2 | Product Select | 16-bit integer |
| 3 | Tool String Update Flags (see Flags, section 4.1)* | 16-bit integer |
| 4-39 | *reserved* | 16-bit integer |
| 40-239 | String Input Regions (see Chapter 6)* | *varies* |

**These registers are only defined for vision sensors with BCR and/or OCR/OCV tools.*

## 4.1 Flags

# OUTPUT FLAGS

These read-only bits are outputs from the PresencePLUS vision sensor (inputs to the PLC). They are used to report the basic status of the vision sensor and the last inspection run.

### PLC Input Register 1 or Holding Register 1001, also Inputs 10001-16

| Input 16 | Input 15 | Input 14 | Input 13 | Input 12 | Input 11 | Input 10 | Input 9 | Input 8 | Input 7 | Input 6 | Input 5 | Input 4 | Input 3 | Input 2 | Input 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| *reserved* | *reserved* | I/O 6 | I/O 5 | I/O 4 | I/O 3 | I/O 2 | I/O 1 | *reserved* | *reserved* | *reserved* | Ready | *reserved* | Error | Fail | Pass |

# ACK FLAGS

These read-only bits are outputs from the PresencePLUS vision sensor (inputs to the PLC). They are used to acknowledge each of the input flags sent from the PLC. For example, if the PLC changes the Trigger bit (from Input Flags, above) from a 0 to a 1, the vision sensor will change the Trigger ACK bit from a 0 to a 1 in response.

### PLC Input Register 2 or Holding Register 1002, also Inputs 10017-32

| Input 32 | Input 31 | Input 30 | Input 29 | Input 28 | Input 27 | Input 26 | Input 25 | Input 24 | Input 23 | Input 22 | Input 21 | Input 20 | Input 19 | Input 18 | Input 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Tool String Change ACK | BCR String Change ACK | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Product Change ACK | Remote Teach ACK | Trigger ACK |

# INPUT FLAGS

These write-able bits are inputs to the PresencePLUS vision sensor (outputs from the PLC). They are used for basic control of the vision sensor. They are also accessible as Coils 00001-16.

### PLC Holding Register 1, also Coils 00001-16

| Coil 16 | Coil 15 | Coil 14 | Coil 13 | Coil 12 | Coil 11 | Coil 10 | Coil 9 | Coil 8 | Coil 7 | Coil 6 | Coil 5 | Coil 4 | Coil 3 | Coil 2 | Coil 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Tool String Change | BCR String Change | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Product Change | Remote Teach | Trigger |

# TOOL STRING UPDATE FLAGS

These write-able bits are inputs to a PresencePLUS vision sensor (outputs from the PLC). These are only used when the vision sensor in question has BCR or OCR/OCV tools (and thus makes use of ASCII string regions). See Chapter 6 for more information on the String Update regions.

### PLC Holding Register 3 (Barcode/OCR/OCV only)

| bit 15 | bit 14 | bit 13 | bit 12 | bit 11 | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Region 6 Mask | Region 5 Mask | Region 4 Mask | Region 3 Mask | Region 2 Mask | Region 1 Mask | *reserved* | *reserved* | *reserved* | *reserved* | Region 6 Data | Region 5 Data | Region 4 Data | Region 3 Data | Region 2 Data | Region 1 Data |

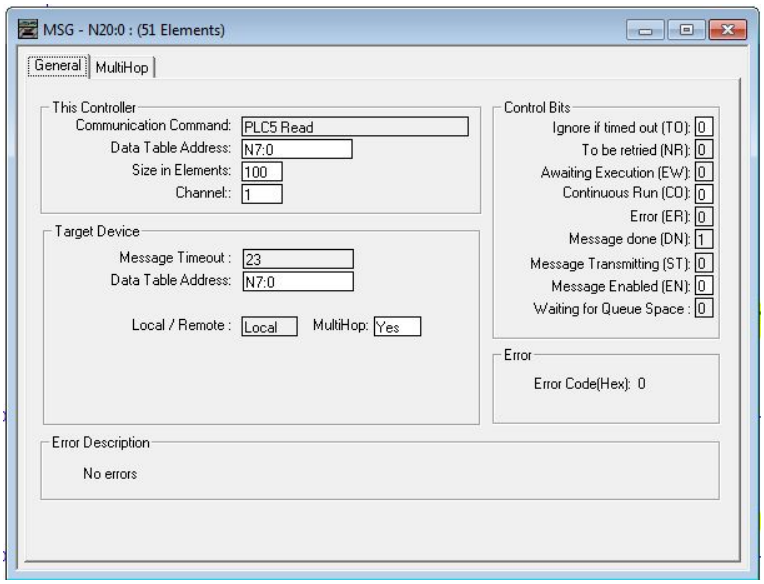
# Chapter 5: PCCC

Allen-Bradley's PLC5 and SLC 500 family of devices use PCCC communications protocol. The PresencePLUS vision sensor will support these PLCs using input and output register arrays.

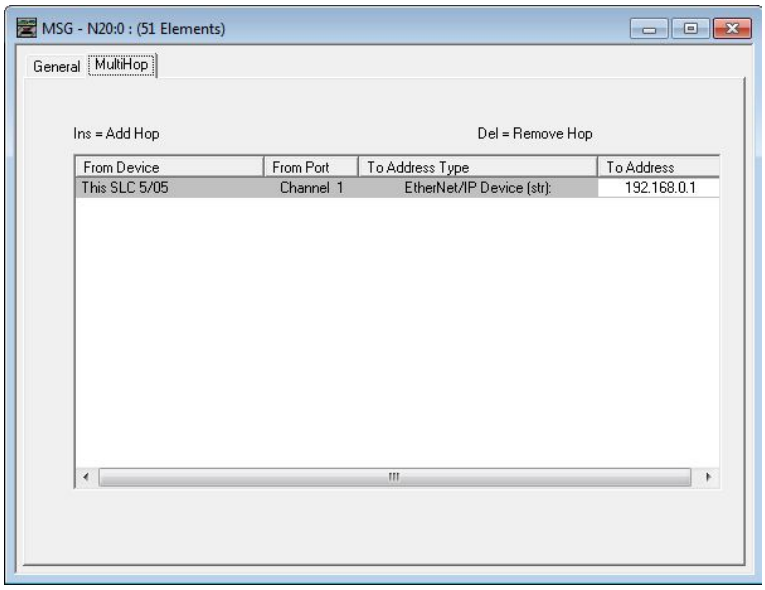
## 5.1 PLC Configuration

The images below represent a typical configuration:

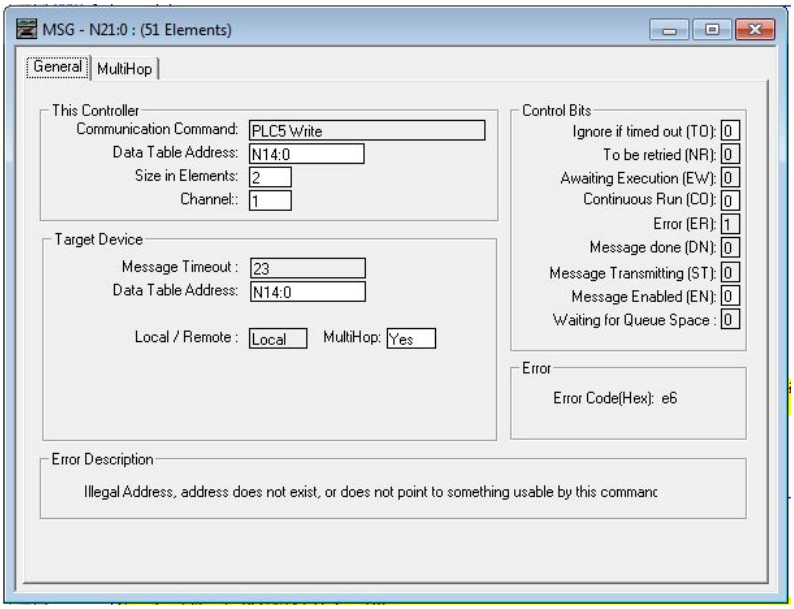
1. Read. Message command reading from N7 table on the vision sensor. We are keeping things easy by copying the vision sensor's N7 table to the PLC's N7 table. You could copy the sensor's N7 table to any Integer (N-type) table you wish.



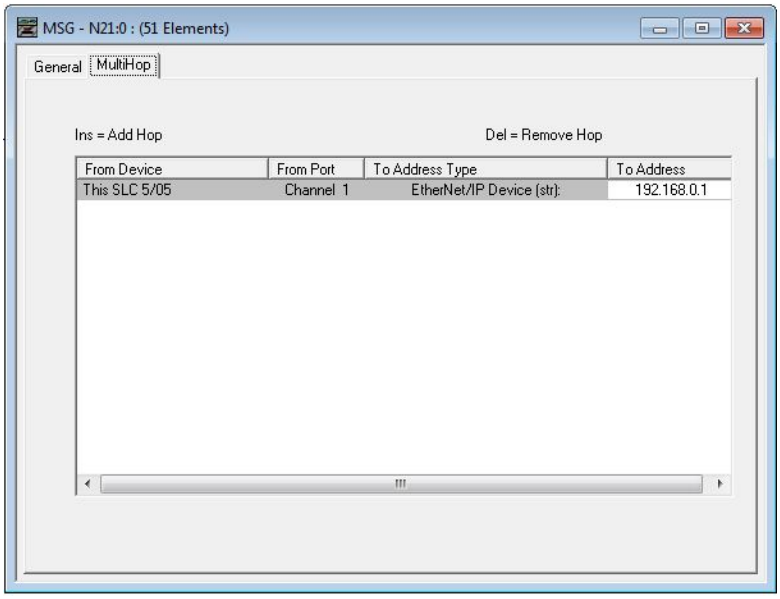

2. Read. IP Address of the vision sensor is entered here.



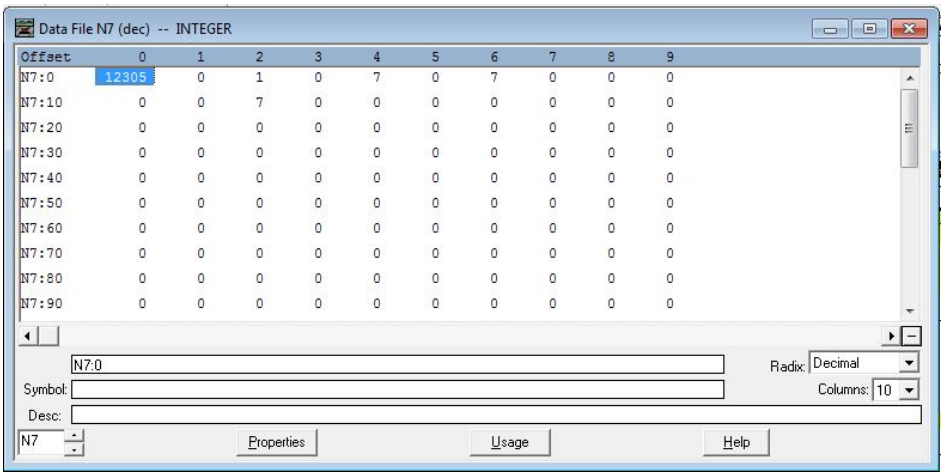
3. Write. Message command writing to the vision sensor's N14 table. We are again keeping things simple by writing the PLC's N14 table to the vision sensor's N14 table. You could write and N-type integer table's values from the PLC to the sensor's N14 table, however.




4. Write. IP Address of the vision sensor is entered here.



5. Here is a sample of the data read by the PLC (in this case we read from the camera's N7 table and placed the data into the PLC's N7 table).




In this case we can tell that the current Pass Count is "7", as a "7" appears as a 32-bit integer in register 6-7. Notice how word 0 (in the upper left corner of the array) reads "12305". This is the Output Flags register, and it is not meant to be read as a single 16-bit integer value. This decimal value equates to binary bits 0, 4, 12, and 13 being "1". This shows that the camera passed its last inspection (bit 0 = 1), is currently Ready (bit 4 = 1), and is a P4 (bits 12 & 13 are for I/O 5 and I/O 6, which are not defined on the P4 platform). See section 5.4 for definitions of the Output Flags register.

## 5.2 Outputs from Vision Sensor (Inputs to PLC)

The Output registers are used to push output values from the vision sensor to the PLC. MSG (message) commands are used to Read the vision sensor's N7 table.

**N7 Table**

**N7: PresencePLUS Output Registers (PCCC Input Registers)**

| REG # | WORD NAME | DATA TYPE |
|---|---|---|
| 0 | Output Flags (see Flags, section 5.4) | 16-bit integer |
| 1 | ACK Flags (see Flags, section 5.4) | 16-bit integer |
| 2 | Inspection Number | 16-bit integer |
| 3 | System Error Count | 16-bit integer |
| 4-5 | Frame Count | 32-bit integer |
| 6-7 | Pass Count | 32-bit integer |
| 8-9 | Fail Count | 32-bit integer |
| 10-11 | Missed Triggers | 32-bit integer |
| 12-13 | Iteration Count | 32-bit integer |
| 14-31 | *reserved* | 16-bit integer |
| 32-33 | Location 1 | 32-bit integer |
| 34-35 | Location 2 | 32-bit integer |
| 36-37 | Location 3 | 32-bit integer |
| … | … | … |
| 956-957 | Location 463 | 32-bit integer |
| 958-959 | Location 464 | 32-bit integer |

*Register Number = 32 + ((Location -1)*2)*

## 5.3 Inputs to the Vision Sensor (Outputs from PLC)

The Input registers are used to push input values from the PLC to the vision sensor. MSG (message) commands are used to Write values to the vision sensor's N14 table.

**N14 Table**

**N14: PresencePLUS Input Registers (PCCC Output Registers)**

| REG # | WORD NAME | DATA TYPE |
|---|---|---|
| 0 | Input Flags (see Flags, section 5.4) | 16-bit integer |
| 1 | Product Select | 16-bit integer |
| 2 | Tool String Update Flags (see Flags, section 5.4)* | 16-bit integer |
| 3-38 | *reserved* | 16-bit integer |
| 39-238 | String Input Regions (see Chapter 6)* | *varies* |

**These registers are only defined for vision sensors with BCR and/or OCR/OCV tools.*

## 5.4 Flags

### OUTPUT FLAGS

These bits are outputs from the PresencePLUS vision sensor (inputs to the PLC). They are used to report the basic status of the vision sensor and the last inspection run.

**N7, Word #0**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *reserved* | *reserved* | I/O 6 | I/O 5 | I/O 4 | I/O 3 | I/O 2 | I/O 1 | *reserved* | *reserved* | *reserved* | Ready | *reserved* | Error | Fail | Pass |

### ACK FLAGS

These bits are outputs from the PresencePLUS vision sensor (inputs to the PLC). They are used to acknowledge each of the input flags sent from the PLC. For example, if the PLC changes the Trigger bit (from Input Flags, above) from a 0 to a 1, the vision sensor will change the Trigger ACK bit from a 0 to a 1 in response.

**N7, Word #1**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Tool String Change ACK | BCR String Change ACK | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Product Change ACK | Remote Teach ACK | Trigger ACK |

### INPUT FLAGS

These bits are inputs to the PresencePLUS vision sensor (outputs from the PLC). They are used for basic control of the vision sensor.

**N14, Word #0**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Tool String Change | BCR String Change | *reserved* | *reserved* | *reserved* | *reserved* | *reserved* | Product Change | Remote Teach | Trigger |

### TOOL STRING UPDATE FLAGS

These bits are inputs to a PresencePLUS vision sensor (outputs from the PLC). These are only used when the vision sensor in question has BCR or OCR/OCV tools (and thus makes use of ASCII string regions). See Chapter 6 for more information on the String Update regions.

**N14, Word #2 (Barcode/OCR/OCV only)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Region 6 Mask | Region 5 Mask | Region 4 Mask | Region 3 Mask | Region 2 Mask | Region 1 Mask | *reserved* | *reserved* | *reserved* | *reserved* | Region 6 Data | Region 5 Data | Region 4 Data | Region 3 Data | Region 2 Data | Region 1 Data |

# Chapter 6: String Input Regions

PresencePLUS vision sensors that include a BCR and/or OCR/OCV tools also have the String tool. String Input Regions in the memory map allow for the changing the String tool's operands and the OCV tool's Expected String input (when that variable is set to Industrial Ethernet in the OCV tool) via the PLC or HMI.

## *6.1 Region Definitions*

There are a total of six string regions. Regions one and two are each 50 registers long, allowing strings of up to 98 ASCII characters (as the first two registers of the region are used to denote the length of the string in ASCII characters). Regions three through six are each 25 registers long, allowing for strings of up to 48 ASCII characters each. Maximum length of strings stored in each of the regions depends on the tool and the specific operand chosen for that tool. The six regions are described below:

- Region 1
  - 50 registers long
  - Starting register is 40 for Modbus/TCP, 39 for EtherNet/IP and PCCC
  - For OCV tool and String tool with the Compare or Find Substring operation:
    - Holds strings of up to 98 ASCII characters long
  - For String tool with Masked String Comparison operation:
    - String and its corresponding mask are each 25 registers long
    - Holds string of up to 48 ASCII characters long
    - Holds string masks of up to 48 characters long
    - Starting register for the Region 1 Mask is 65 for Modbus/TCP, 64 for EtherNet/IP and PCCC

- Region 2
  - 50 registers long
  - Starting register is 90 for Modbus/TCP, 89 for EtherNet/IP and PCCC
  - For OCV tool and String tool with the Compare or Find Substring operation:
    - Holds strings of up to 98 ASCII characters long
  - For String tool with Masked String Comparison operation:
    - String and its corresponding mask are each 25 registers long
    - Holds string of up to 48 ASCII characters long
    - Holds string masks of up to 48 characters long
    - Starting register for the Region 1 Mask is 115 for Modbus/TCP, 114 for EtherNet/IP and PCCC

- Region 3
  - 25 registers long
  - Starting register is 140 for Modbus/TCP, 139 for EtherNet/IP and PCCC
  - For OCV tool and String tool with the Compare or Find Substring operation:
    - Holds strings of up to 48 ASCII characters long
  - For String tool with Masked String Comparison operation:
    - String and its corresponding mask are each 12 registers long
    - Holds string of up to 22 ASCII characters long
    - Holds string masks of up to 22 characters long

        - Starting register for the Region 1 Mask is 152 for Modbus/TCP, 151 for EtherNet/IP and PCCC

- Region 4
    - 25 registers long
    - Starting register is 190 for Modbus/TCP, 189 for EtherNet/IP and PCCC
    - For OCV tool and String tool with the Compare or Find Substring operation:
        - Holds strings of up to 48 ASCII characters long
    - For String tool with Masked String Comparison operation:
        - String and its corresponding mask are each 12 registers long
        - Holds string of up to 22 ASCII characters long
        - Holds string masks of up to 22 characters long
        - Starting register for the Region 1 Mask is 177 for Modbus/TCP, 176 for EtherNet/IP and PCCC

- Region 5
    - 25 registers long
    - Starting register is 190 for Modbus/TCP, 189 for EtherNet/IP and PCCC
    - For OCV tool and String tool with the Compare or Find Substring operation:
        - Holds strings of up to 48 ASCII characters long
    - For String tool with Masked String Comparison operation:
        - String and its corresponding mask are each 12 registers long
        - Holds string of up to 22 ASCII characters long
        - Holds string masks of up to 22 characters long
        - Starting register for the Region 1 Mask is 202 for Modbus/TCP, 201 for EtherNet/IP and PCCC

- Region 6
    - 25 registers long
    - Starting register is 215 for Modbus/TCP, 214 for EtherNet/IP and PCCC
    - For OCV tool and String tool with the Compare or Find Substring operation:
        - Holds strings of up to 48 ASCII characters long
    - For String tool with Masked String Comparison operation:
        - String and its corresponding mask are each 12 registers long
        - Holds string of up to 22 ASCII characters long
        - Holds string masks of up to 22 characters long
        - Starting register for the Region 1 Mask is 227 for Modbus/TCP, 226 for EtherNet/IP and PCCC

### 6.2 Specifying Updated Regions

To allow for the individual updating of one or more of the six possible string regions, we make use of the Tool String Update Flags register. The bits making up this 16-bit word are used to tell the PresencePLUS vision sensor which particular regions or masks should update their ASCII string data. Changing one or more of the bits from a 0 to a 1 signifies that that region or mask will be included in the update once the Tool String Change bit is toggled (bit 9 in the Input Flags register). Multiple regions can be changed in a single update. The user should clear these bits once the Tool String Change ACK bit is received from the PresencePLUS vision sensor.

### 6.3 Applying Region Updates

When the Tool String Change bit is set, new ASCII data for all the modified string regions, as defined by the Tool String Update Flags register, will be copied to the vision sensor's internal buffer. After the copy operation is complete, the vision sensor will set the Tool String Change ACK bit in the Output Flags register to indicate the new data has been received.

The updated ASCII string data will be retained in the internal buffer. They will be applied to their respective tools only when the next trigger is received. Please note that by setting and clearing this flag multiple times between triggers, the data copied into the sensor's internal buffers may be overwritten, without ever being copied into the inspection.

Users should reset the Tool String Update flag after the data copy acknowledgement has been received from the sensor.

A corresponding Tool String ACK flag appears in the ACK Flags register.

### 6.4 Changing the String Input Regions

Operands in the String tool and the 'Expected String' value in the OCV tool can be modified remotely via an Industrial Ethernet connection. Prior to attempting to do this, the desired String tool operand and/or the OCV tool 'Expected String' must be associated with one of the six string Regions. To create this association in the PresencePLUS software, select "Industrial Ethernet" as the expected string source in the OCV tool drop-down menu, or select "Industrial Ethernet" as the input tool for one of the String tool's operands. After this step is complete, a new drop-down menu listing all the available string regions will be displayed. A suitable region should then be selected from that menu. After this operation is complete, the selected string region will be associated with that tool. Note that it is possible to associate a given region with more than one tool. The resulting inspection should then be saved to the vision sensor's permanent flash memory then enabled on the sensor.

To make use of the desired string region, follow these steps:

- Write a new ASCII string into the register space assigned to the string region you are updating.
    - More than one region and more than one mask may be updated at one time.

- Optionally, write a new mask into the correct register space. The mask is only valid if a String tool with Masked String Comparison operation has been created and is associated with a string region.
- Set the bits in the Tool String Update flags register corresponding to the updated regions and masks.
- Toggle the Tool String Change bit in the Input Flags register.
- When you see the Tool String Change ACK bit go from 0 to 1 in the ACK Flags register, toggle the Tool String Change bit back to 0.
- Toggling the Tool String Change bit causes the PresencePLUS vision sensor to copy the new strings and masks into the temporary internal buffer. Use the Tool String Change ACK bit to verify that the change request was received by the vision sensor.

When the next trigger is received, the strings and masks (if present) are validated and copied into the tools that are associated with the updated regions.

To be validated, the new data stored in the register space must meet both requirements:

- Both the string and mask (if present) must be stored in the format documented above – length followed by ASCII string data. Length is the number of ASCII characters included in the string (not including any null termination characters).
- The string length can be different than the string length of the original inspection.

The mask is optional and is only examined if the region is associated with a String tool with Masked String Comparison operation selected. If the mask length is non-zero, and the mask length equals the new compare string length, it will be used. The mask is a character string of the same length as the compare string. It is a character by character binary mask, wherein the following characters are valid:

- '0' (0x30) – wild card
- '1' (0x31) – check this character

The mask is a string of '0' and '1' characters, indicating which character postions in the compare string should be compared against. Any characters other than '0' and '1' in the mask will cause it to be flagged as invalid.

An example mask (using the Standard String Format) would be:

[00 05| 0x31 0x31 | 0x31 0x31 | 0x30 00]

In this example, the mask (and thus the compare string) are 5 ASCII characters long. The first 4 characters will be compared to the compare string, and the fifth character is a wild card where any data character will be counted as a match.

# Chapter 7: Data Types

The PresencePLUS vision sensor stores data as one of 4 data types

- 16-bit integer
- 32-bit integer
- Floating point
- ASCII character string

## 5.1 16-bit Integer

The vision sensor stores the following data as 16-bit integers:

- Output Flags (meant to be read as 16 individual bits)
- ACK Flags (meant to be read as 16 individual bits)
- Inspection Number
- System Error Count
- Input Flags (meant to be read as 16 individual bits)
- Product Select

## 5.2 32-bit Integer

The vision sensor stores the following data as 32-bit (unsigned) integers:

- Frame Count
- Pass Count
- Fail Count
- Missed Trigger Count
- Iteration Count

In addition, some tool results, such as counts, as stored as 32-bit integers. Refer to the inspection PLC map, generated by the Communication tool, for more information (see Chapter 2).

32-bit integers are stored in two adjacent 16-bit words/registers. The data can be formatted in two ways, depending on the '32-bit Format' setting (set in the PresencePLUS software→ System button, Communication tab, Communication Setup Channel = Industrial Ethernet).

**LSW-MSW**
*Least Significant Word-Most Significant Word*. In this 32-bit format, the Least Significant Word will be stored first, in the lowest numbered register.

**MSW-LSW**
*Most Significant Word-Least Significant Word.* In this 32-bit format, the Most Significant Word will be stored first, in the lowest numbered register.

## 5.3 Floating Point

The vision sensor stores the following data as 32-bit (unsigned) integers:

- Execution Time

- Distances

The floating point values used by the vision sensor are 32-bit single precision floating point numbers defined in IEEE-754.

Floating point values are stored in two adjacent 16-bit words/registers. The data can be formatted in two ways, depending on the '32-bit Format' setting (set in the PresencePLUS software→ System button, Communication tab, Communication Setup Channel = Industrial Ethernet).

**LSW-MSW**

*Least Significant Word-Most Significant Word*. In this 32-bit format, the Least Significant Word will be stored first, in the lowest numbered register.

- Bits 15-0 of first register = Bits 31-16 of the floating point number
- Bits 15-0 of the second register = Bits 15-0 of the floating point number

**MSW-LSW**

*Most Significant Word-Least Significant Word.* In this 32-bit format, the Most Significant Word will be stored first, in the lowest numbered register.

- Bits 15-0 of first register = Bits 15-0 of the floating point number
- Bits 15-0 of the second register = Bits 31-16 of the floating point number

## 5.4 ASCII Character String

The vision sensor stores some information as strings of ASCII characters, including:

- Tool names
- Barcode data read
- OCR data read

Character strings consist of a length value (describing how many ASCII characters are in the string) followed by a variable number of 8-bit character bytes. The NULL byte (0x00) is not included in the length. The entire storage area is filled with zeroes before the string is written into it. Therefore, if there is sufficient storage space allocated for the string data, the string will be NULL terminated.

**NOTE:** When making vision tool string output data available to the PLC using a Communication tool, the user selects the number of strings to store and the maximum size of each string. If insufficient storage space is allocated, strings are truncated.

There are two string format options in the PresencePLUS vision sensor. The type used depends on the choice of Industrial Ethernet Protocol selected (PresencePLUS software→ System button, Communication tab, Communication Setup Channel = Industrial Ethernet, Protocol). The difference between these two options has to do with the size of the integer used to denote the string length.

In addition to a Protocol-based difference in how the length of the ASCII string is reported, there is also a user controlled setting defining how the ASCII characters are packed into each register. This choice relates to whether the first ASCII character in the string should be stored in the "high byte" of the 16-bit integer (i.e. bits 8-15) or the "low byte" (bits 0-7). This setting is found in the PresencePLUS software→ System button, Communication tab, Communication Setup Channel = Industrial Ethernet, Character String Order.

When Modbus/TCP or PCCC are selected, characters are packed into strings that feature a 16-bit integer for the length. This integer records how many ASCII characters are going to be placed in the registers that follow. Furthermore, the default setting for Character String Order is 'High Byte- Low Byte'.

***Standard String Format***

**High Byte-Low Byte, 16-bit integer length (Modbus/TCP, PCCC default)**

| Word 0 | Word 1 | | Word 2 | | Word 3 | | Word 4 | | … |
|---|---|---|---|---|---|---|---|---|---|
| 16-bit length | byte 0 | byte 1 | byte 3 | byte 4 | byte 5 | byte 6 | byte 7 | byte 8 | … |

The alternative ASCII character packing scheme (Low Byte-High Byte) reverses the order of the ASCII characters.

**Low Byte-High Byte, 16-bit integer length**

| Word 0 | Word 1 | | Word 2 | | Word 3 | | Word 4 | | … |
|---|---|---|---|---|---|---|---|---|---|
| 16-bit length | byte 1 | byte 0 | byte 4 | byte 3 | byte 6 | byte 5 | byte 8 | byte 7 | … |

When EtherNet/IP is selected as the protocol, the vision sensor always uses the so-called ControlLogix string format, as defined by Allen Bradley ControlLogix PLCs native string data type. In this format, the string uses a 32-bit integer for the length variable and the Low Byte-High Byte option for Character String Order.

***ControlLogix String Format***

**32-bit integer length, Low Byte-High Byte (EtherNet/IP default)**

| Word 0 | Word 1 | Word 2 | | Word 3 | | Word 4 | | … |
|---|---|---|---|---|---|---|---|---|
| 32-bit length LSW | 32-bit length MSW | byte 1 | byte 0 | byte 3 | byte 2 | byte 5 | byte 4 | … |