Communications and Programming Guide Checker 4G

User manual

HSD 0290.00 2.- 511e

Contents

About this Manual

The *Checker 4G Communications and Programming Guide* provides information about how to integrate a Checker 4G sensor into your particular environment, including:

- Network configuration
- Industrial network protocols
- Integration with PLCs
- Port usage

Networking

You can connect your Checker 4G sensor via a simple Ethernet connection. The Checker 4G sensor can be on the same subnet as your PC or it can be on a different subnet.

If your Checker 4G sensor is on the same subnet as your machine, it supports the following Ethernet network assignments:

- Link Local Addressing
- DHCP client
- Static IP
- No IP address

Checker 4G sensors are by default configured to use DHCP. You can, however, assign a static IP address to them, too.

Automatic IP Address Assignment

When the Checker 4G sensor is configured to support a DHCP server (default), the sensor first checks if it is directly connected to a PC. If so, connection is established through Link Local Addressing. This reduces connection time in case no DHCP server is found.

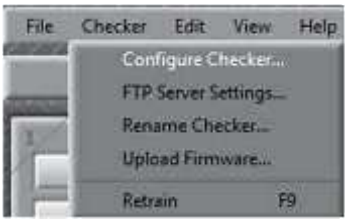
If no Link Local Addressing occurs, the Checker 4G sensor waits for a DHCP server to assign an address.

If the DHCP server cannot find a valid IP address, the Checker 4G sensor functions normally in run mode. The sensor continues to listen for an address assignment. During this time, however, no configuration is possible.

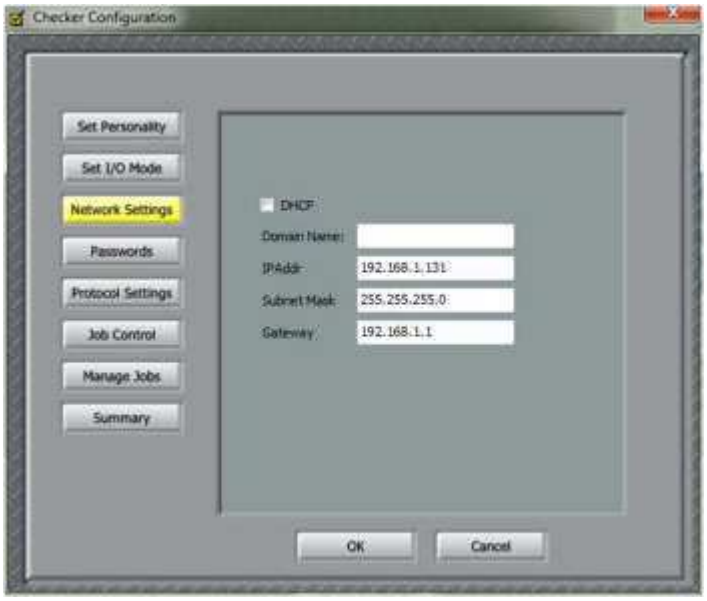
Static IP Address Assignment

You can configure your Checker 4G sensor to use a static IP address. In this case, the sensor only responds to the assigned address and does not use either Link Local Addressing or addresses assigned by the DHCP server. Your connection time, in this case, will be reduced.

1. In the main menu toolbar, select **Configure Checker**.



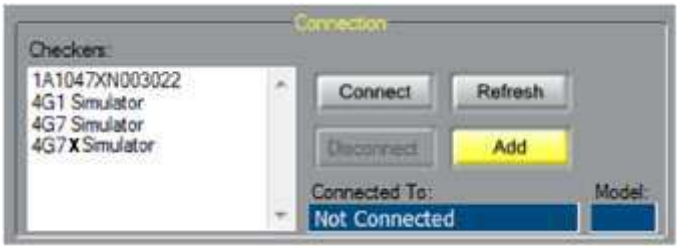
2. In the window that pops up, you can configure the IP settings according to your needs.



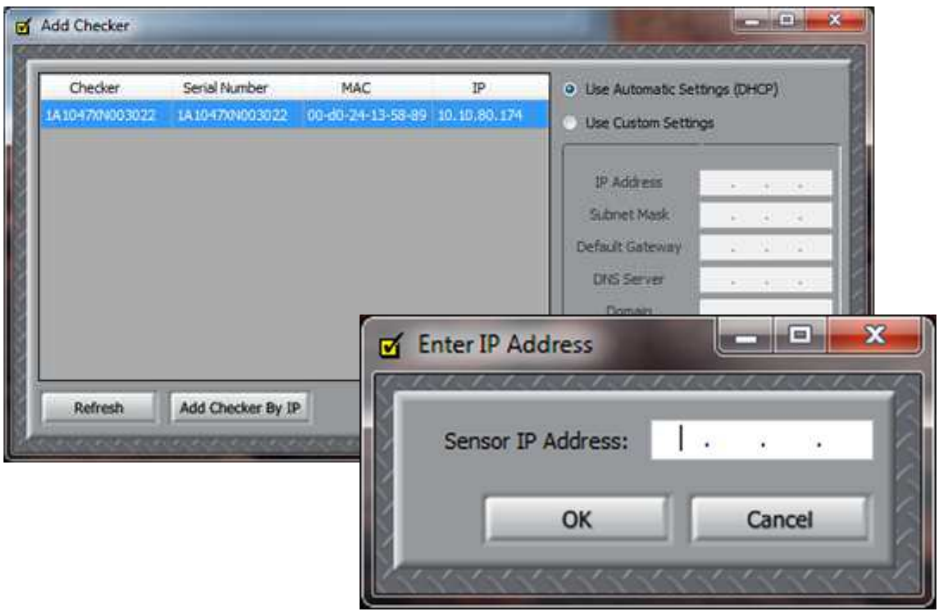
Connecting Your Checker 4G across Subnets

The following options can be used to connect to the Checker 4G sensor with the PC across subnets if you already know the IP Address of the Checker 4G sensor.

1. In the **Get Connected** step, click **Add** in the **Connection** frame.



2. In the **Add Checker** dialog box, click **Add Checker By IP**, then enter the IP address of the Checker 4G sensor.



3. Click **OK**, then close the **Add Checker** dialog box. The Checker should now appear in the **Checkers** list.

If your Checker is configured in a way that its IP address is assigned by a DHCP server, but is connected to a network where there is no DHCP server, then the Checker does not appear in the **Connection** list. Click **Add**. You will see your Checker sensor in the window that appears. Here you can further configure your device, if necessary.

If your Checker is configured in a way that its IP address is set to be static, but you connect it to a network where there is a DHCP server but the IP range is different, your Checker does not appear in the **Connection** list. Click **Add.** You will see your Checker sensor in the window that appears. Here you can further configure your device, if necessary.

Industrial Network Protocols

Checker 4G uses industrial network protocols that are based on standard Ethernet protocols. These protocols, such as EtherNet/IP™ and PROFINET, are enhanced to provide more reliability than standard Ethernet. A Generic Factory Floor Protocol (Generic FFP) is available for communication if neither EtherNet/IP nor PROFINET are supported.

By default, industrial network protocols are disabled on Checker. To enable a protocol, connect to Checker, select **Checker->Configure Checker…**, then click **Protocol Settings** and enable the desired protocol.

EtherNet/IP

EtherNet/IP is a communication system suitable for use in industrial environments. EtherNet/IP allows industrial devices to exchange time-critical application information. These devices include simple I/O devices such as sensors/actuators, as well as complex control devices such as robots, programmable logic controllers, welders, and process controllers.

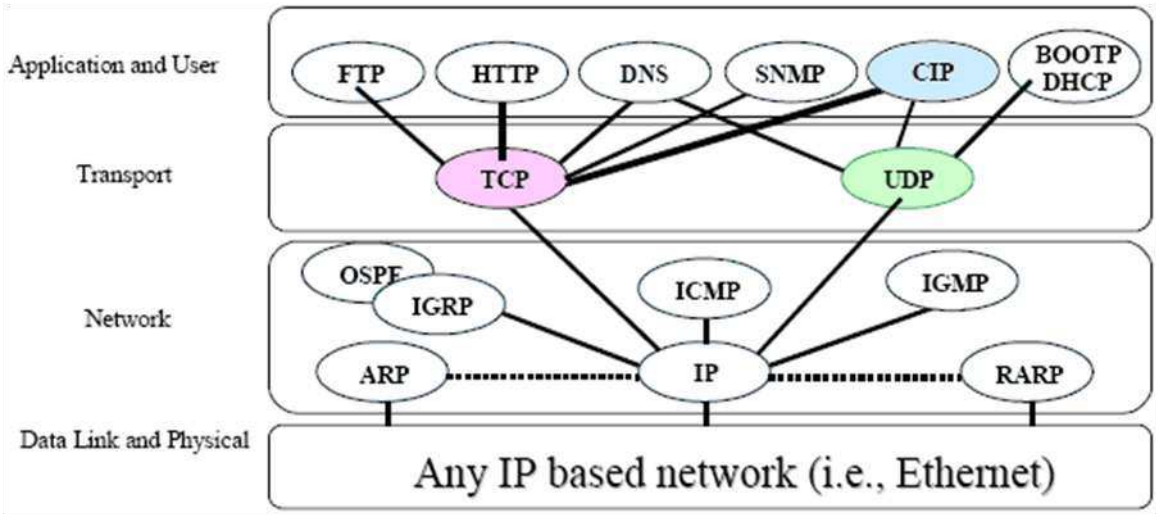
To understand Ethernet/IP you must understand the Common Industrial Protocol (CIP). Ethernet/IP can best be described as both an extension and wrapper of the Common Industrial Protocol (CIP). It operates over standard IEEE 802.3 Ethernet using the TCP/IP protocol suite.

The Ethernet/IP implementation used by di-soric sits directly above the Ethernet stack and uses ordinary sockets functionality. There are no non-standard additions that attempt to improve determinism.

Common Industrial Protocol (CIP)

The Common Industrial Protocol (CIP) is a media independent, connection-based, object-oriented protocol designed for automation applications. It encompasses a comprehensive set of communication services for automation applications: control, safety, synchronization, motion, configuration and information. Currently CIP forms the core of EtherNet/IP, DeviceNet, CompoNet and ControlNet.

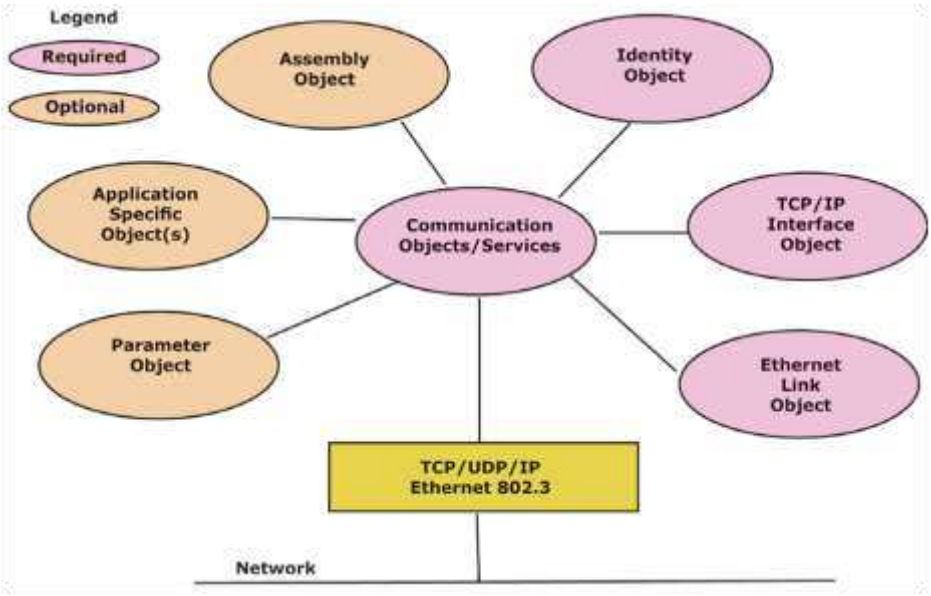
On Ethernet CIP straddles both UDP and TCP. CIP “explicit” messaging uses TCP to pass non-time critical point-to-point messages and commands from one device to another. CIP “implicit” messaging uses UDP for time-critical control and I/O.



CIP is based on an object model. These objects encapsulate a device’s data and functionality (at least those portions you wish to expose to the outside world). The model provides a common way to access this data and functionality.

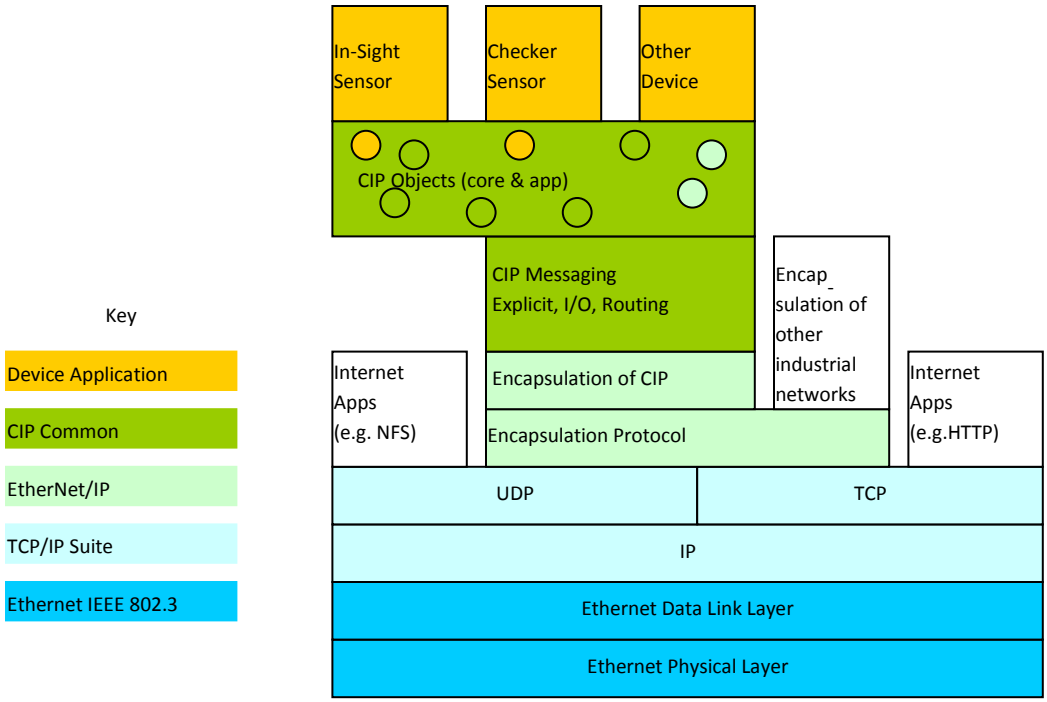
A number of predefined objects are required in all EtherNet/IP implementations. These primarily deal with device identity and communications. Other predefined objects may be optionally included. The CIP specification contains a complete library of predefined objects (such as Assembly, Discrete Output, Analog Input, AC/DC drive, and so on). Vendors may also create their own application specific objects.

For an illustration of the EtherNet/IP simplified object model, see the following figure.



EtherNet/IP Implementation of CIP

For the relationship between CIP, EtherNet/IP, and TCP/IP, see the following figure. The primary components of EtherNet/IP are two new CIP objects and an encapsulation layer.



The encapsulation layer is a wrapper which allows native CIP to pass across standard IEEE 802.3 Ethernet using the TCP/IP protocol suite. It also provides extensions to CIP routing functionality to deal with the unique aspects of Ethernet.

Two new CIP objects have been added to support EtherNet/IP; the TCP/IP Interface Object and the Ethernet Link Object. The Interface object provides the mechanism to configure a device's TCP/IP network interface (IP address, network mask, and so on). The link object maintains link-specific configuration, status and diagnostic information for an Ethernet 802.3 communications interface.

CIP Messaging Types

CIP provides "implicit" messaging for real-time I/O or control. CIP provides "explicit" messaging for non-time critical data transfers or informational messages. Currently Checker supports only implicit messages.

Transmission Type	Message Type	Description	Example
Information	Explicit	Non-time-critical information	Request/response command (such as setting a configuration parameter value)

Transmission Type	Message Type	Description	Example
I/O data	Implicit	Real-time I/O data	Real-time data from remote I/O device (such as sensor, switch, or motor control)

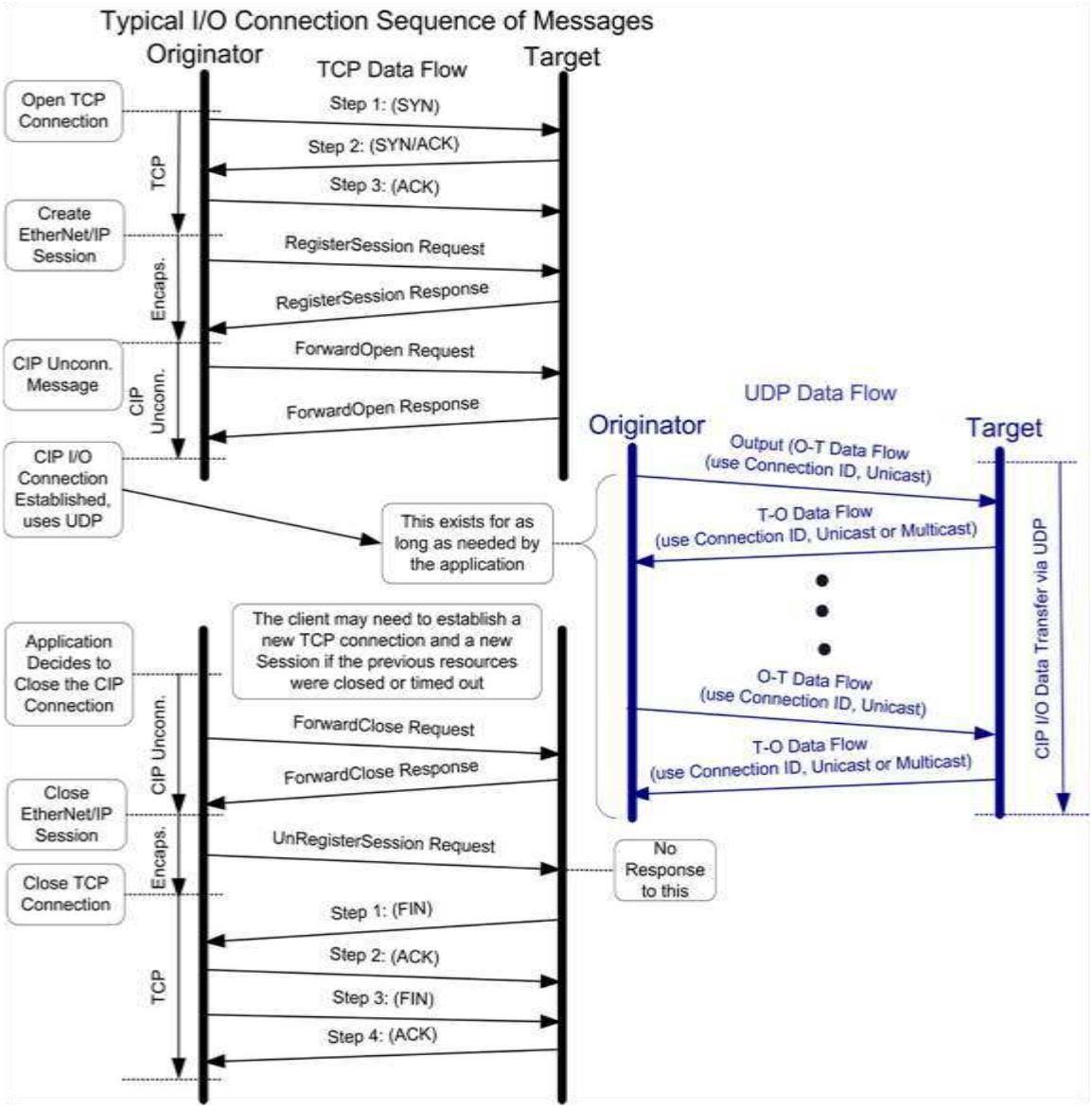
Implicit Messaging

Often referred to as "I/O", this type of communication is typically used for real-time data exchange, where speed and low latency are important. Implicit messages include very little information about their meaning, so the transmission is more efficient, but less flexible than explicit. The interpretation of the transmitted data is fast.

With Implicit Messaging you establish an association (a "CIP connection") between two devices. After the connection is established, a device will produce the Implicit Messages according to a predetermined trigger mechanism. Triggers can be Cyclic (most common), Change of State (CoS) or application-specific. Both devices know and agree on the data formats they will use (that is, the format is "implied"). For EtherNet/IP, Implicit Messaging uses UDP and can be multicast or unicast.

Implicit messaging are based on the assemblies. For detailed information on Assembly Objects, see Section Assembly Object.

For an illustration of a typical I/O connection sequence of messages, see the following figure.

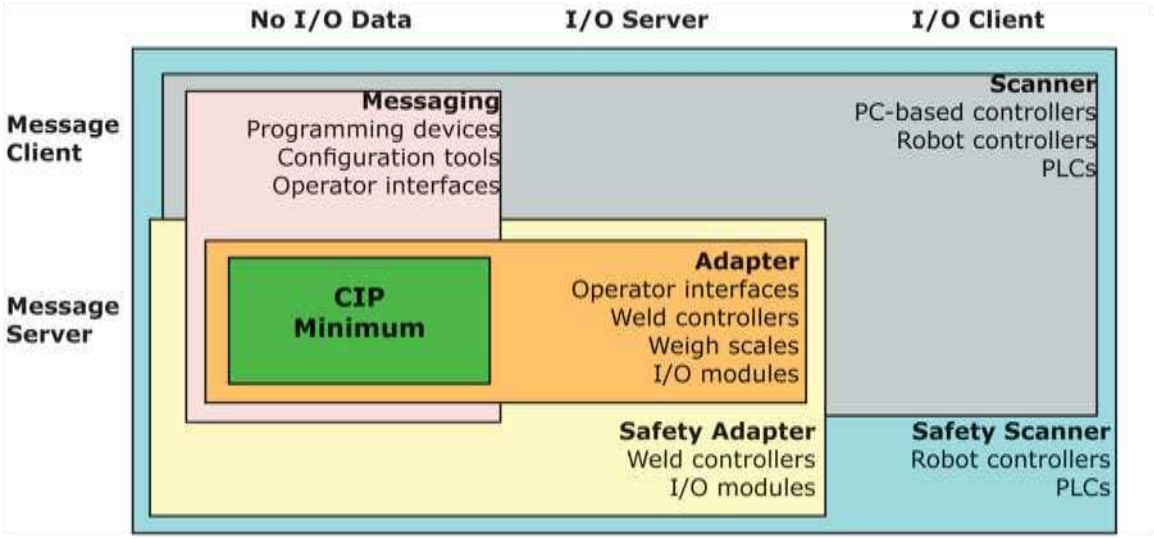


Device Classifications

There are several device classifications, based on general behavior and types of EtherNet/IP communications that they support. Most devices fit into one of the following classifications. However, certain cases require a device to be multi-classified.

All EtherNet/IP devices are required to have minimal explicit server capability, to respond to device identification and configuration requests.

For an illustration of a matrix of communication types and device classifications, see the following figure.



Explicit Message Server

An explicit message server responds to request/response oriented communications initiated by explicit message clients. An example of an explicit message server could be a kiosk scrolling text display.

Explicit Message Client

An explicit message client initiates request/response oriented communications with other devices. Message rates and latency requirements are typically not too demanding. Examples of explicit message clients are HMI devices, programming tools, or PC based applications that gather data from control devices.

I/O Adapter

An I/O adapter receives implicit communication connection requests from an I/O scanner then produces its I/O data at the requested rate. An I/O adapter is also an explicit message server. An I/O adapter can be a simple digital input device, or something more complex such as a modular pneumatic valve system.

I/O Scanner

An I/O scanner initiates implicit communications with I/O adapters. A scanner is typically the most complex type of EtherNet/IP device, as it must deal with issues such as configuration of which connections to make, and how to configure the adapter device. Scanners also typically support initiating explicit messages. A programmable controller (PLC) is an example of an I/O scanner.

Checker and EtherNet/IP

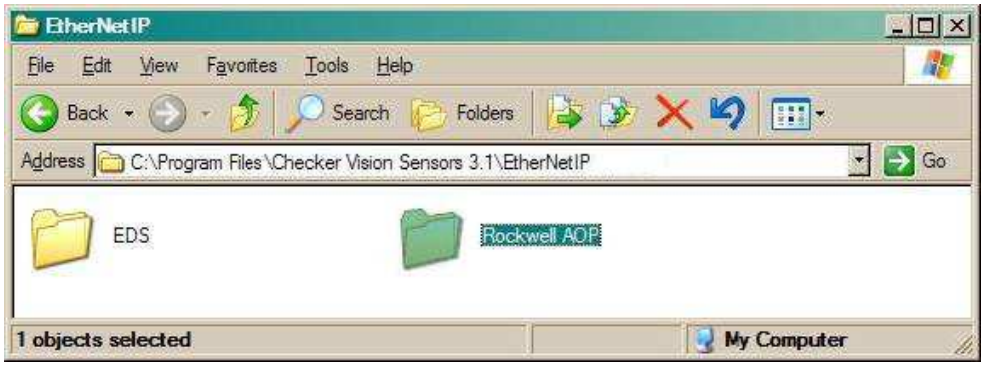
Checker 4G supports EtherNet/IP, an application level protocol based on the Common Industrial Protocol (CIP). EtherNet/IP provides an extensive range of messaging options and services for the transfer of data and I/O over Ethernet. All devices on an EtherNet/IP network present their data to the network as a series of data values called attributes. Attributes can be grouped with other related data values into sets, these are called Assemblies.

Enabling EtherNet/IP involves the following main steps:

- Make sure you have the Rockwell Software tool on your machine.
- Set up the Rockwell Software tool so that it recognizes your Checker sensor.
- Install the Checker Electronic Data Sheet (EDS) for the Checker sensor.

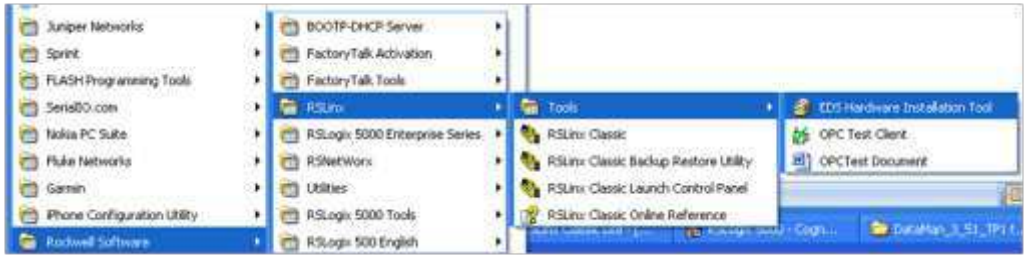
Perform the following steps to set up EtherNet/IP:

4. Verify that the Rockwell Software is on your PC.
5. Make sure you select the Add on Profile installation and the Samples installation. Add on Profile is only used with Rockwell ControlLogix or CompactLogix PLCs.
6. Install the Rockwell Add on Profiles by navigating to the following directory:



7. Select the Rockwell AOP directory.
8. If you have previously not installed the Rockwell AOP, now run MPSetup.exe from this directory.

9. From the Start menu, go to Programs → Rockwell Software → RSLinx → Tools → EDS Hardware Install Tool.



10. Run the ESD Install tool.

NOTE

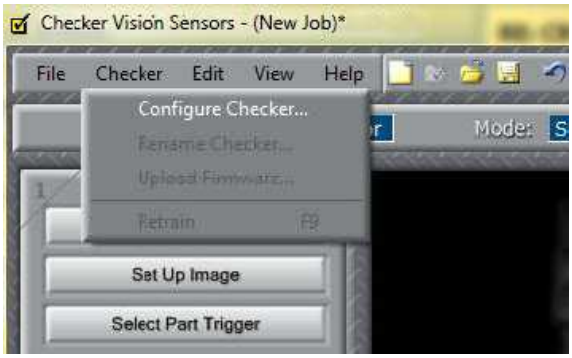
If you have an existing EDS file, uninstall it first, then install the latest version of the EDS.

11. Run Checker. If a window pops up notifying you that a firmware upgrade or downgrade is necessary, act accordingly. Otherwise, continue with the next step.

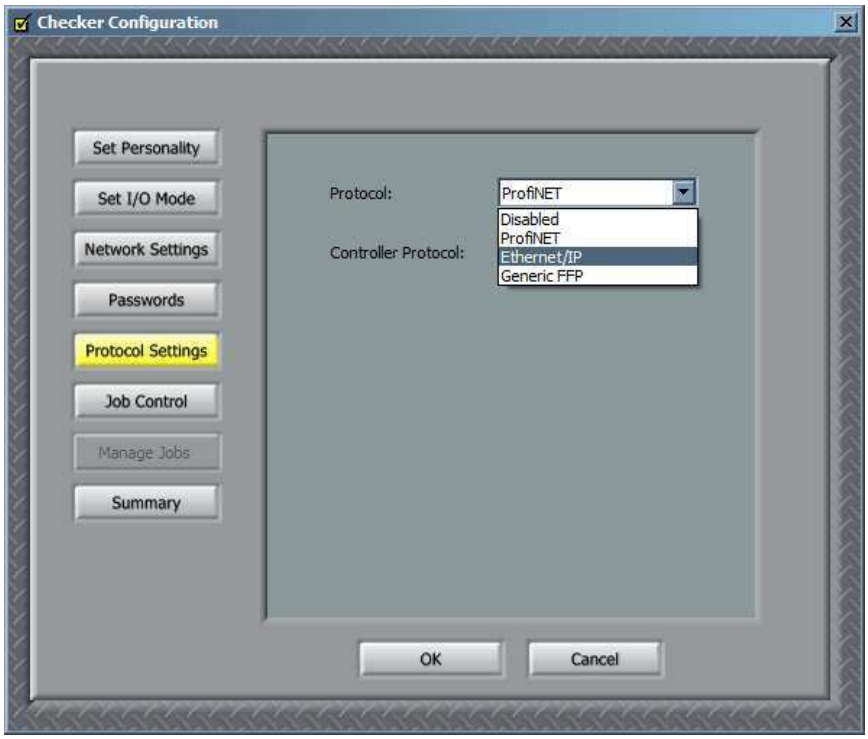
12. By default the Checker 4G has the EtherNet/IP protocol disabled. To enable the protocol, perform the following steps:

    a. In the upper menu toolbar, click Checker.

    b. Select Configure Checker.



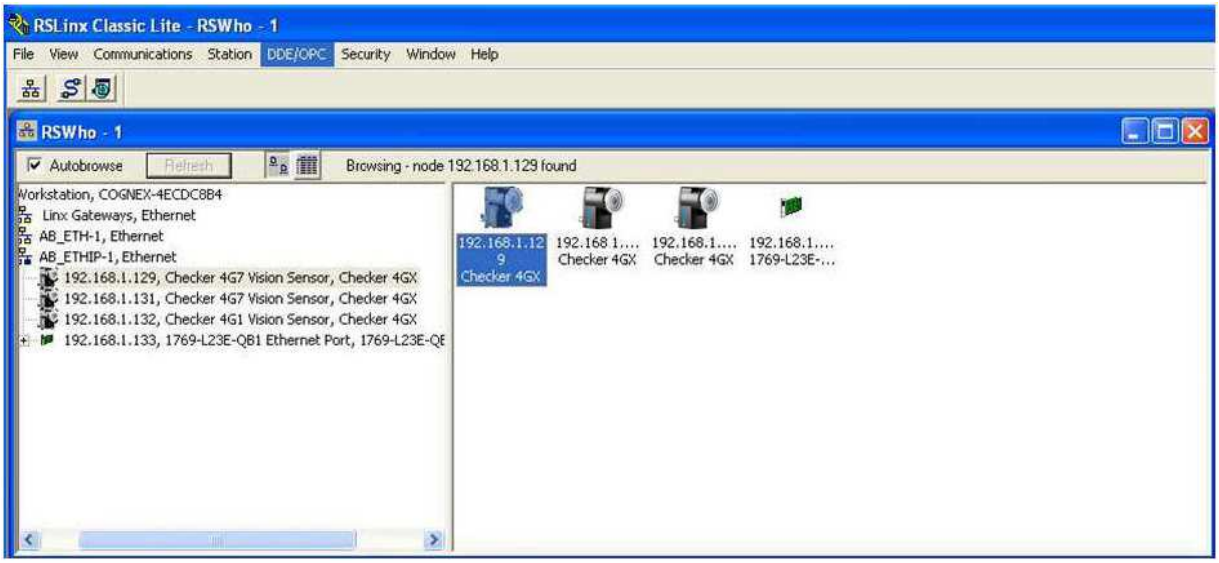
    c. In the Checker Configuration window that pops up, click Protocol Settings and select EtherNet/IP.



d. Click OK.

For these settings to take effect, the sensor automatically reboots.

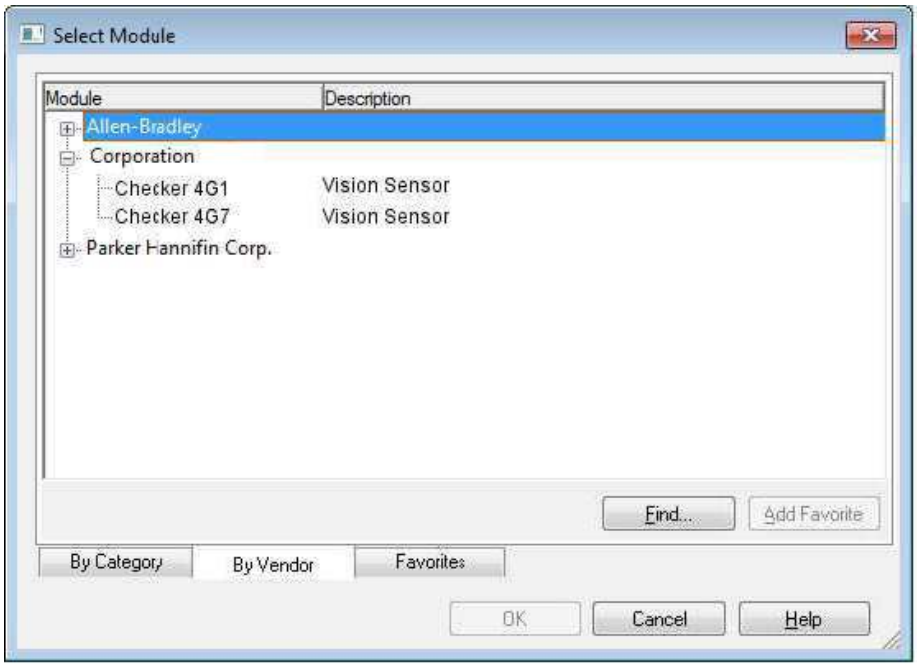
13. Your Checker sensor is now visible in the RSWHO.



If your Checker 4G sensor is visible, but the icon is a question mark, repeat the ESD Installation.

14. Open one of your jobs and integrate your Checker 4G sensor into your program using the Add on Profile.

Alternatively, you can add the Checker 4G sensor as a Module on your network.



Inputs

Inputs act as "virtual" inputs. When the value of an Input changes from 0 → 1 the action associated with the event will be executed. When the action completes the corresponding InputAck bit will change from 1 → 0 to signal completion. The acknowledge bit will change back to 0 when the corresponding Input bit is set back to 0.

Services

The Checker object supports the following Common CIP services:

Service Code	Service Name	Description
0x05	Reset	Resets the Checker object.
0x0E	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	Set_Attribute_Single	Modifies the specified attribute.

Acquire Service

The Acquire Service will cause an acquisition to be triggered (if the acquisition system is ready to acquire an image). If the acquisition could not be triggered, then the Missed Acquisition bit will be set until the next successful acquisition.

Acquisition Sequence

Checker can be triggered to acquire images implicitly via the Assembly object.

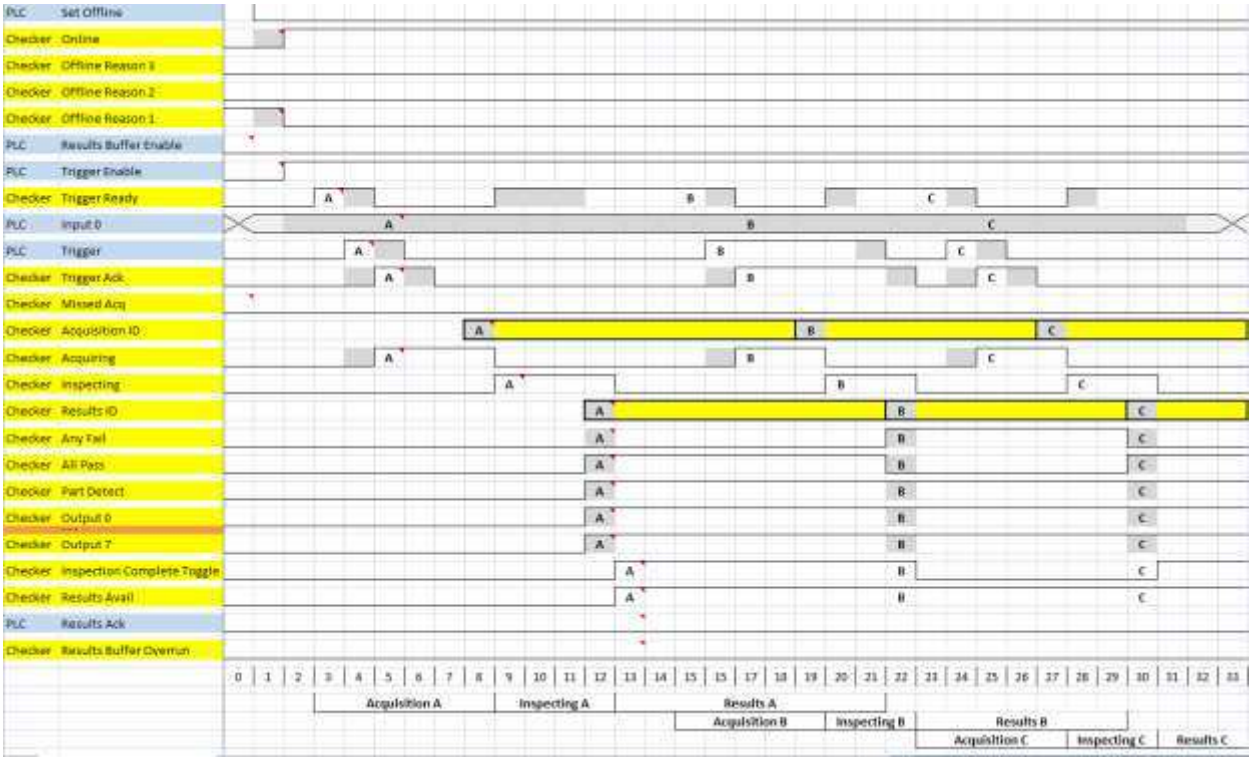
On startup the TriggerEnable attribute will be false. It must be set to true to enable triggering. When the device is ready to accept triggers, the Trigger Ready bit will be set to true.

While the TriggerEnable attribute is true and the Trigger Ready bit is true, each time the Checker object sees the Trigger attribute change from 0 to 1, it will initiate an image acquisition. When setting this via the assembly objects, the attribute should be held in the new state until that same state value is seen in the Trigger Ack bit (this is a necessary handshake to guarantee that the change is seen by the Checker object).

During an acquisition, the Trigger Ready bit will be cleared and the Acquiring bit will be set to true. When the acquisition is completed, the Acquiring bit will be cleared and the Trigger Ready bit will again be set true once the acquisition system is ready to begin a new image acquisition.

To force a reset of the trigger mechanism set the TriggerEnable attribute to false, until the Trigger Ready is not set to 0. Then, TriggerEnable can be set to true to re-enable acquisition.

Triggering is only available when Checker is in External Trigger Mode. See the following figure for a typical acquisition sequence when results buffering is turned off.



NOTE

The state of the Any Fail, All Pass, Part Detect, Output 0 - 7 lines depends on the results of the inspection. The states shown are just examples.

Results Buffer Enable = false will disable Results Avail Ack and Results Buffer Overrun. Results Avail will stay true after the first set of results. Use Inspection Complete Toggle to gate the results from the IO buffers into PLC memory.

Missed Ack will go true if Trigger Enable = true and Trigger Ready == false and a leading edge of Trigger occurs. Missed Ack will stay true until a successful acquisition occurs (Trigger Ready == true and leading edge of Trigger).

If Set Offline is true or the Checker application goes to Setup mode, then Online will go false and Offline Reason 1 will go true.

Trigger Enable should only go true if Online is true first. Trigger Enable = true if one of the signals required for Trigger Ready goes true.

Trigger Ready will be true if Online = true, Trigger Enable = true, Acquiring = false and <acquire buffer available>.

Trigger can go true any time Trigger Ready is true. Trigger should go false once Trigger Ack goes true. The leading edge of Trigger going true will cause Trigger Ready to go false and will cause Acquiring to go true.

The Input 0 line can be in any state during the sequence. The value at the leading edge of the Acquiring signal is the value that will be used.

Trigger Ack goes true when Trigger goes true. Trigger Ack goes false when Trigger goes false.

Acquiring will go true if Trigger Ready == true and after the leading edge of the Trigger signal. Acquiring means that the camera is taking the picture and moving the image to an acquire buffer. The trailing edge of Acquiring is the gating signal for the Acquisition ID and will cause Inspecting to go true.

The [Trigger] Acquisition ID must be stable valid when the Acquiring signal goes false. This number will be used to match up the acquired image with the inspection output by using the same number for the [Inspection] Results ID.

Inspecting will go true immediately after Acquiring goes false as it is the next process in the pipeline. The trailing edge of Inspecting is the gating signal for Results ID and all of the results data. Inspection Complete Toggle and Results Avail are also activated by the falling edge of Inspecting.

Results ID must be valid and stable prior to the trailing edge of Results Available. The Inspection Complete Toggle will transition and Results Available will go true on the trailing edge of Inspecting. The Results ID value must be the same as the [Trigger] Acquisition ID value for the matching camera image.

Any Fail must be valid and stable prior to the trailing edge of Results Available. The Inspection Complete Toggle will transition and Results Available go true on the trailing edge of Inspecting.

All Pass must be valid and stable prior to the trailing edge of Results Available. The Inspection Complete Toggle will transition and Results Available go true on the trailing edge of Inspecting.

Part Detect must be valid and stable prior to the trailing edge of Results Available. Inspection Complete Toggle will transition and Results Available will go true on the trailing edge of Inspecting.

Output 0 - 7 must be valid and stable prior to the trailing edge of Results Available. Inspection Complete Toggle will transition and Results Available will go true on the trailing edge of Inspecting.

Inspection Complete Toggle changes at the same time as the trailing edge of Inspecting. The Results ID is incremented every time Results Available becomes true (that is, when Inspection Complete Toggle changes status).

[Inspection] Results Avail immediately follows the trailing edge of Inspecting and must only go true once the results data is solidly stable valid.

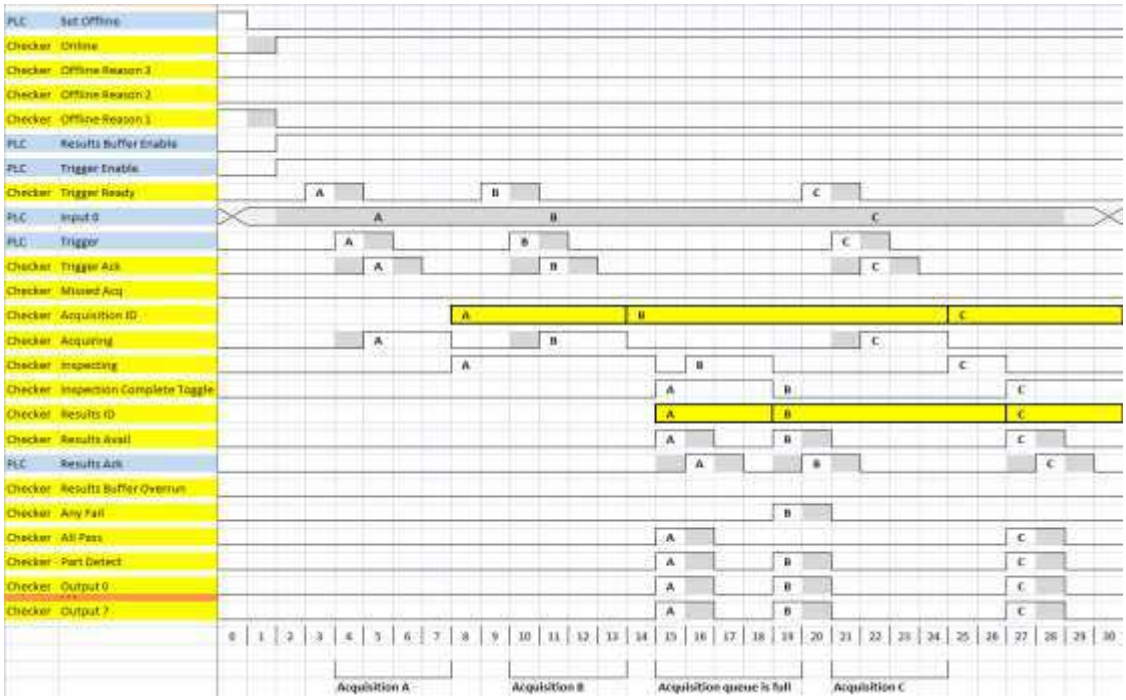
NOTE

For Results Buffer Enable = false, after the first result, the Results Available signal will always be true. Use Inspection Complete Toggle as the enable signal to copy new result data from the IO result buffers to PLC memory.

Results Ack is not used if Results Buffer Enable = false.

Results Buffer Overrun is not used if Results Buffer Enable = false.

Missed acquisition means that a trigger was issued at a point in time when the Checker was not able to act on it. This allows the PLC to skip over its normal processing logic and re-issue a new trigger. For an acquisition that is faster than the inspection sequence, see the following figure.



The states of the Any Fail, Any Pass, Part Detect, and Output 0 to 7 lines depend on the results of the inspection. The states shown are just examples.

Inspection / Result Sequence

When an image is acquired it is placed in a queue for processing. While the processing is running on the image, the Inspecting bit of the InspectionStatusRegister is set. When the inspection is complete, the Inspecting bit is cleared and the Inspection Completed bit is toggled.

Use cases for having separate Acquiring and Inspection indicators are, for example, to improve speed, or robotics. While Acquiring the robot must remain still. However, during Inspection the robot is allowed to move.

The BufferResultsEnable attribute determines how inspection results are handled by the Checker object. If the BufferResultsEnable attribute is set to false, then the inspection results are immediately placed into the InspectionResults attribute and Results Available is set to true.

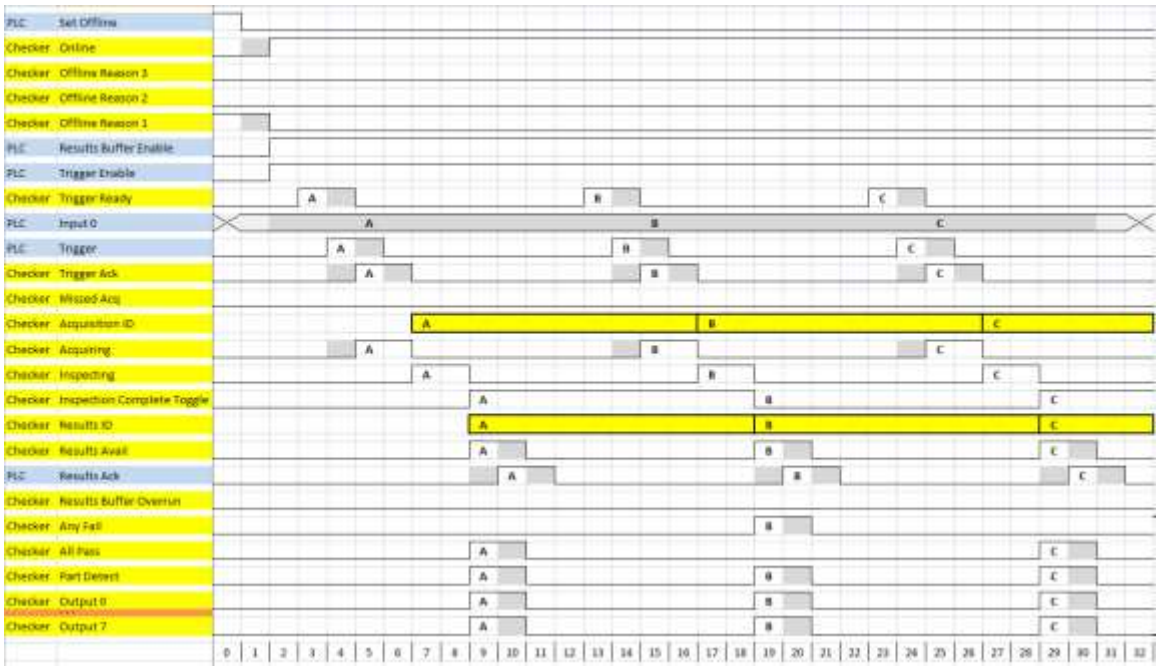
If the BufferResultsEnable attribute is set to true the new results are queued. When using internal triggering, the next trigger will overwrite the results. When using external triggering, the next external trigger will overwrite the result even if it is not used by the application.

Behavior of Inspection Bits

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
1	Inspecting	Set when inspecting an image.	Set when decoding an image.
2	Inspection Complete	Toggled on completion of an image inspection.	Toggled on completion of an image inspection.
3	Results Buffer Overflow	Remains set to zero.	Set when inspection results could not be queued because the client failed to acknowledge a previous result. Cleared when the inspection result is successfully queued. Overflow tells you something was lost. Depending on whether you have buffering enabled it means you lost an earlier inspection (no buffering) or lost the latest inspection (buffering with queue full).

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
4	Results Available	Becomes true after the first inspection and stays true.	Set when new results are placed into the results bits (Any Fail, All Pass, Part Detect and Output). Stays set until the results are acknowledged by setting Results Available Ack to true. **NOTE** that the Results Available / Results Available Ack handshake logic always applies.

See the following figure for a sequence when buffering is enabled.



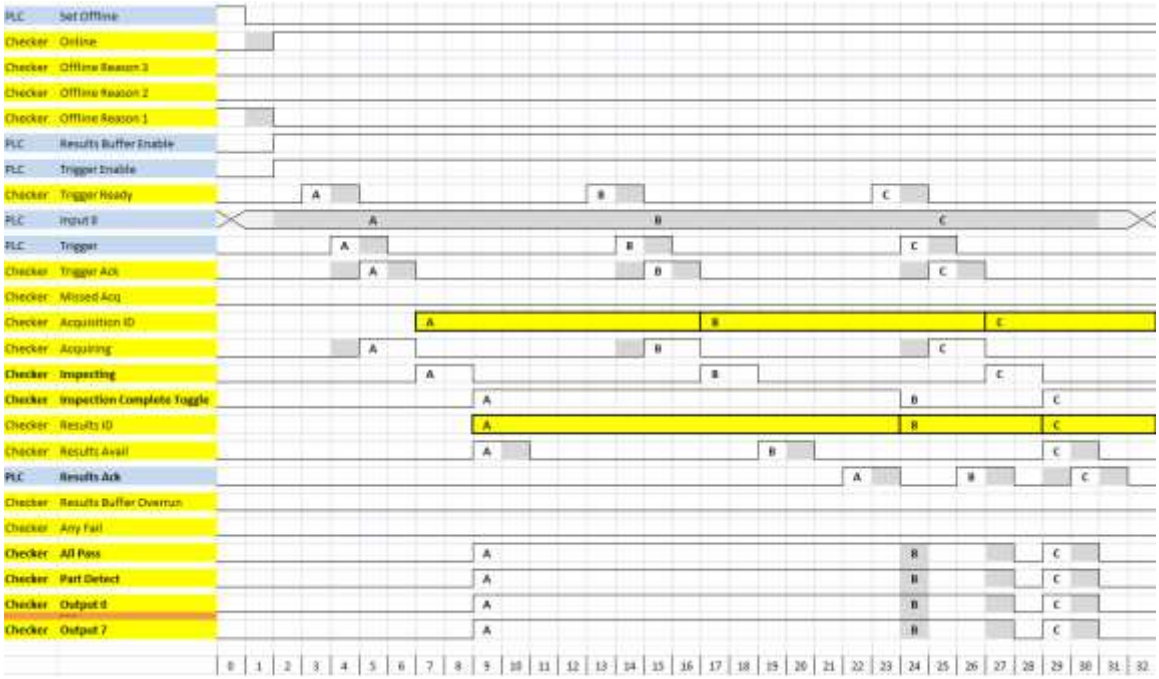
The states of the Any Fail, Any Pass, Part Detect, and Output 0 to 7 lines depend on the results of the inspection. The states shown are just examples.

Results Buffering

There is an option to enable a queue for inspection results. If enabled, this allows a finite number of inspection results data to queue up until the client (PLC) has time to read them. This is useful to smooth out data flow if different parts of the system (including the external PLC) slowdown for short periods of time.

In general, if inspections are occurring faster than results can be sent out, the primary difference between buffering or not buffering determines which results get discarded. If buffering is not enabled the most recent results are kept and the earlier result (which was not read by the PLC fast enough) is lost. Essentially the most recent result will simply overwrite the earlier result. If buffering is enabled (and the queue becomes full) the most recent results are discarded until room becomes available in the results queue.

As shown in the following figure, first the results corresponding to the A Acq go to the results buffer and to the output lines. This is followed by the results of B going to the result buffer, but still the results of A remain in the output lines. After the PLC read the results of A, it disappears from both the buffer and the output lines.



Assembly Object

Assembly objects use implicit messaging. In the abstract they are just blocks of data which are transmitted as the raw payload of implicit messaging packets. These implicit messaging packets are produced (transmitted) repeatedly at a predefined chosen rate (10ms, 200ms, and so on).

Assemblies are generally combinations of selected attributes (data items) from different CIP objects within a device. The device vendor defines assemblies according to their needs. They combine data together in useful groupings according to the requirements of the application.

The designation of Input & Output assembly can be confusing. Checker is an I/O adapter class device. The convention for adapters is that Input Assemblies produce (transmit) data for another device (i.e. Checker → PLC) and Output Assemblies consume (receive) data from another device (i.e. PLC → Checker). Essentially Checker acts as an I/O module for another device.

Checkers have a single input assembly and single output assembly. These assemblies combine selected attributes (data) of the Checker object into groupings that minimize network bandwidth and still allow for efficient control and processing. The data in these assemblies can also be accessed individually from the Checker object. However, using the assembly objects is much more efficient. This is the reason that they are the primary means of runtime communication between a Checker and a PLC.

Input Assembly

The Input Assembly Instance provides status information, process state, and inspection results. The following table shows the Input Assembly (data sent from the Checker), where Reserved[1] means reserved for future use. If the Observer bit is set to 1, the EIP controller cannot change the status of the Checker and can only observe its state.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
11	0	Online	Offline Reason • 0 – Online • 1 – Setup • 2-7 – Reserved			Missed Acq	Acquiring	Trigger Ack	Trigger Ready
	1	General Fault	Job Load Failed	Job Load Complete	Observer	Results Available	Results Buffer Overrun	Inspection Complete Toggle	Inspecting
	2	Reserved[1]	Reserved[1]	Reserved[1]	Retrain Failed	Retrain Completed	Any Fail	All Pass	Part Detect
	3	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
	4	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9	Output 8
	5	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17	Output 16
	6	Acquisition ID (16-bit integer)							
	7								
	8	Inspection Result ID (16-bit integer)							
	9								

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	10	Reserved							
	11	Reserved							
	12	Sensor 1 Result Data 0							
	…	…							
	27	Sensor 1 Result Data 15							
	28	Sensor 2 Result Data 0							
	…	…							
	43	Sensor 2 Result Data 15							
	44	Sensor 3 Result Data 0							
	…	…							
	411	Sensor 25 Result Data 15							
	412	Sensor 26 Result Data 0							
	…	…							
	427	Sensor 26 Result Data 15							

Output Assembly

The Output assembly instance contains control signals, software event signals, and any user data required for the trigger and inspection. The following table shows the Output Assembly Instance (data sent to the Checker), where Reserved[1] means reserved for future use. Setting the Lights Off bit to 1 turns off the external illumination regardless of the job setting. Setting the SM Volatile bit to 0 selects the static results data that does not change for a job. Setting it to 1 selects the volatile data that varies from part to part. This data is returned in the Sensor Meter table values.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
21	0	Set Offline	Lights Off	Retrain	Job Change	Results Available Ack	Buffer Results Enable	Trigger	Trigger Enable
	1	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1	Input 0
	2	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9	Input 8
	3	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17	Input 16
	4	Job Number (8-bit integer)							
	5	Reserved[1]							SM Volatile
	6	Pad byte							
	7	Pad byte							

Static and Volatile Data Summary

The static and volatile data returned in the in the Sensor Meter table is the same for all protocols (EtherNet/IP, PROFINET, and Generic FFP).

The following table summarizes the static data elements returned in the Sensor Meter table values.

Notation:

- X (with yellow background) means static data.
- N/A means the data type is not available for the specific sensor type.

Static table – 12 bytes							
	Setpoint 1	Setpoint 2	Sensor Angle	Sensor Angle Range 1	Sensor Angle Range 2	Sensor Width	Sensor Height
bits	8	8	16	16	16	16	16
Part Finder	X	N/A	N/A	N/A	N/A	X	X
Pattern Presence	X	N/A	N/A	X	X	X	X
Brightness Presence	X	N/A	N/A	N/A	N/A	X	X
Contrast Presence	X	N/A	N/A	N/A	N/A	X	X
Edge Presence	X	X	X	X	X	X	X
Contour Presence	X	N/A	N/A	X	X	X	X
Color Presence	X	X	N/A	N/A	N/A	X	X
Measurement Width	X	X	X	N/A	N/A	X	X
Measurement Height	X	X	X	N/A	N/A	X	X
Measurement Diameter	X	X	N/A	N/A	N/A	X	X
Pattern Position	N/A	N/A	N/A	X	X	X	X
Area (Blob) Position	N/A	N/A	N/A	N/A	N/A	X	X
Edge Position	N/A	N/A	X	X	X	X	X

The following table summarizes the volatile data elements returned in the Sensor Meter table values.

Notation:

- X (with white background) means volatile data.
- X (with yellow background) means static data.
- N/A means the data type is not available for the specific sensor type.

Volatile table – Overall 16 bytes and 30 sensors allowed									
	Analog Out	Pass / Found	Sensor Center X	Sensor Center Y	Feature Center X	Feature Center Y	Feature Width	Feature Height	Feature Angle
bits	8	8	16	16	16	16	16	16	16
Part Finder	X	X	X	X	X	X	X	X	N/A
Pattern Presence	X	X	X	X	N/A	N/A	N/A	N/A	N/A
Brightness Presence	X	X	X	X	N/A	N/A	N/A	N/A	N/A
Contrast Presence	X	X	X	X	N/A	N/A	N/A	N/A	N/A
Edge Presence	X	X	X	X	N/A	N/A	N/A	N/A	N/A
Contour Presence	X	X	X	X	X	X	X	X	X
Color Presence	X	X	X	X	N/A	N/A	N/A	N/A	N/A
Measurement Width	X	X	X	X	X	X	X	X	N/A
Measurement Height	X	X	X	X	X	X	X	X	N/A
Measurement Diameter	X	X	X	X	X	X	X	X	N/A
Pattern Position	N/A	X	X	X	X	X	X	X	X
Area (Blob) Position	N/A	X	X	X	X	X	X	X	N/A
Edge Position	N/A	X	X	X	X	X	N/A	X	X

NOTE

The following values are valid for the Pass / Found field:

Binary Value (decimal equivalent)	Meaning
00 (0)	Not Found
01 (1)	Found but not passed
11 (3)	Found and passed

Checker Implementation

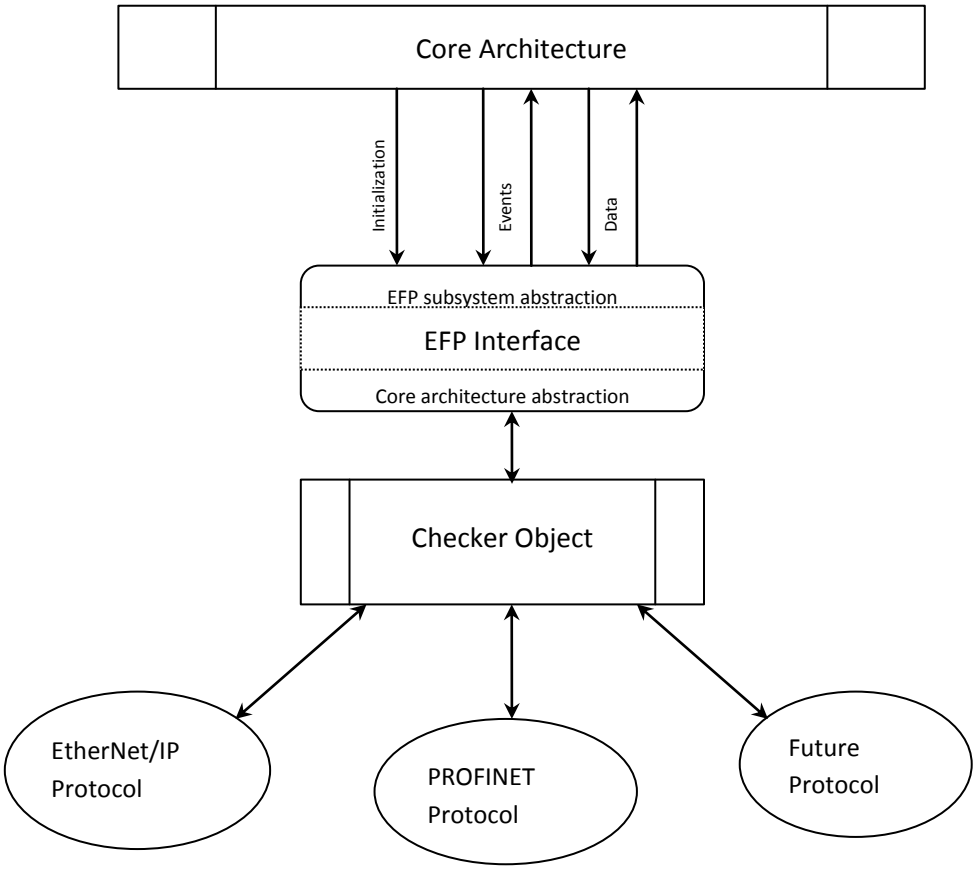
Ethernet Factory Protocol Subsystem

The Ethernet Factory Protocol (EFP) subsystem is a grouping of implementations of all "factory floor" Ethernet industrial protocols that have been implemented for the Checker products. Currently this includes EtherNet/IP and PROFINET.

This subsystem has been abstracted into a common interface. The interface provides a common method of initialization, event trafficking, and data flow.

The interface is abstracted in both directions. It abstracts the details of the EFP sub-system from the core Checker architecture. It also abstracts the details of the Checker core architecture from the protocols in the EFP subsystem.

In general only one protocol may be in operation at a given time. Central to the EFP sub-system is the Checker Object. The Checker Object models all data and functionality available in the Checker (at least everything the end user is allowed to view or touch via the Ethernet factory protocols). The Checker Object ensures that the Checker device has common data and behavior regardless of which factory protocol is in use.



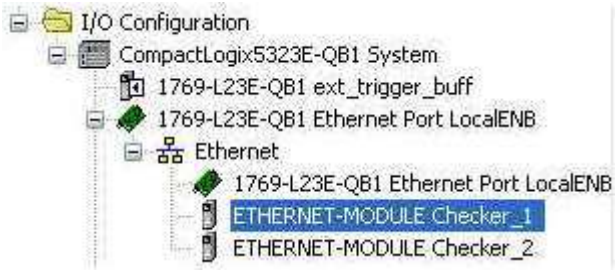
Establishing a Generic Implicit Messaging Connection

If you have not installed AOP or the EDS file, you can still use your Checker as described in this section. To setup an EtherNet/IP implicit messaging connection between a Checker and a ControlLogix controller, the Checker sensor must first be added to the ControlLogix I/O Configuration tree. This can be accomplished with the Rockwell provided generic profile.

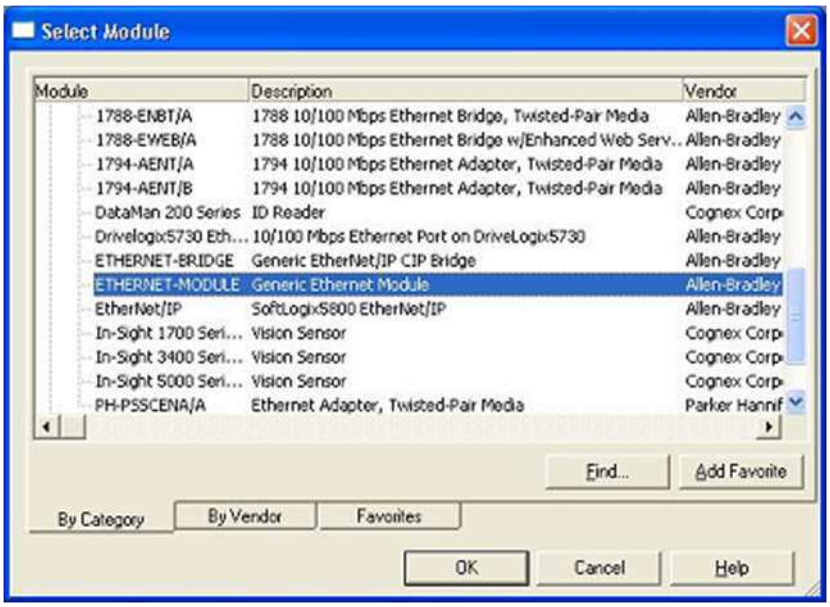
To establish a generic implicit messaging connection with a ControlLogix PLC:

1. Open RSLogix5000 and load your project (or select **File**->**New…** to create a new one).

2. From the I/O Configuration node, select the *Ethernet* node under the project Ethernet Module, right-click on the icon and select New Module from the menu:

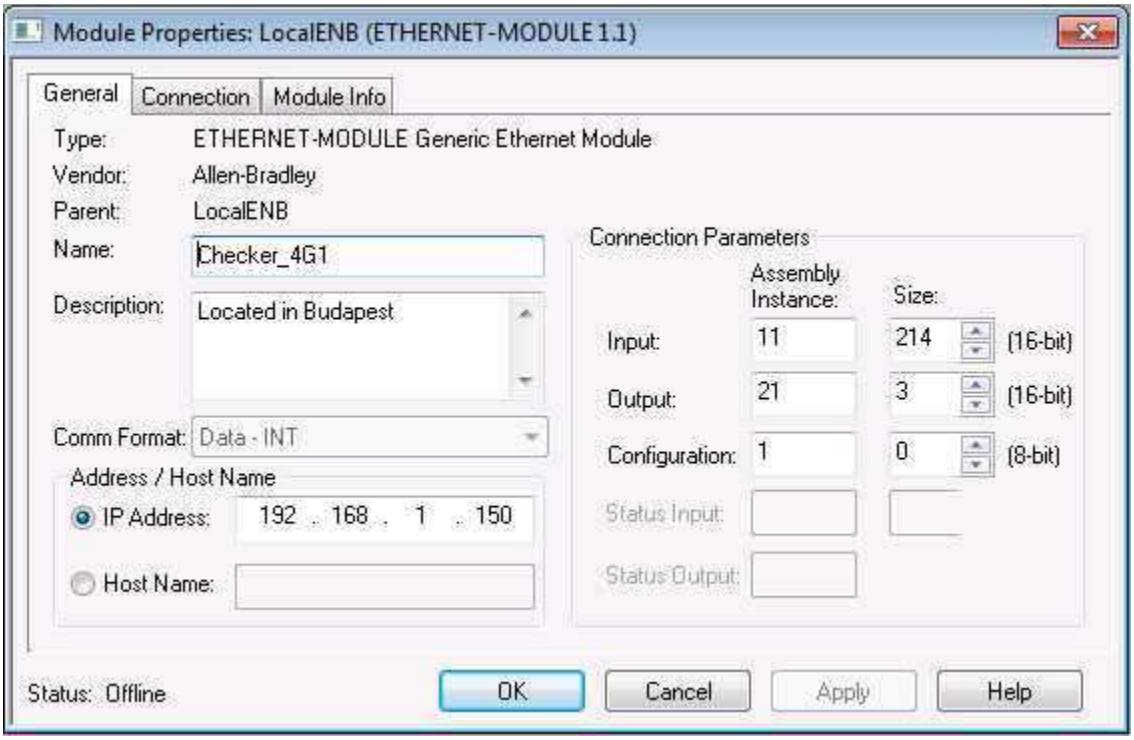


3. From the Select Module dialog, choose the Allen-Bradley Generic Ethernet Module.

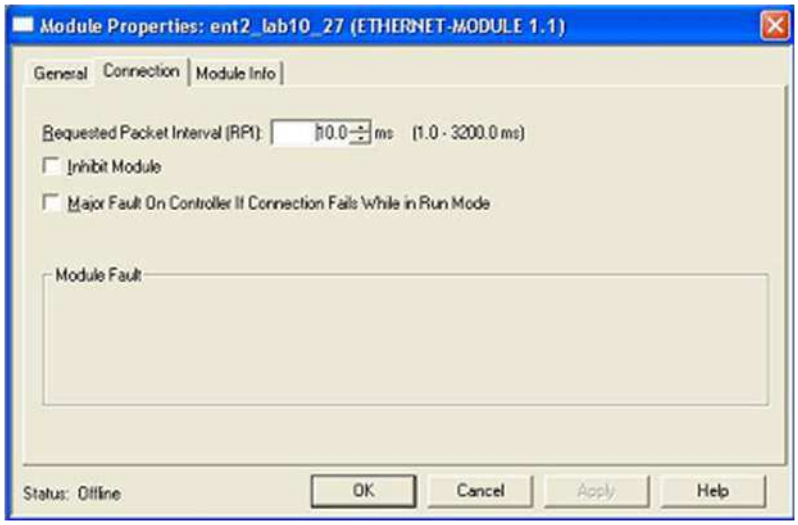


4. After the selection is made, the configuration dialog for the Generic Ethernet Module will be displayed. Configure the following:

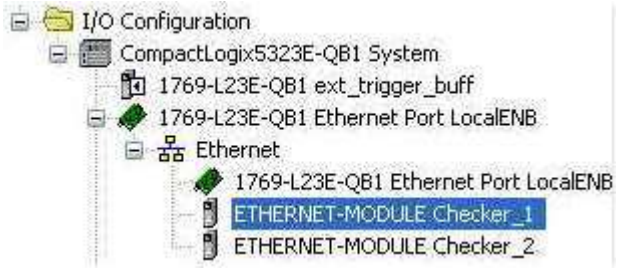
- Give the module a name.
- Enter your Checker's IP address.
- Set the Comm Format to "Data – INT". This tells the module to treat the data as an array of 16-bit integers.
- Input Assembly: Set instance 11. Set the size to the amount of Input Assembly data you want the PLC to receive. The size should be 5 words (80bit).
- Output Assembly: Set instance 21. Set the size to 3 integers. This size is sufficient to send all required "Control" data/command to the Checker.
- Configuration Assembly: Set instance 1. Set size to zero (no used).



5. The final step is configuring the connection rate. The rate at which data is transmitted/received is defined as the Requested Packet Interval (RPI). The RPI defines how frequently the data is transmitted/received over the connection. To optimize network performance this rate should be set no lower than absolutely required by a given application. In no case should it be set to lower than ½ the median scan rate of the PLC ladder program. Setting it lower wastes bandwidth and does not improve processing performance.



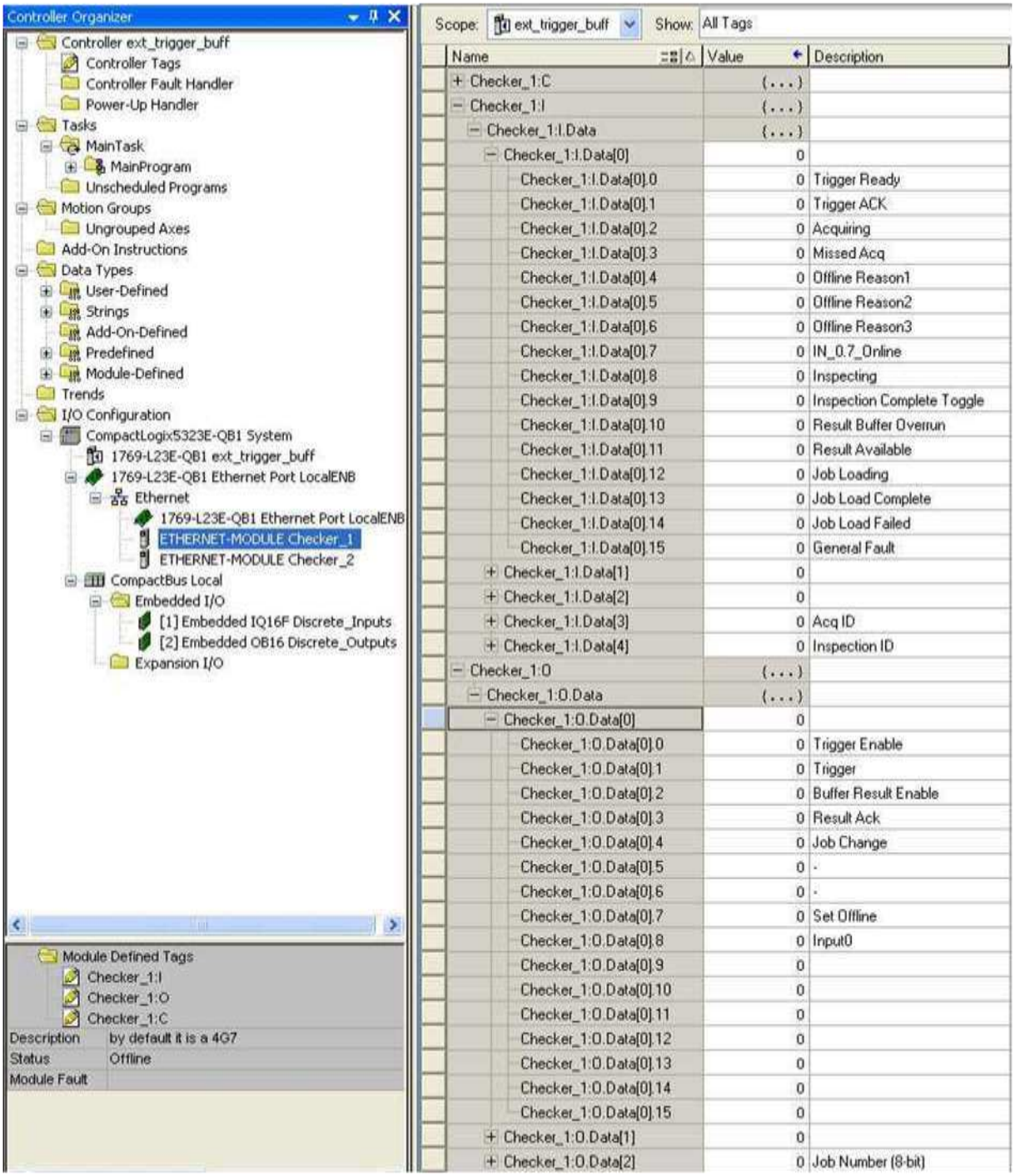
6. After adding the generic module to ControlLogix, the I/O tree should appear as follows.



7. When the Generic Module is added to the I/O tree RSLogix 5000 creates tags that map to the Checker sensor Input and Output Data (that is, the Input and Output Assembly Objects in the Checker sensor). These tags can be found under the “Controller Tags” node of the project tree.

NOTE

The base name of these tags is the name you gave to the Generic Module that you added to the I/O Configuration earlier.



The tags are organized in three groups: Config “Checker_1:C”, Input “Checker_1:I”, and Output “Checker_1:O”. You can ignore the Config tags (not used). The Input tags represent all the data being received (from the Checker). The Ouput tags represent all the data being sent (to the Checker).

These tags are the data table representation of the Checker Assembly Object contents. The PLC ladder is written to access these tag values. By monitoring or changing these tag values the PLC ladder is actually monitoring and changing the Checker Assembly Object contents.

NOTE

There is a time delay between the Checker and these PLC tag values (based on the configured RPI). All PLC ladder must be written to take that time delay into account.

Accessing Generic Implicit Messaging Connection Data

Unlike the Checker Add-On-Profile, the Generic profile does not automatically generate named tags representing the individual data items within an Assembly Object. Instead it simply generates an array of data according to the size of the connection you defined.

To access individual data items within an Assembly Object you must manually select the correct tag offset and data subtype (if necessary) within the tag array that the Generic profile provided. This can be awkward and error prone since it requires you to manually reference the vendor documentation which defines the Assembly Objects.

NOTE

The start of the Input tags “Checker_1:I.Data[0]” maps directly to the start of the Checker Input Assembly. Likewise, the start of the Output tags “Checker_1:O.Data[0]” maps directly to the start of the Checker Output Assembly.

Examples

Output Assembly Trigger Enable: Bit 0 of word 0 of the Output Assembly. From the Output tag array for the Checker select bit 0 of word 0.

− Checker_1:0	{...}	
− Checker_1:0.Data	{...}	
− Checker_1:0.Data[0]	0	
Checker_1:0.Data[0].0	1	Trigger Enable
Checker_1:0.Data[0].1	0	Trigger
Checker_1:0.Data[0].2	0	Buffer Result Enable
Checker_1:0.Data[0].3	0	Result Ack
Checker_1:0.Data[0].4	0	Job Change
Checker_1:0.Data[0].5	0	-
Checker_1:0.Data[0].6	0	-
Checker_1:0.Data[0].7	0	Set Offline
Checker_1:0.Data[0].8	0	Input0
Checker_1:0.Data[0].9	0	
Checker_1:0.Data[0].10	0	
Checker_1:0.Data[0].11	0	
Checker_1:0.Data[0].12	0	
Checker_1:0.Data[0].13	0	
Checker_1:0.Data[0].14	0	
Checker_1:0.Data[0].15	0	
+ Checker_1:0.Data[1]	0	
+ Checker_1:0.Data[2]	0	Job Number (8-bit)

Device Access Codes

The following table contains the access codes of the Checker sensor.

	EtherNet/IP Vendor ID	EtherNet/IP Device Type	EtherNet/IP Product Code
Checker 4G1	0x2a6	0x306	0x601
Checker 4G7	0x2a6	0x306	0x607

PROFINET

PROFINET is an application-level protocol used in industrial automation applications. This protocol uses standard Ethernet hardware and software to exchange I/O data, alarms, and diagnostics.

Checker supports PROFINET I/O. This is one of the 2 “views” contained in the PROFINET communication standard. PROFINET I/O performs cyclic data transfers to exchange data with Programmable Logic Controllers (PLCs) over Ethernet. The second “view” in the standard, PROFINET CBA (Component Based Automation), is not supported.

Enabling PROFINET involves the following main steps:

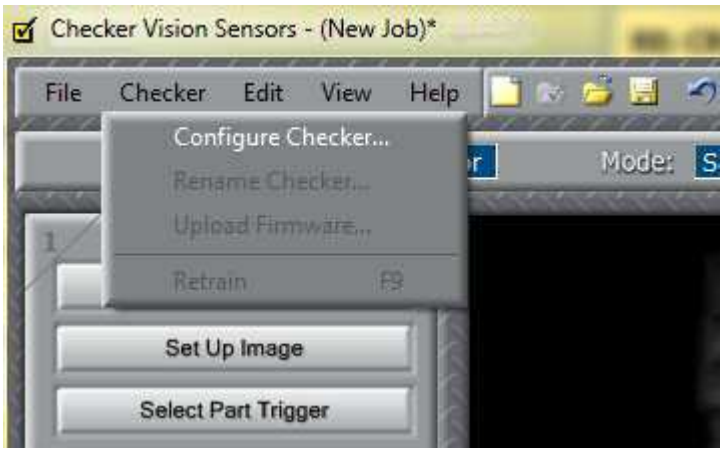
- Make sure you have the Siemens Step 7 programming software (SIMATIC) installed.
- Set up the Siemens Software tool so that it recognizes your Checker device.

  Install the Generic Station Description (GSD) file.
- Enable PROFINET in the Checker GUI

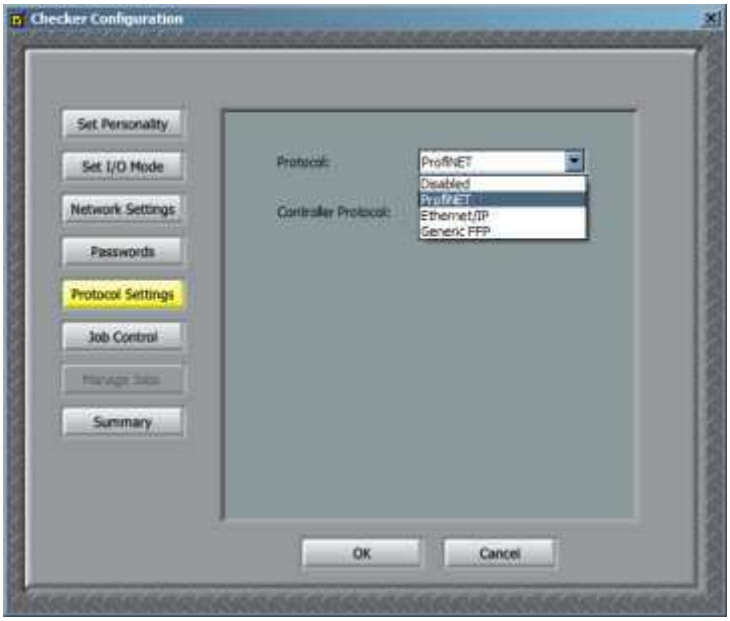
Enabling PROFINET in the Checker GUI

By default, Checker has the PROFINET protocol disabled. The protocol can be enabled in the Checker GUI. Perform the following steps:

1. In the upper menu toolbar, click Checker.
2. Select Configure Checker.



3. In the Checker Configuration window that pops up, click Protocol Settings and select PROFINET.



4. Click OK.

For the changes to take effect, the sensor automatically reboots.

Setting up PROFINET

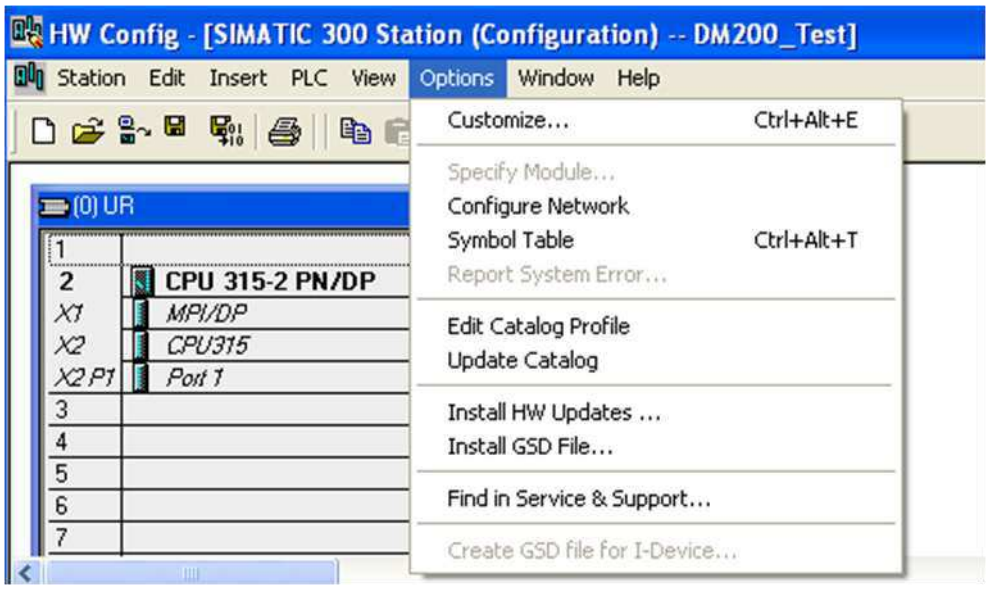
Perform the following steps to set up PROFINET:

1. Verify that SIMATIC is on your machine.

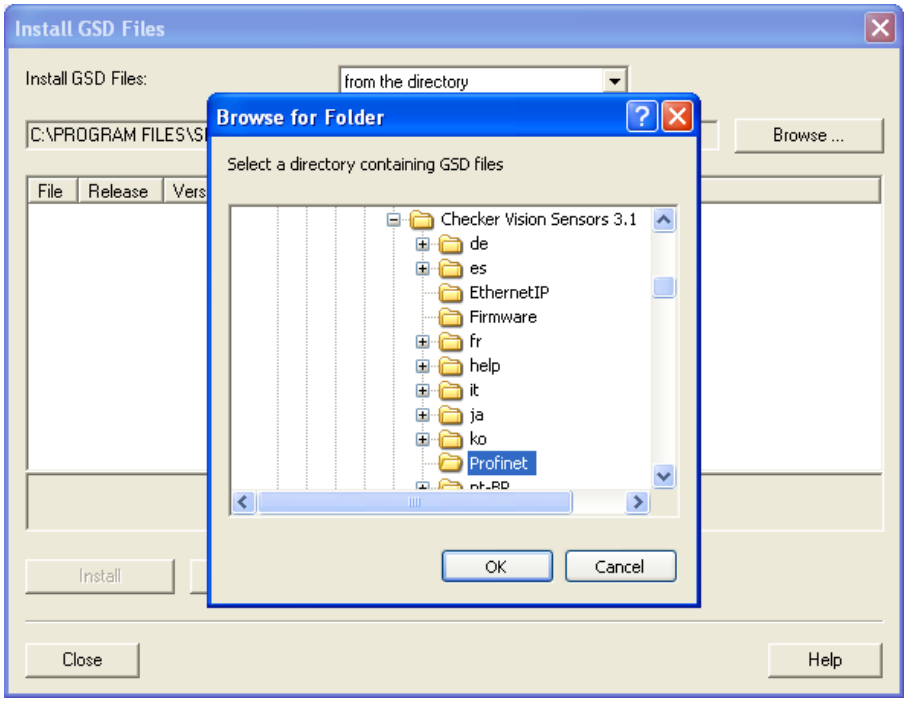
2. From the Windows Start menu, launch the SIMATIC Manager.

3. If you already have a project, select **Cancel** to skip past the New Project wizard. Otherwise, let the wizard guide you through creating a new project.

4. Once the Manager has opened the project, double-click the **Hardware** icon to open the **HW Config** dialog screen. From the main menu, select **Options→Install GSD File…**.



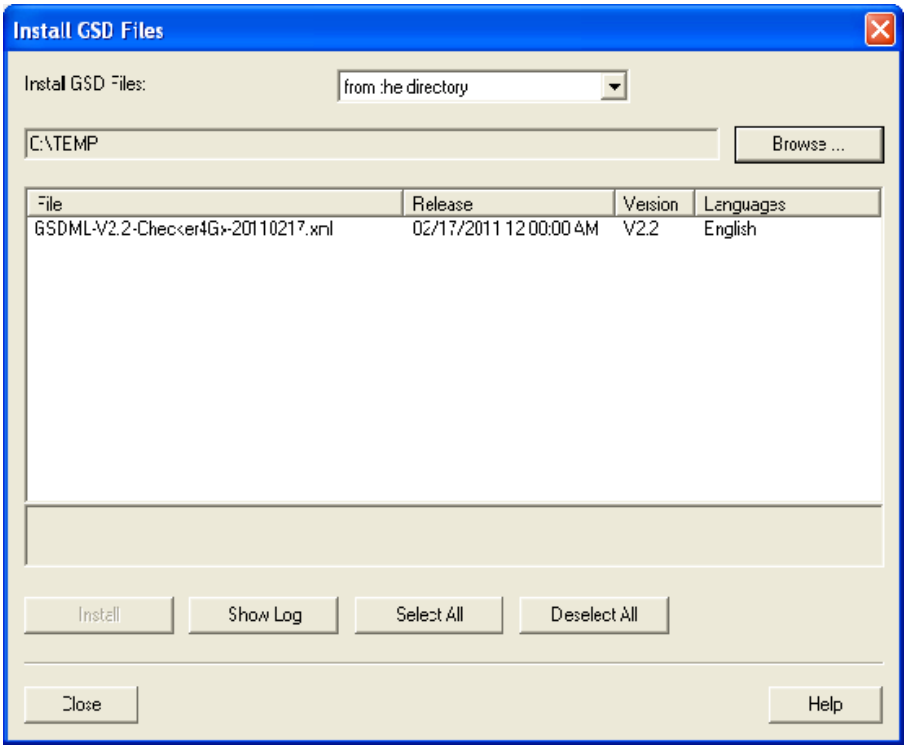
5. Browse to the location where the GSD file was installed (or the location where you saved the GSD file if it was downloaded from the web).



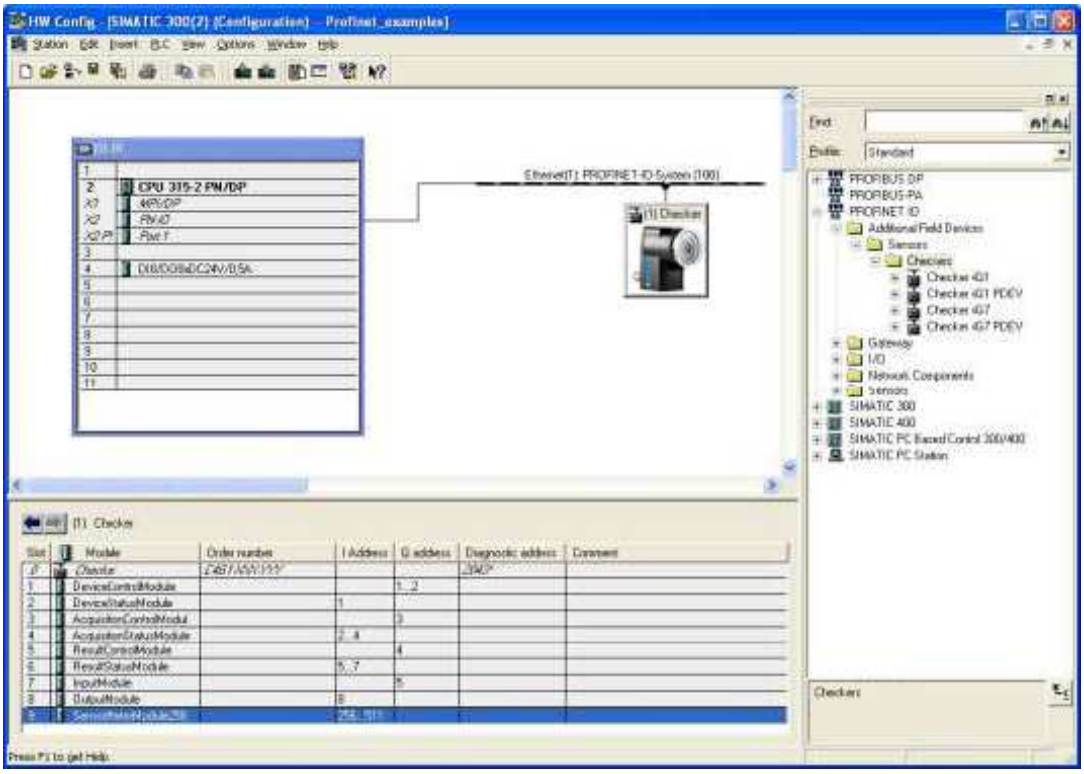
6. Select the GSD file you wish to install and follow the displayed instructions to complete the installation.

NOTE

There may be more than one GSD file in the list. If you are unsure which to install, choose the one with the most recent date.

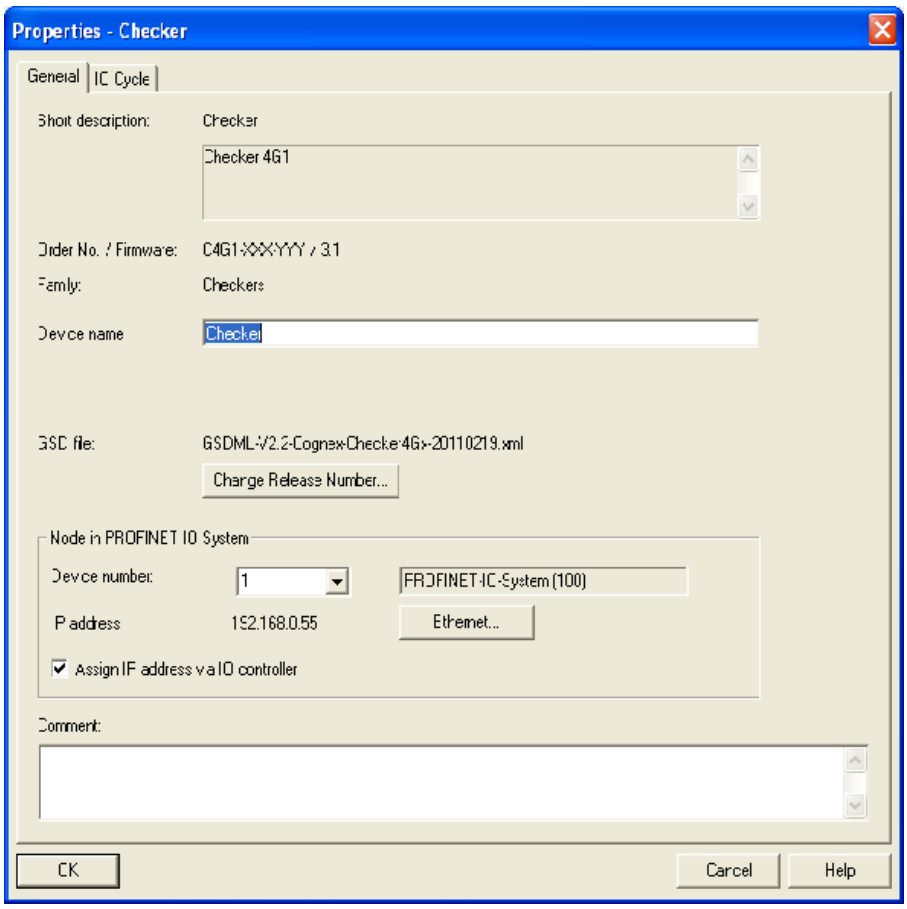


7. Add your Checker sensor to your project. This makes the Checker available in the Hardware Catalog. Launch the SIMATIC Hardware Config tool.
8. In the main menu, select View → Catalog.
9. The catalog is displayed. Expand the PROFINET IO tree to the “di-soric Checker” node.
10. With the left mouse button, drag the Checker sensor over and drop it on the PROFINET IO network symbol in the left pane.



The HW Config tool automatically maps the Checker I/O modules into the memory space.

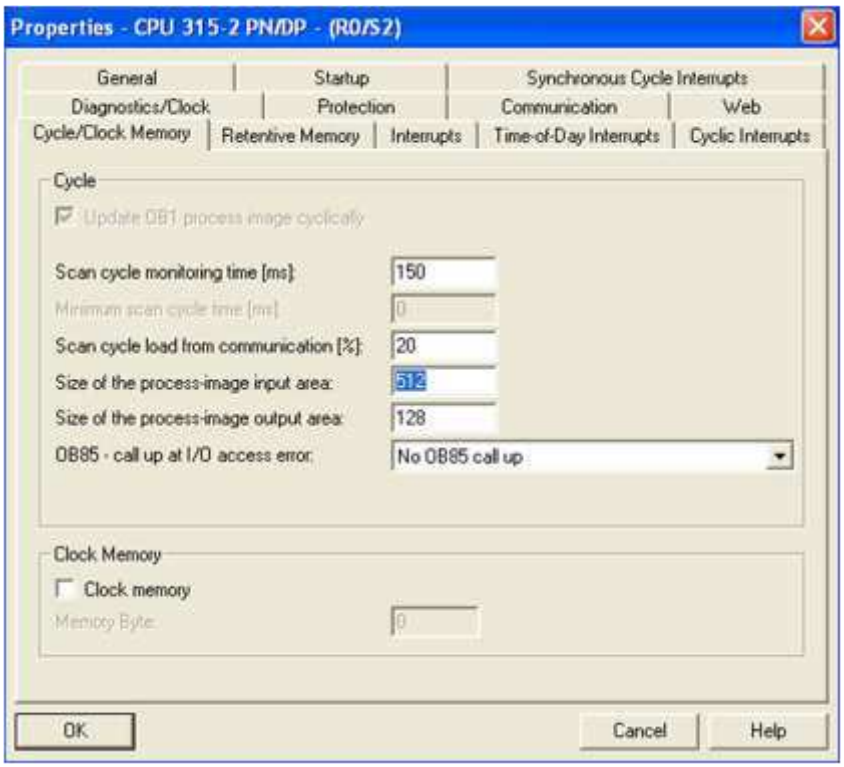
11. Right-click on the Checker icon and select Object Properties…
12. Give the sensor a name. This must match the name of your actual Checker sensor. The name must be unique and conform to DNS naming conventions. Refer to the SIMATIC Software help for details.
13. If your Checker sensor is configured to use its own static IP, uncheck the Assign IP address via IO controller box. Otherwise if you wish the PLC to assign an IP address, select the Ethernet button and configure the appropriate address.



14. In the IO Cycle tab, select the appropriate cyclic update rate for your application.

15. By default, the SIMATIC software maps the User Data & Result Data Modules to offset 256. This is outside of the default process image area size of 128. That is, by default, data in these modules are inaccessible by some SFCs such as BLKMOV. As a solution, either remap the modules to lower offsets within the process image area or expand the process image area to include these modules.

If you choose to expand the process image area, make the size large enough for the module size plus the default 256 offset.



NOTE

Expanding the process image can have a performance impact on the PLC scan cycle time. If your scan time is critical, use the minimal acceptable module sizes and manually remap them down lower in the process image.

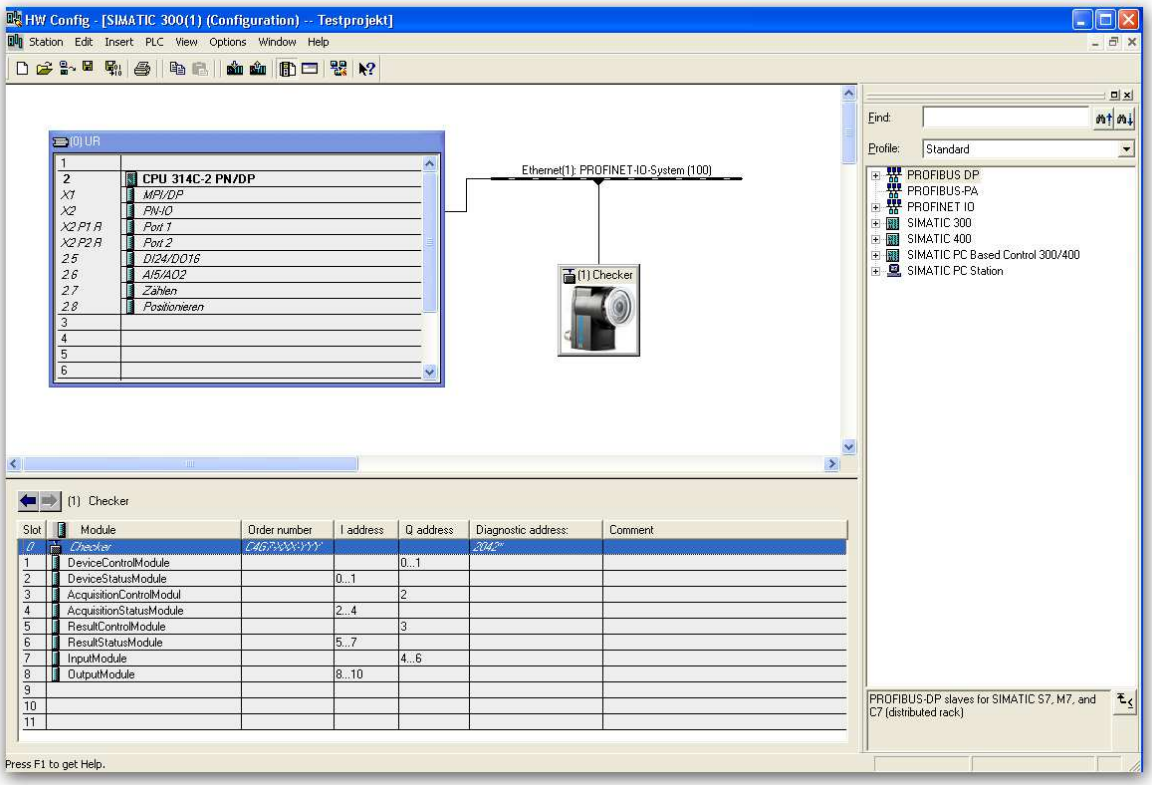
Modules

The PROFINET implementation on Checker consists of the following I/O modules:

1. Device Control Module
2. Device Status Module
3. Acquisition Control Module
4. Acquisition Status Module
5. Result Control Module
6. Result Status Module
7. Input

8. Output

9. Sensor Meter Module



Device Control Module
Controls the Checker device. This module consists of data that is sent from the PLC to the Checker sensor.

Slot number: 1

Total Module size: 2 byte

Bit	Name	Description
0	Set Offline	Setting this bit puts the Checker into Offline mode. This means that the Checker suspendsall activities and does not react to any control commands.
1	Job Change	This bit is used for changing jobs. Before (or at the same time) this bit is changed to 1 by the PLC, make sure that the Job Number isfilled in.
2	Retrain	This bit is used to retrain the sensor.

3	Lights Off	Setting the Lights Off bit to 1 turns off the external illumination regardless of the job setting.
4	Volatile	Setting the Volatile bit to 0 selects the static results data that does not change for a job. Setting it to 1 selects the volatile data that varies from part to part.
8 – 15	Job Number	See Job Change.

Device Status Module
Indicates the Checker device status. This module consists of data sent from the Checker device to the PLC.

Slot number: 2

Total Module size: 2 byte

Bit	Name	Description
0	Online	If the Checker is online, its value is 1 and the Checker reacts to commands. Otherwise its value is 0. See also Set Offline.
1	Offline Reason 1	If this is 1, the sensor is Offline. If this is 0, the sensor is Online. Setup Mode.
2	Offline Reason 2	Not used.
3	Offline Reason 3	Not used.
4	General Fault	It is always 0.
5	Job Load Complete	If the Job Change is successful, this changes to 1.
6	Job Load Failed	If the Job Change is unsuccessful, this changes to 1.
7	Retrain Completed	If this is 1, the retrain operation for the sensor completed successfully.
8	Retrain Failed	If this is 1, the retrain operation for the sensor failed.
9	Observer	If this is 1, the PROFINET controller cannot change the status of the Checker and can only observe its state.

Acquisition Control Module
Controls image acquisition. This module consists of data sent from the PLC to the Checker device.

Slot number: 3

Total Module size: 1 byte

Bit	Name	Description
0	Trigger	Setting this bit triggers an acquisition when the following conditions are met: • Trigger Enable is set • No acquisition is currently in progress • The device is ready to trigger
1	Trigger Enable	Setting this bit enables triggering via PROFINET. Clearing this bit disables triggering.
2 – 7	Reserved	Reserved for future use

Acquisition Status Module

Indicates the current acquisition status. This module consists of data sent from the Checker device to the PLC.

Slot number: 4

Total Module size: 3 bytes

Bit	Name	Description
0	Trigger Ready	Indicates when the device is ready to accept a new trigger. Bit is true when Trigger Enable has been set and the device is ready to accept a new trigger.
1	Trigger Ack	Indicates that the Checker has received a new Trigger and the trigger process has started. This bit will remain true as long as the Trigger bit remains true (that is, it is interlocked with the Trigger bit).
2	Acquiring	Indicates that the Checker is currently acquiring an image.
3	Missed Ack	Indicates that the Checker was unable to successfully trigger an acquisition. Bit is cleared when the next successful acquisition occurs.
4 – 7	Reserved	Reserved for future use

Bit	Name	Description
8–23	Acquisition ID	ID value of the next trigger to be issued (16-bit integer). Used to match issued triggers with corresponding result data received later. This same value will be returned in Result ID of the result data.

Results Control Module

Controls the processing of result data. This module consists of data that is sent from the PLC to the Checker sensor.

Slot number: 5

Total Module size: 1 byte

Bit	Name	Description
0	Results Buffer Enable	Enables queuing of Result Data. If enabled, the current result data will remain until acknowledged (even if new results arrive). New results are queued. The next set of results are pulled from the queue (made available in the Result Data module) each time the current results are acknowledged. The Checker will respond to the acknowledgement by clearing the Results Available bit. Once the Results Ack bit is cleared, the next set of read results will be posted and Results Available will be set true. If results buffering is not enabled, newly received read results will simply overwrite the content of the Result Data module.
1	Results Ack	This bit is used to acknowledge that the PLC has successfully read the latest result data. When set true the Result Available bit will be cleared. It is only used if Results Buffering is enabled.
2 – 7	Reserved	Reserved for future use

Results Status Module

Indicates the acquisition and result status. This module consists of data sent from the Checker device to the PLC.

Slot number: 6

Total Module size: 3 bytes

Bit	Name	Description
0	Part Detect	Its value is 1 if the Part Finding Sensor was successful. If unsuccessful, its value is 0. It is valid if Results Available is 1.
1	Inspecting	Its value is 1 when the Checker is inspecting. Otherwise its value is 0.
2	Inspection Complete Toggle	It changes value when the inspection is finished.
3	Result Buffer Overrun	This bit is only valid in case of Result Buffering (Buffer Result Enable = 1). Its value changes to 1 if there is no place in the buffer for the last inspection result.
4	Result Available	Its value changes to 1 when the inspection is finished. See also Inspection Complete Toggle.
5	Any Fail	Its value is 1 if the inspection of at least one of the sensors defined in the current Job is unsuccessful. Its opposite is All Pass. If the inspections of all sensors are successful, its value is 0. This bit is valid if Result Available is 1.
6	All Pass	Its value is 1 if all the inspections of all the sensors defined in the current Job are successful. If any of them is not successful, its value is 0. Its opposite is Any Fail. This bit is valid if Result Available is 1.
7 - 22	Result ID Register	This bit is the pair of Acquisition ID.

Input Module

Slot number: 7

Total Module size: 3 bytes

Bit	Name	Description
0 – 23	Inputs 0 through 23	These bits are used as Inputs in the Ladder Logic of the Checker GUI.

Output Module

Slot number: 8

Total Module size: 3 bytes

Bit	Name	Description
0 – 23	Outputs 0 through 23	The function of these bits can be changed in the Checker GUI. They can be any of the following virtual outputs: • Part Detect • All Pass • All Fail • Coil • External Trigger • External Retrain Pass • External Retrain Fail • Job Change Pass • Job Change Fail These bits are only valid if the Result Available is 1.

Sensor Meter Module

Slot number: 9

Total Module size: 256 bytes

For the Sensor Meter information, see the Static and Volatile Data Summary section.

NOTE

The order of the sensor meter values available through PROFINET is determined by the order of the sensors you set in the Sensor Meter table.

Operation

Acquisition Sequence

Checker can be triggered to acquire images implicitly via the Assembly object.

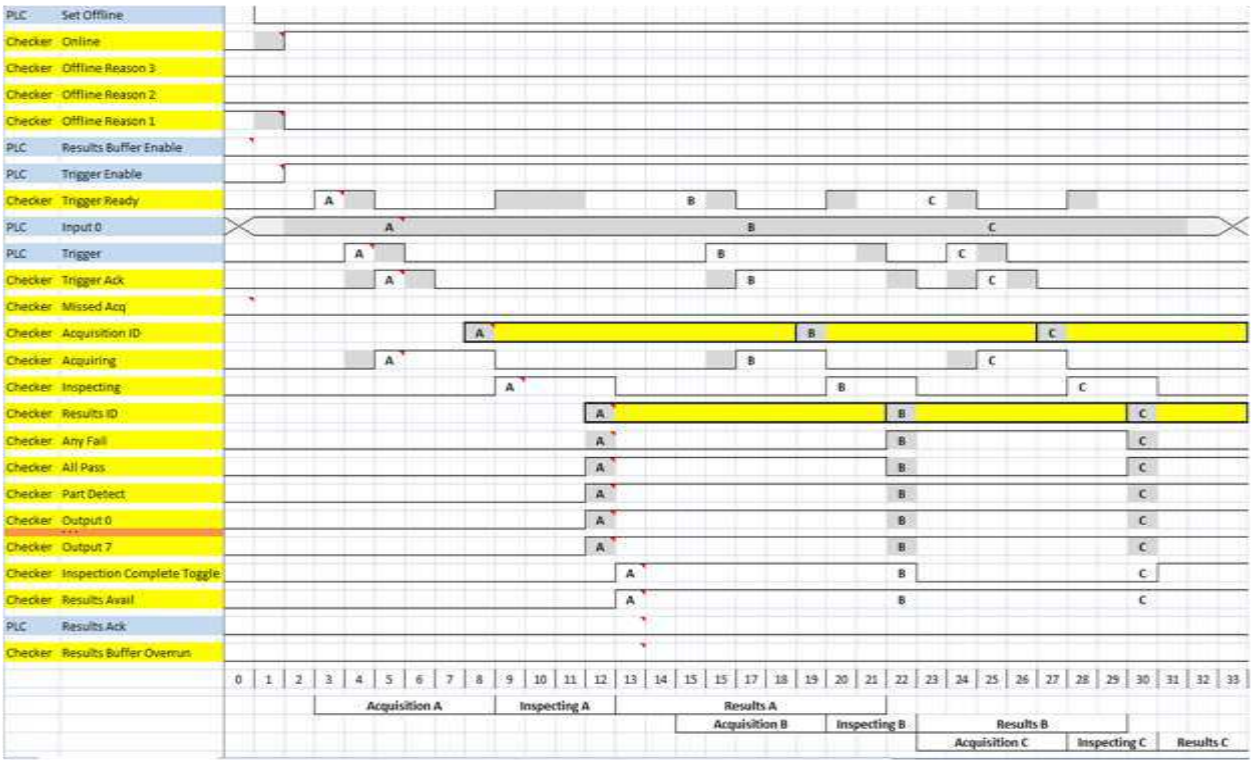
On startup the TriggerEnable attribute will be false. It must be set to true to enable triggering. When the device is ready to accept triggers, the Trigger Ready bit in the AcqStatusRegister will be set to true.

While the TriggerEnable attribute is true and Trigger Ready bit is true, each time the Checker object sees the Trigger attribute change from 0 to 1, it will initiate an image acquisition. When setting this via the assembly objects, the attribute should be held in the new state until that same state value is seen in the Trigger Ack bit of the AcqStatusRegister (this is a necessary handshake to guarantee that the change is seen by the Checker object).

During an acquisition, the Trigger Ready bit in the AcqStatusRegister will be cleared and the Acquiring bit will be set to true. When the acquisition is completed, the Acquiring bit will be cleared and the Trigger Ready bit will again be set true once the acquisition system is ready to begin a new image acquisition.

To force a reset of the trigger mechanism set the TriggerEnable attribute to false, until the AcqStatusRegister is 0. Then, TriggerEnable can be set to true to re-enable acquisition.

Triggering is only available when Checker is in External Trigger Mode. See the following figure for a typical acquisition sequence where results buffering is disabled.



NOTE

The state of the Any Fail, All Pass, Part Detect, Output 0 - 7 lines depends on the results of the inspection. The states shown are just examples.

Results Buffer Enable = false will disable Results Avail Ack and Results Buffer Overrun. Results Avail will stay true after the first set of results. Use Inspection Complete Toggle to gate the results from the IO buffers into PLC memory.

Missed Ack will go true if Trigger Enable = true and Trigger Ready == false and a leading edge of Trigger occurs. Missed Ack will stay true until a successful acquisition occurs (Trigger Ready == true and leading edge of Trigger).

If Set Offline is true or the Checker application goes to Setup mode, then Online will go false and Offline Reason 1 will go true.

Trigger Enable should only go true if Online is true first. Trigger Enable = true if one of the signals required for Trigger Ready goes true.

Trigger Ready will be true if Online = true, Trigger Enable = true, Acquiring = false and <acquire buffer available>.

Trigger can go true any time Trigger Ready is true. Trigger should go false once Trigger Ack goes true. The leading edge of Trigger going true will cause Trigger Ready to go false and will cause Acquiring to go true.

The Input 0 line can be in any state during the sequence. The value at the leading edge of the Acquiring signal is the value that will be used.

Trigger Ack goes true when Trigger goes true. Trigger Ack goes false when Trigger goes false.

Acquiring will go true if Trigger Ready == true and after the leading edge of the Trigger signal. Acquiring means that the camera is taking the picture and moving the image to an acquire buffer. The trailing edge of Acquiring is the gating signal for the Acquisition ID and will cause Inspecting to go true.

The [Trigger] Acquisition ID must be stable valid when the Acquiring signal goes false. This number will be used to match up the acquired image with the inspection output by using the same number for the [Inspection] Results ID.

Inspecting will go true immediately after Acquiring goes false as it is the next process in the pipeline. The trailing edge of Inspecting is the gating signal for Results ID and all of the results data. Inspection Complete Toggle and Results Avail are also activated by the falling edge of Inspecting.

Results ID must be valid and stable prior to the trailing edge of Results Available. The Inspection Complete Toggle will transition and Results Available will go true on the trailing edge of Inspecting. The Results ID value must be the same as the [Trigger] Acquisition ID value for the matching camera image.

Any Fail must be valid and stable prior to the trailing edge of Results Available. The Inspection Complete Toggle will transition and Results Available go true on the trailing edge of Inspecting.

All Pass must be valid and stable prior to the trailing edge of Results Available. The Inspection Complete Toggle will transition and Results Available go true on the trailing edge of Inspecting.

Part Detect must be valid and stable prior to the trailing edge of Results Available. Inspection Complete Toggle will transition and Results Available will go true on the trailing edge of Inspecting.

Output 0 - 7 must be valid and stable prior to the trailing edge of Results Available. Inspection Complete Toggle will transition and Results Available will go true on the trailing edge of Inspecting.

Inspection Complete Toggle changes at the same time as the trailing edge of Inspecting. The Results ID is incremented every time Results Available becomes true (that is, when Inspection Complete Toggle changes status).

[Inspection] Results Avail immediately follows the trailing edge of Inspecting and must only go true once the results data is solidly stable valid.

NOTE

For Results Buffer Enable = false, after the first result, the Results Available signal will always be true. Use Inspection Complete Toggle as the enable signal to copy new result data from the IO result buffers to PLC memory.

Results Ack is not used if Results Buffer Enable = false.

Results Buffer Overrun is not used if Results Buffer Enable = false.

Inspection/Result Sequence

After an image is acquired it is inspected. While being inspected, the Inspection bit of the Result Status Module is set. When the inspection is complete, the Inspection bit is cleared and the Inspection Complete bit is toggled.

The Results Buffer Enable bit determines how inspection results are handled by the sensor. If the Results Buffer Enable bit is set to false, then the inspection results are immediately placed into the Results Module and Results Available is set to true.

If the Results Buffer Enable bit is set to true the new results are queued. The earlier inspection results remain in the Results Module until they are acknowledged by the client by setting the Results Ack bit to true. After the Results Available bit is cleared, the client should set the Results Ack bit back to false to allow the next queued results to be placed in to the Results Module. This is a necessary handshake to ensure the results are received by the Checker client (PLC).

Behavior of InspectionStatusRegister

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
1	Inspecting	Set when inspecting an image.	Set when inspecting an image.
2	Inspection Complete	Toggled on completion of an image inspection.	Toggled on completion of an image inspection.

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
3	Results Buffer Overflow	Remains set to zero.	Set when inspection results could not be queued because the client failed to acknowledge a previous result. Cleared when the inspection result is successfully queued.
4	Results Available	Becomes true after the first inspection and then stays true.	Set when new results are placed in the Results Module. Stays set until the results are acknowledged by setting Results Ack to true.

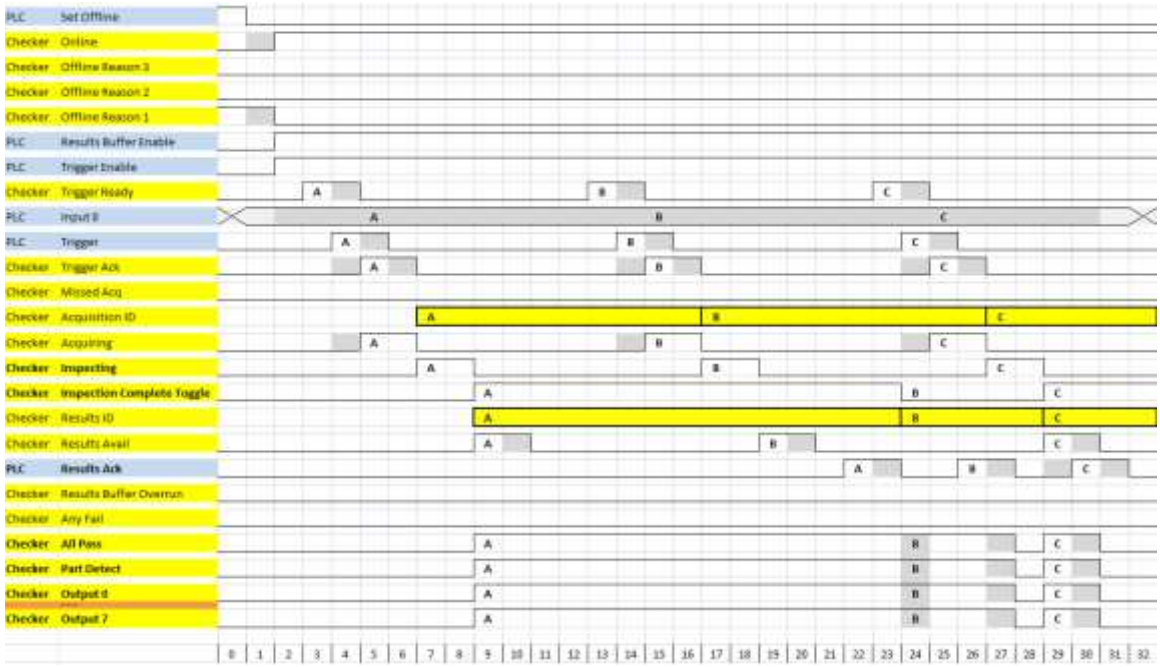
Results Buffering

There is an option to enable a queue for inspection results. If enabled this allows a finite number of inspection result data to queue up until the client (PLC) has time to read them. This is useful to smooth out data flow if the client (PLC) slows down for short periods of time or if there are surges of read activity.

Also, if result buffering is enabled the device will allow overlapped acquisition and inspection operations. Depending on the application this can be used to achieve faster over all trigger rates. See Acquisition Sequence description above for further detail.

In general, if reads are occurring faster than results can be sent out, the primary difference between buffering or not buffering determines which results get discarded. If buffering is not enabled the most recent results are kept and the earlier result (which was not read by the PLC fast enough) is lost. Essentially the more recent result will simply over write the earlier result. If buffering is enabled (and the queue becomes full) the most recent results are discarded until room becomes available in the results queue.

As shown in the following figure, first the results corresponding to the A Acq go to the results buffer and to the output lines. This is followed by the results of B going to the result buffer, but still the results of A remain in the output lines. After the PLC read the results of A, it disappears from both the buffer and the output lines.



Siemens Examples

This section gives some examples of using the Checker with a Siemens S7-300 PLC. It is assumed that you are familiar with the S7-300 and the SIMATIC programming software.

Symbol Table

Although not required, defining symbols for the Checker I/O module elements can be extremely helpful. It makes the code much easier to read and reduces mistakes. This sample table shows symbols defined for a typical instance of a Checker sensor. Note that Checker I/O modules may be at different addresses in your project. Make sure to adjust your symbol definitions based on the specific offsets of the I/O modules.

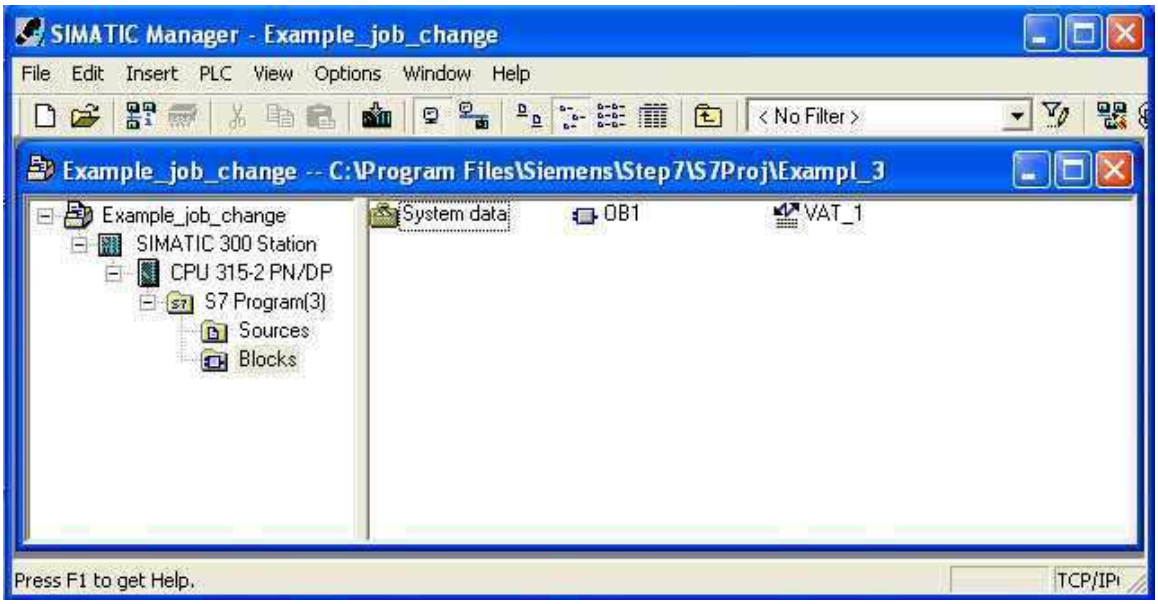
Name	Data Type	Address	Comment
TEMP		0.0	
OB1_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB_RESERVED_1	Byte	4.0	Reserved for system
OB_RESERVED_2	Byte	5.0	Reserved for system

Name	Data Type	Address	Comment
OB1_PREV_CYCLE	In	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And_Time	12.0	Date and time OB1 started
L_InitVars	Bool	20.0	
L_AfterFB4	Bool	20.1	
L_MultiTriggerPrev	Bool	20.2	
L_T0Prev	Bool	20.3	
L_Sink	Bool	20.4	

Variable Tables

Each example program contains at least one variable table you can use to monitor and change the values of variables present in the example programs during program execution.

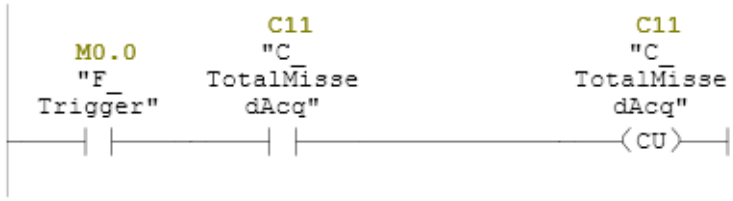
For example, to open the variable table for the Example_job_change program, open the blocks of the program and double-click on VAT_1 (the variable table file). See the following screenshot of the blocks.



A similar editable table will pop up:

VAT_1 -- Example_Trigger_non_buf\SIMATIC 300 Station\CPU 315-2 ...

	Address	Symbol	Display format	Status value	Modify value
1	//start automated triggering with the PLC (1 - Triggering, 0 - Not triggering)				
2	M 0.0	"F_Trigger"	BOOL		
3	//Handling of the triggers on the PLC side				
4	Q 3.0	"Q_Trigger"	BOOL		
5	Q 3.1	"Q_TriggerEnable"	BOOL		
6	//Handling other stuff on the PLC side				
7	Q 1.0	"Q_SetOffline"	BOOL		
8	Q 4.0	"Q_BufferEnable"	BOOL		
9	Q 4.1	"Q_ResultsAvailAck"	BOOL		
10	Q 5.0	"Q_Input0"	BOOL		
11	Q 1.1	"Q_JobChange"	BOOL		
12	QB 2	"Q_JobNumber"	HEX		
13	//Basic inputs about the checker state (Sent by the Checker)				
14	I 1.0	"I_Online"	BOOL		
15	I 1.1	"I_OfflineReason1"	BOOL		
16	I 1.2	"I_OfflineReason2"	BOOL		
17	I 1.3	"I_OfflineReason3"	BOOL		
18	I 1.4	"I_GeneralFault"	BOOL		
19	//Trigger related Inputs (Sent by the Checker)				
20	I 2.0	"I_TriggerReady"	BOOL		
21	I 2.1	"I_TriggerAck"	BOOL		
22	// Other Inputs from the Checker				
23	I 2.2	"I_Acquiring"	BOOL		
24	I 2.3	"I_MissedAquisition"	BOOL		
25	MW 3	"I_AcquisitionID"	HEX		
26	I 5.0	"I_PartDetect"	BOOL		
27	I 5.1	"I_Inspecting"	BOOL		
28	I 5.2	"I_InspCompleteToggle"	BOOL		
29	I 5.3	"I_ResultsBufferOverrun"	BOOL		
30	I 5.4	"I_ResultsAvailable"	BOOL		
31	I 5.5	"I_AnyFail"	BOOL		
32	I 5.6	"I_AllPass"	BOOL		
33	MW 6	"I_ResultID"	HEX		
34	//Virtual Outputs sent by the Checker (Inputs in the PLC side!)				
35	I 8.0	"I_Output0"	BOOL		
36	I 8.1	"I_Output1"	BOOL		
37	I 8.2	"I_Output2"	BOOL		
38	I 8.3	"I_Output3"	BOOL		
39	I 8.4	"I_Output4"	BOOL		
40	I 8.5	"I_Output5"	BOOL		



NOTE

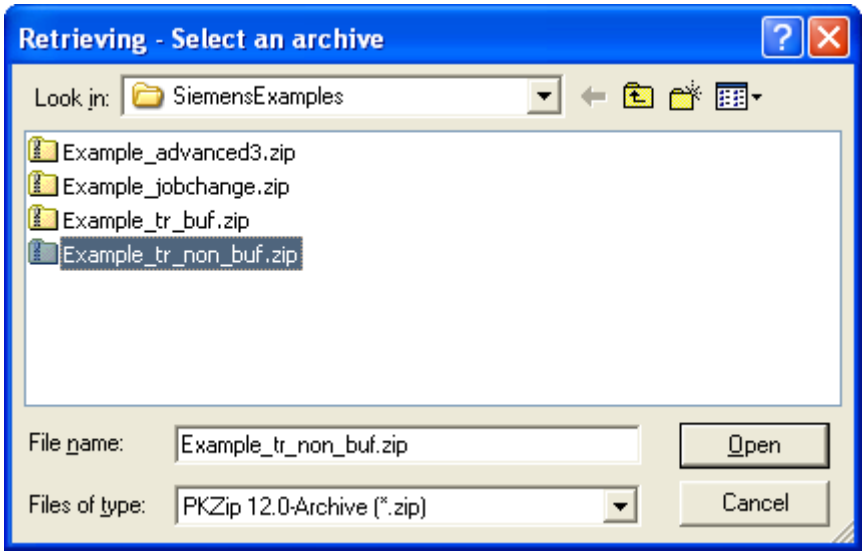
To run PROFINET programs, Checker must be in run mode with PROFINET enabled. When running the "Triggering without Buffering" and "Triggering with Buffering" program examples, make sure External Trigger Mode is turned on.

Triggering without Buffering

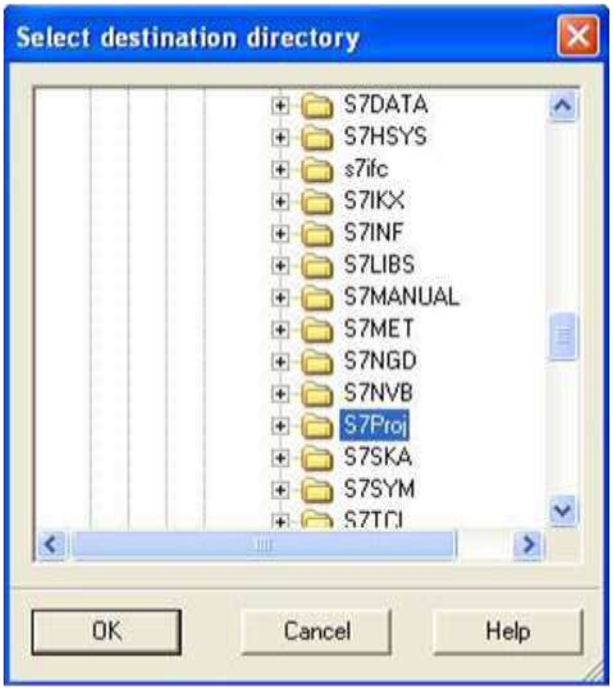
Perform the following steps to install the "Example_tr_non_buf" sample program:

1. Start the SIMATIC Manager software.
2. Close any open applications.
3. From the main menu, select **File → Retrieve…**
4. Browse to find the sample file on your PC.

5. Look for the Siemens folder and select the “Example_Tr_Non_Buf.zip” file.



6. Select a destination directory to save the project on your PC.



The Siemens software extracts the sample archive and makes it available.
In case you have problems opening the Siemens S7 PLC project because of language settings, perform the following steps:

1. Go to the C:\Program Files\Siemens\Step7\S7Proj folder (default) or the place where you keep your project.

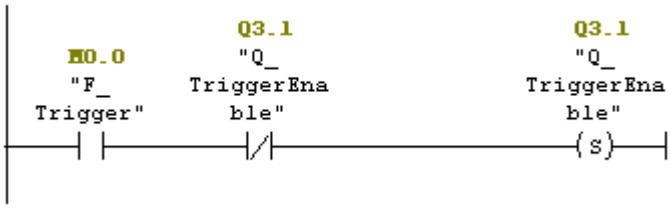
2. Open the Global folder.

3. Open the language file which contains your regional settings.
4. Copy the content.
5. Go to the project that you received, open the language file and overwrite its content.
6. See that now you can open the project.

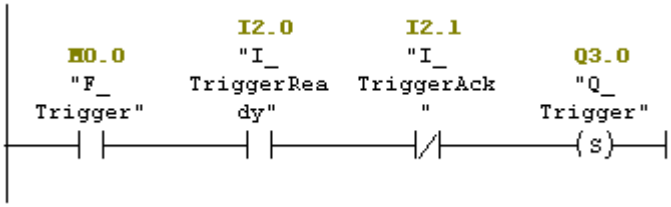
To run the sample program after opening it, download it to the PLC and then turn the value of F_Trigger to 1.

Reduced to the basics, the process of external triggering without using a buffer consists of the following:

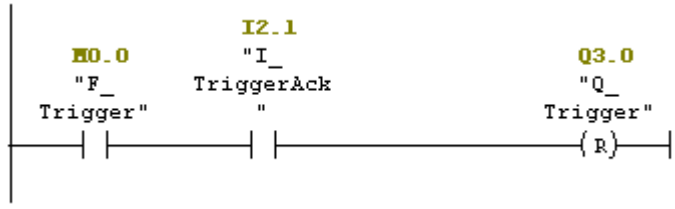
1. Enable the external trigger on the sensor:



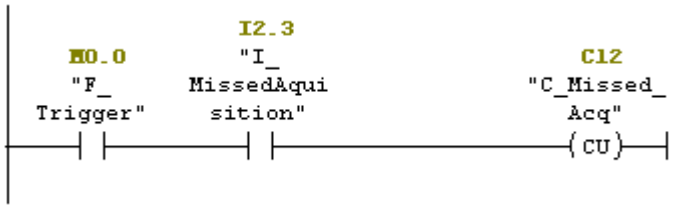
2. Set the trigger signal to start the acquisition.



3. As soon as the trigger signal is acknowledged, clear the trigger signal.



4. Check for any Missed Acquisition (you should not be detecting any).



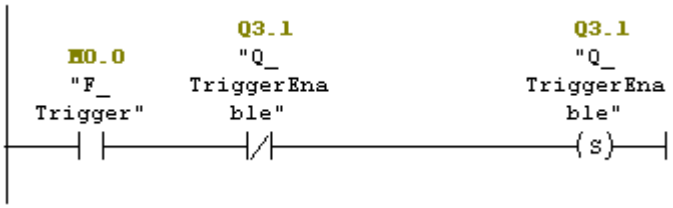
Triggering with Buffering

The main reason for buffering is that if the Checker triggers faster than the PLC can read out the result, some result will be stored in the Checker. In this case, the Acquisition ID and the Result ID might go out of sync.

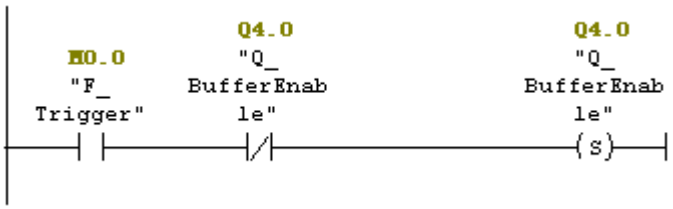
Install and run the “Example_Tr_Buf” sample program by selecting the “Example_tr_buf.zip” file the same way as described in section Triggering without Buffering.

Reduced to the basics, the process of external triggering using a buffer consists of the following:

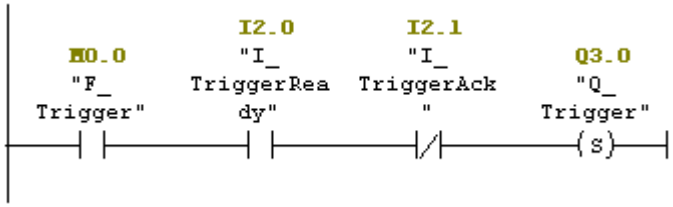
1. Enable the external trigger on the sensor:



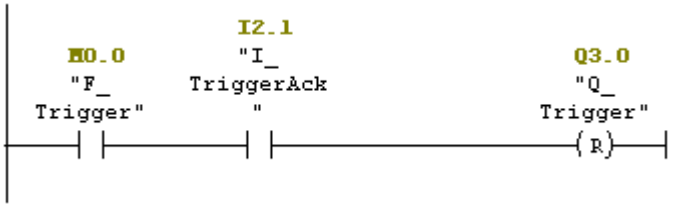
2. Turn on buffering.



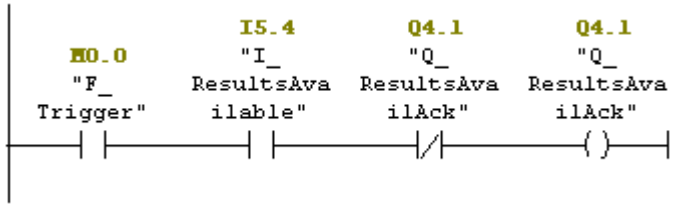
3. Set the trigger signal to start the acquisition.



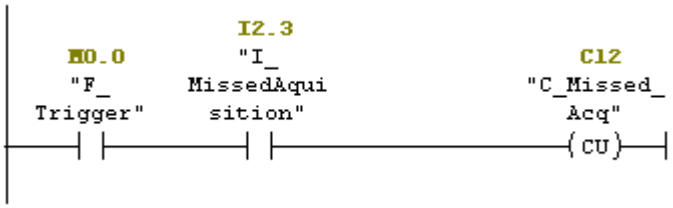
4. As soon as the trigger signal is acknowledged, clear the trigger signal.



5. If Results Available (sent by the Checker) is high, it means that there are some results not read out by the PLC. Set ResultAvailableAck to read.
   If ResultAvailable is low, set resultAvailableAck back to low so that the PLC becomes ready to read out the next result once it becomes available.



6. Check for any Missed Acquisition (you should not be detecting any).



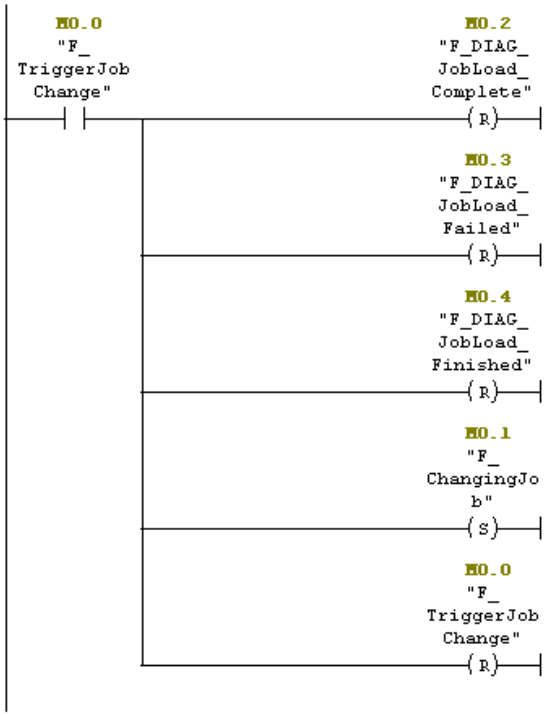
Example Job Change

This program attempts to load a job to the Checker from the Job Control table. (You can find it in the Checker Configuration menu in the PC Application.)

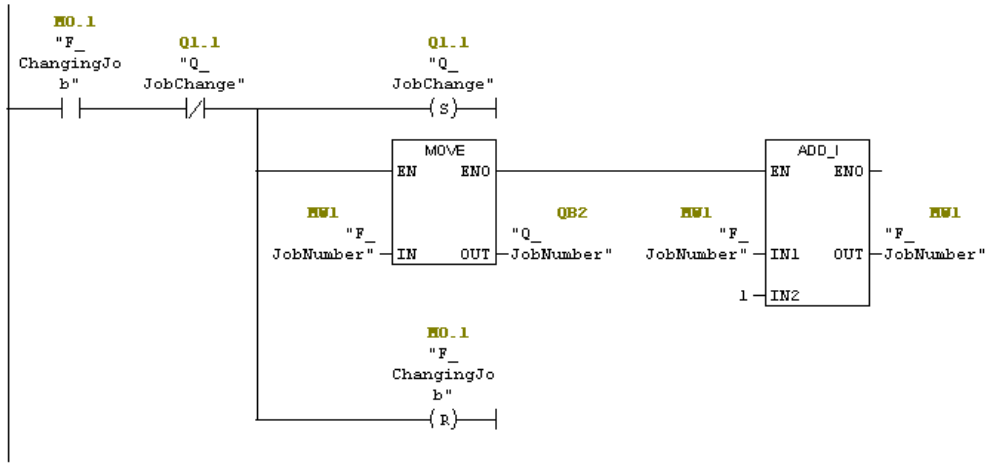
Install and run the “Example_job_change” sample program by selecting the “Example_Jobchange.zip” file the same way as described in section Triggering without Buffering.

Reduced to the basics, the process of changing a job consists of the following:

1. Start JobChange with triggering M0.0 manually. JobNumber goes up from 0.

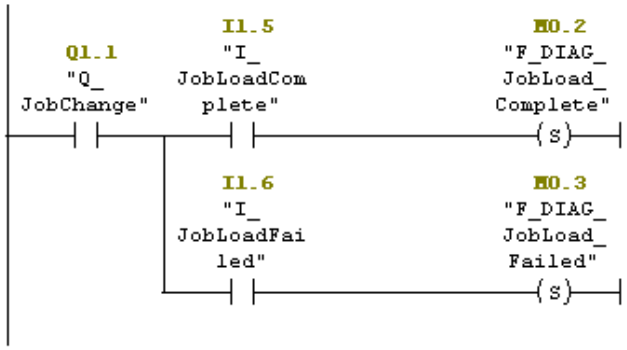


2. Load the current F_JobNumber to the Checker, then increase this value by one for the next job change process.

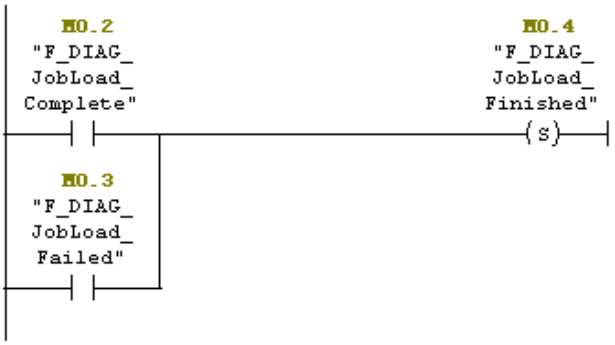


3. Safety network. If Q_JobChange remained high but there is no active job change, set it back to 0.

4. After the PLC recieved information from the Checker about the job change, set the correct DIAG bits. The reason behind this is that the Checker and the PLC set back the corresponding bits so fast that you would not be able to verify the result with your eyes. DIAG bits keep their values until the next job change starts.



5. After the PLC recieved information from the Checker about the job change, set the correct DIAG bits. If JobLoad_Complete or JobLoad_Failed is high, JobChange is finished too.



6. After the job change finished, set back Q1.1 to 0, and the PLC will be ready for the next job change.

A Complex Example

A complex example is available containing various functionalities (such as triggering without buffering, triggering with buffering, or job change) as blocks.

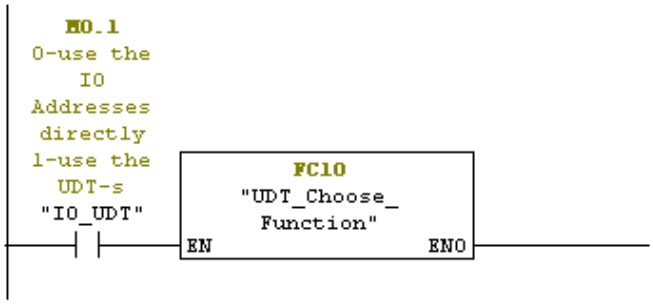
It is assumed that you are familiar with the SIMATIC programming software.

Install and run the “Profinet_examples” sample program by selecting the “Example_Complex.zip” file the same way as described in section Triggering without Buffering.

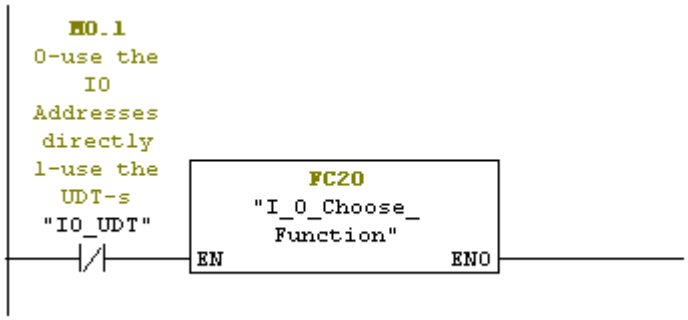
OB1 is the main part of the Checker example program.

In this example, you can choose between two types of programs:

1. UDT based programs: These programs use the Checker data block (Checker DB) representation (DB1 - Checker4Gx_1) for getting and setting the correct I/O values. In this way, the programs themselves will be almost independent of further GSDML changes or HW Config changes. You need to maintain only the FC1 (UDT_Get_Input) and the FC2 (UDT_Set_Output) functions and nothing else.



2. I/O-Address-based programs: These programs use the I/O Address areas directly. As a result, there is no need to use a separate DB on the PLC side. However, because I/O Address areas are accessed directly, these programs are much more sensitive to GSDML and Hardware changes. This is because the initial memory addresses of the Checker shift if a new device is configured before the Checker in the HW Configuration (and the whole program would have to be rewritten); also, the memory layout might change in a new GSDML. If the Checker I/O Address goes to another memory part, you need to actualize all Checker-related gates.



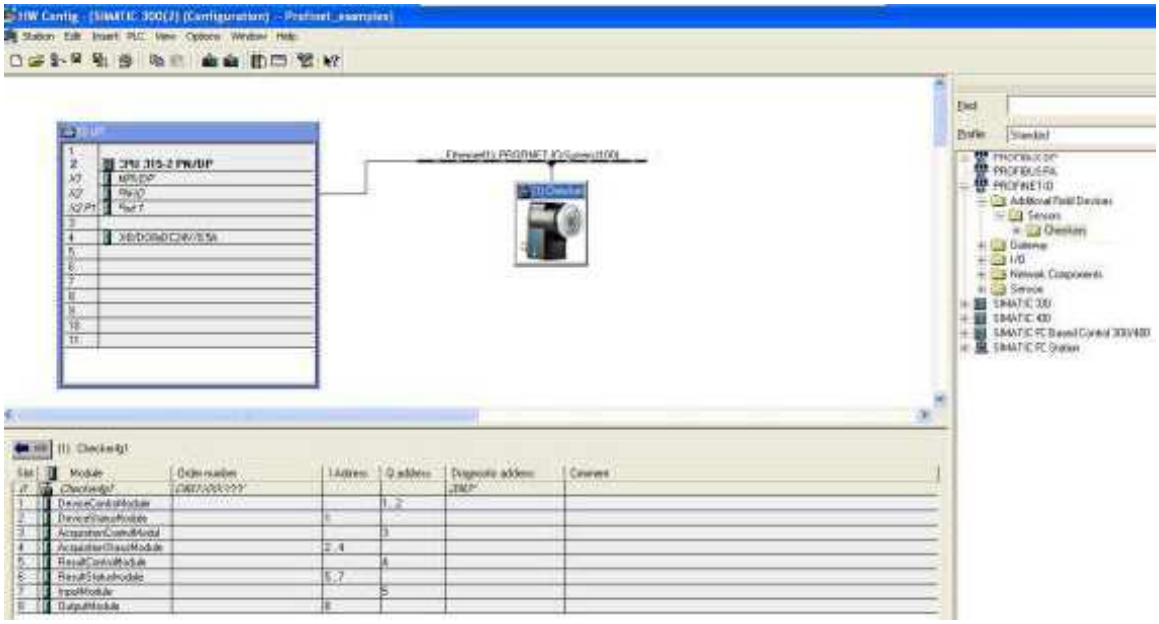
The following table summarizes the functionalities of the blocks present in the example program:

Name			Functionality/Purpose
Block Level			
1	2	3	
OB1			The main part of the Checker example program.
	FC10		For calling the various functions for UDT-based jobs.
		FC1	For reading inputs from PROFINET Input area.
		FC11	For job change handling for UDT-based jobs.
		FC12	For sending triggers to the Checker as fast as possible (non-buffered, UDT based).
		FC13	For sending triggers to the Checker as fast as possible (buffered, UDT based).
		FC2	For reading out the values from the DB and copying them to Checker Outputs.
	FC20		For calling the various functions for I/O-Address-based jobs.
		FC21	For job change handling for I/O-Address-based jobs.
		FC22	For sending triggers to the Checker as fast as possible (non-buffered, I/O Address based).
		FC23	For sending triggers to the Checker as fast as possible (buffered, I/O Address based).
OB86			"Loss Of Rack Fault" If this OB exists on the PLC, the PLC does not stop after a PN device becomes missing. The error LED is still blinking, but the program is running normally.
OB100			"Complete Restart" Run only once after PLC restart (warm restart is enough).
OB122			"Module Access Error" Same as OB86, but it handles the general periphery read error.
DB1			Checker data block.
UDT1			Checker data type table.
UDT2			Checker data type table.
UDT3			Checker data type table.
UDT4			Checker data type table.
Ch1_I_O _monitor			Checker variable table.
UDT_I_O _Table			Checker variable table.

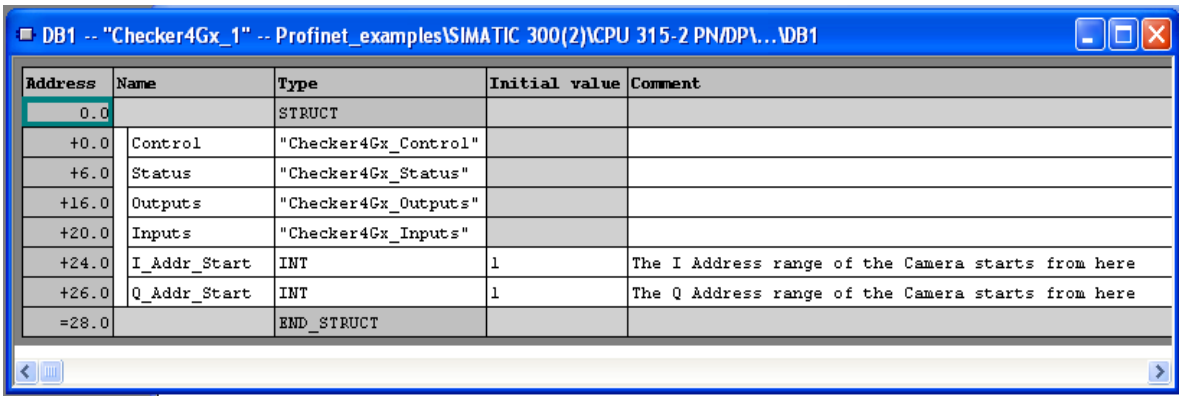
For more information on a program, open the program block either by right-clicking it and selecting **Called Block → Open** from the opened higher-level block or by right-clicking it and selecting **Open Object** in the list of blocks in the SIMATIC Manager window.

Using UDT

In the provided examples, the starting values for I/O addresses have been preconfigured, which you can verify in the HW Config view. See the following screenshot of the HW Config view.



If you want to use different hardware configurations (for example, you might want to modify the existing hardware configuration by adding a new Checker), you must set the proper I_Addr_Start and Q_Addr_Start values in the DB1 block.



DB1 -- "Checker4Gx_1" -- Profinet_examples\SIMATIC 300(2)\CPU 315-2 PN/DP\...\DB1

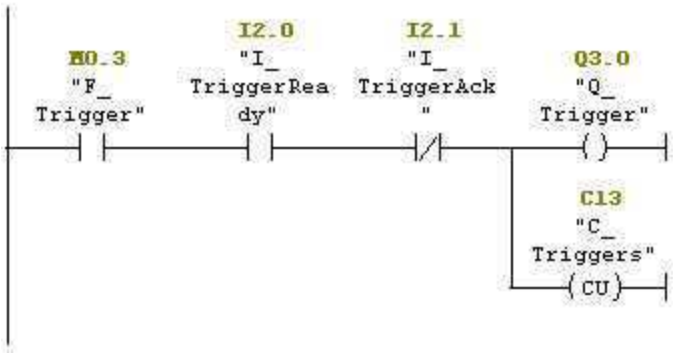
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Control	"Checker4Gx_Control"		
+6.0	Status	"Checker4Gx_Status"		
+16.0	Outputs	"Checker4Gx_Outputs"		
+20.0	Inputs	"Checker4Gx_Inputs"		
+24.0	I_Addr_Start	INT	1	The I Address range of the Camera starts from here
+26.0	Q_Addr_Start	INT	1	The Q Address range of the Camera starts from here
=28.0		END_STRUCT		

You might also want to use Checker4G7 instead of Checker4G1. In this case, you will have to exchange Checker4G1 with Checker4G7 in the HW Config view by first deleting Checker4G1 and then adding Checker4G7. If you first add the new Checker and then delete the old one, the starting memory addresses of the new Checker in the hardware configuration will not correspond to those in the DB1 block.

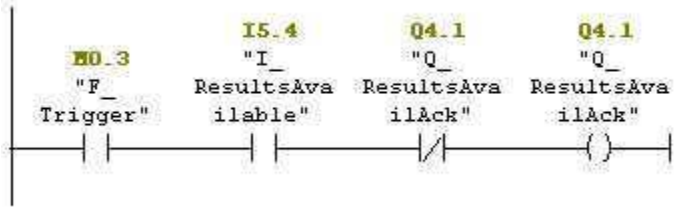
Example Notes

Be aware that in some cases you cannot effectively modify the values of certain variables because the PLC overwrites their values periodically with a short cycle time.

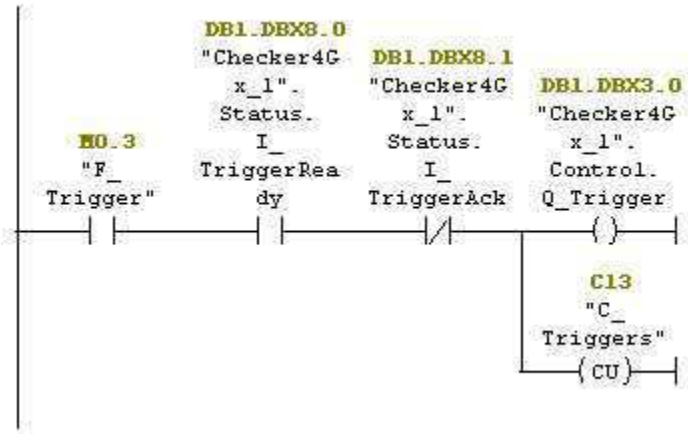
For example, you cannot modify elsewhere the trigger output (Q3.0) present in the following network:

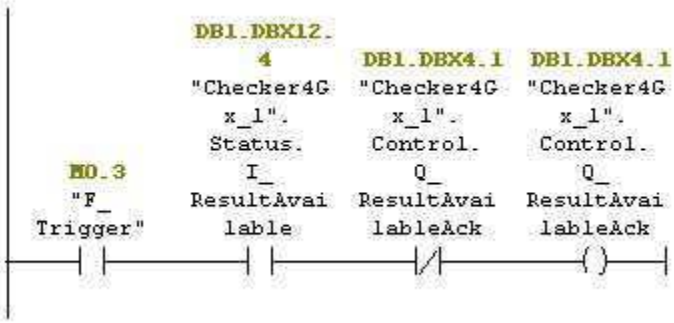


For another example, you cannot modify elsewhere the results available acknowledgement (Q4.1) present in the following network:



The following examples demonstrate the same as above using UDT database:





Creating PROFINET communication between Checker and the Siemens S7-1200 PLC

This section describes how to use Checker with a Siemens S7-1200 PLC. It is assumed that you are familiar with/has access to the S7-1200 and the Totally Integrated Automation (TIA) Portal.

1. Create a new project on the TIA portal.
2. Go to Devices and networks | Configure a device.
3. Click on Add new device and configure your S7 PLC.
4. Click Done and you will be navigated out of the Portal and inside Device View.
5. Choose Network view in the main project window.
6. Go to Catalog | Hardware catalog.
7. Look in other field devices/PROFINET IO/Sensors/Cognex Corp/Cognex ID Readers/Checker xxx and drag & drop that object into the main project window (somewhere close to the PLC drawing).

NOTE: If the reader is not yet available there, go to Options | Install general station description file (GSD) and install the GSD file that comes with the device. Once this is done, repeat Step 7.

8. Click once on the image of the Checker (a blue box appears around the image) and the small green rectangle (but not around the text close to the image).
9. In the Properties tab below there is General tab, choose Profinet interface [X1] ->Ethernet address. In Interface networked with click on Add new subnet. After that PN/IE_1 will be in the Subnet dropdown box.
10. Under IP protocol, check the Use IP protocol checkbox. Choose Set IP address using a different method (with this setting Checker keeps its IP address set in the Checker application, please use a static one.)
11. Most important: under PROFINET, set the PROFINET device name and Converted name to the reader's Profinet name. This must match the name of the actual Checker device (set via Checkmate). The name has to be unique and has to conform to DNS naming conventions.
12. Since you have clicked on Add new subnet there is now a green line originating from the small green rectangle in the Checker object (always in the network view). There you have a label PN/IE_1. Drag that label and drop it into the small green rectangle of the PLC object.
13. Now click on the blue Not assigned text near the Checker and select the IO (PLC) controller from the list.

Generic FFP

Generic FFP is a simple UDP/IP based protocol developed to allow Checker to communicate with PLCs without EtherNet/IP or PROFINET support.

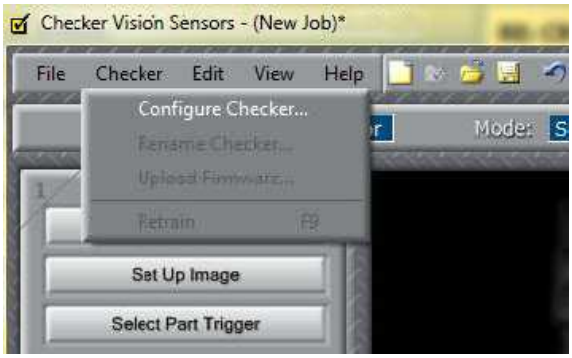
The basic communication scheme of Generic FFP is the same as of EtherNet/IP or PROFINET: lines are represented as bits in a byte array and peers interpret and react on changes as necessary.

Checker supports 24 virtual inputs and outputs through Generic FFP.

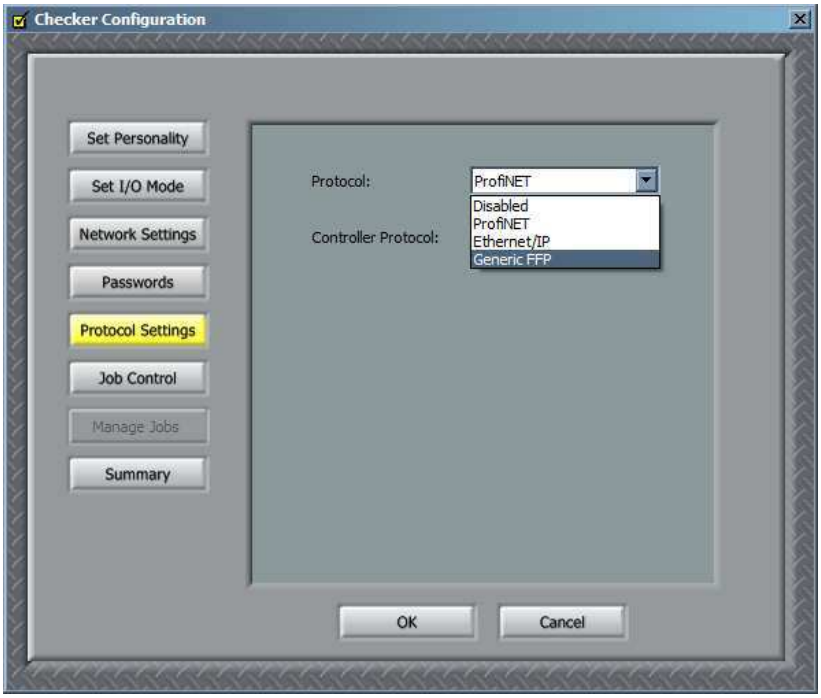
Enabling and Configuring Generic FFP in the Checker GUI

By default, Checker has Generic FFP disabled. To enable the protocol in the Checker GUI, perform the following steps:

1. In the upper menu toolbar, click Checker.
2. Select Configure Checker.



3. In the Checker Configuration window that pops up, click Protocol Settings and select Generic FFP.

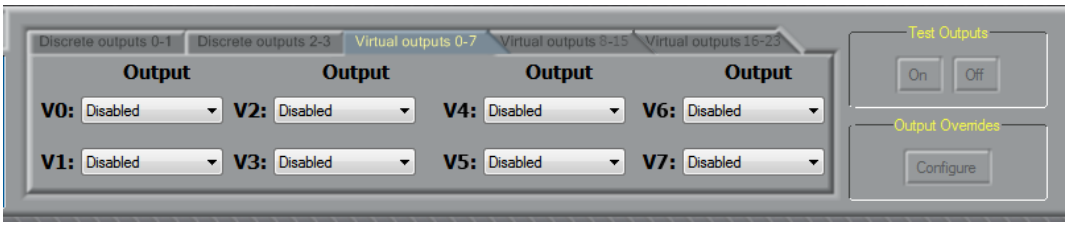


4. Click OK.

For the changes to take effect, the sensor automatically reboots.

24 Virtual Inputs and Outputs

The 24 virtual outputs can be configured in the Set Up Outputs step when Generic FFP is selected, see the following screenshot. These outputs are available to a PLC in the input assembly sent from the Checker.



The 24 virtual inputs can be selected in the ladder logic program. These inputs are set by a PLC and transmitted to the Checker in the output assembly.

Report Meter Values and Threshold Set-points

When Generic FFP is enabled, the input assembly contains the current values and threshold set-points for the sensors. Sensors can be configured to send their results via the input assembly that is sent from the Checker.

When a sensor is created, it is automatically added to a sensor meter table. Each sensor automatically has its values added to a table of results. The first entry contains the part finder value. If the part finder is not present, these entries are set to one (1).

The Sensor Meter table, available from the Set Up Outputs step, displays all sensors that have been added to the Checker job.

You can order the sensors by clicking a cell and dragging it to another cell. Cells do not need to be contiguously populated. Sensors that are not to be transmitted can be unchecked. They are displayed disabled.

Retraining Patterns

It is now possible to retrain a part finder or pattern sensor from a PLC. The PLC sets a bit in the output assembly and all retrainable sensors are retrained when this bit is set.

Upon completion of the retrain, a bit in the input assembly is set to indicate that the retrain action has occurred.

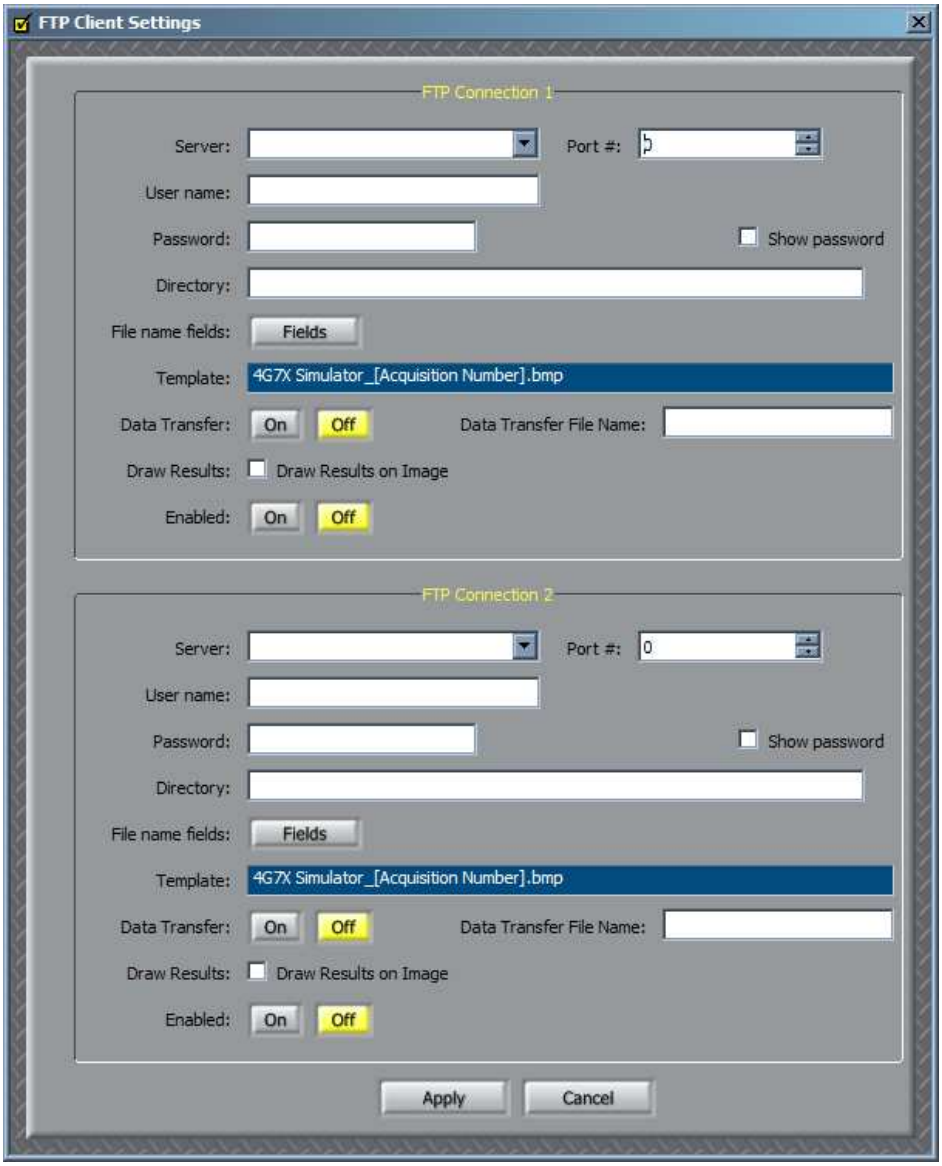
The Retrain Ack bit in the input assembly is set when a retrain is completed. Once the Retrain Ack bit is acknowledged by the PLC, the PLC can clear the Retrain bit in the output assembly. Once the Retrain bit has been cleared by the PLC, the Retrain Ack bit is cleared by the Checker.

Controlling Internal Illumination

A PLC is able to control the external illumination of a Checker by setting the Illumination Off bit in the output assembly. When set, the Checker will turn off the external illumination and disable strobe output. If the PC application connects to the Checker while the illumination is turned off and the PC application takes the Checker out of run mode, the LEDs are set back to the state that they are set to in the job.

FTP results data from Inspections

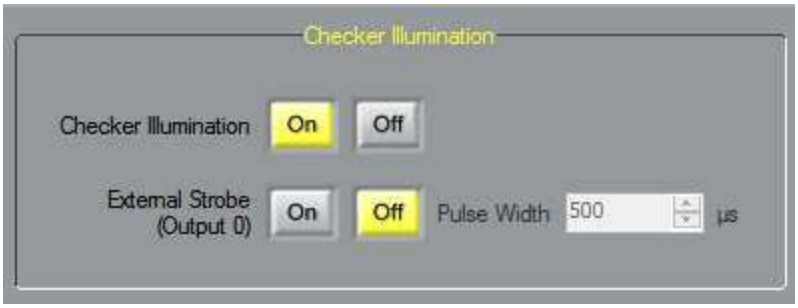
For FTP connections, you can enable the transmission of inspection results data via FTP. When Data Transfer is enabled, the results of each inspection are transmitted to an FTP server and appended to the file you indicate in the Data Transfer File name field. See the following screenshot.



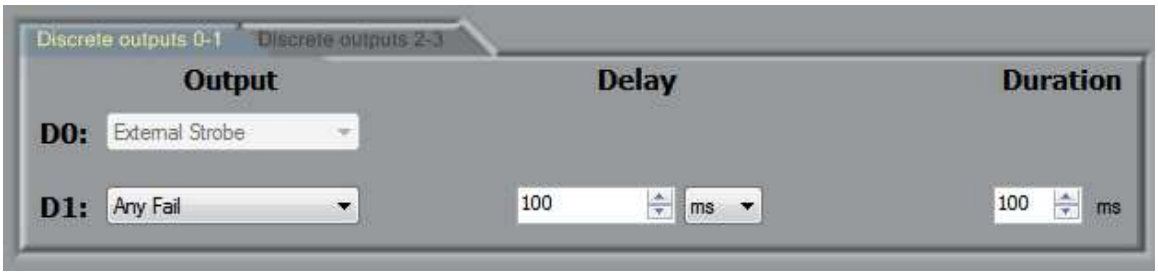
The FTP Data is transmitted using an XML formatted file.

Strobe Control

You can configure Checker to use an external strobe. When configured for external strobe, you lose one of the output lines reducing the number of outputs from 2 (Encoder mode) to 1 or from 4 to 3 (Job Change mode).



The external strobe, when enabled, uses Output 0 to trigger the strobe. This output is no longer available in the Set Up Outputs section. The Pulse Width value is the duration that the strobe is on. This value is an input to Checker and used to control the exposure/gain settings for the illumination slider. This value is not used by Checker for the duration of the output. The duration of the output is controlled by the illumination slider. Changing the illumination slider changes the exposure of the duration of the acquisition. While the Checker is acquiring, output D0 is enabled. This allows external light controllers in addition to external strobe controllers to be used.



Operation

The PLC (or PC) sends a plain binary UDP message to the Checker IP address to the 7733 port. The message should be a filled output assembly. Checker receives the data and replies by sending an input assembly immediately. There is no communication between two output assembly UDP packets; therefore, the period time is set dynamically by the master (PLC or PC).

Pseudocode for a simple usage cycle:

Loop

Create a socket with parameters AF_INET and SOCK_DGRAM

Set up output assembly buffer for 10 bytes:

- 2 bytes protocol identifiers:
  1. 0xAE – The communication magic to prevent corrupted messages from hitting the Checker accidentally.
  2. 0x01 – The protocol version number, which is 1 for now.
- 8 bytes payload

Send the buffer to Checker IP address, port 7733.

Receive 266 bytes from the same socket. (It is recommended that you set up a timeout for receiving because UDP packets could be lost somewhere between the sender and the receiver.)

Close the socket.

Wait according to your needs (this pause will set the period time).

Loop end

Input Assembly

The following table shows the Input Assembly (data sent from the Checker), where Reserved[1] means reserved for future use. If the Observer bit is set to 1, the Generic FFP controller cannot change the status of the Checker and can only observe its state.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
11	0	Online	Offline Reason • 0 – Online • 1 – Setup • 2-7 – Reserved			Missed Acq	Acquiring	Trigger Ack	Trigger Ready
	1	General Fault	Job Load Failed	Job Load Complete	Observer	Results Available	Results Buffer Overrun	Inspection Complete Toggle	Inspecting
	2	Reserved[1]	Reserved[1]	Reserved[1]	Retrain Failed	Retrain Completed	Any Fail	All Pass	Part Detect
	3	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
	4	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9	Output 8
	5	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17	Output 16
	6 7	Acquisition ID (16-bit integer)							
	8 9	Inspection Result ID (16-bit integer)							
	10	Sensor 1 Result Data 0							
	…	…							
	25	Sensor 1 Result Data 15							
	26	Sensor 2 Result Data 0							
	…	…							
	41	Sensor 2 Result Data 15							

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	42	Sensor 3 Result Data 0							
	…	…							
	409	Sensor 25 Result Data 15							
	410	Sensor 26 Result Data 0							
	…	…							
	425	Sensor 26 Result Data 15							

For information on the values returned in the Sensor Meter table, see the Static and Volatile Data Summary section.

Output Assembly

The following table shows the Output Assembly Instance (data sent to the Checker), where Reserved[1] means reserved for future use. Setting the Lights Off bit to 1 turns off the external illumination regardless of the job setting. Setting the SM Volatile bit to 0 selects the static results data that does not change for a job. Setting it to 1 selects the volatile data that varies from part to part. This data is returned in the Sensor Meter table values.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
21	0	Set Offline	Lights Off	Retrain	Job Change	Results Available Ack	Buffer Results Enable	Trigger	Trigger Enable
	1	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1	Input 0
	2	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9	Input 8
	3	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17	Input 16
	4	Job Number (8-bit integer)							
	5	Reserved[1]							SM Volatile
	6	Pad byte							
	7	Pad byte							

Checker Generic Input Command Handling

Definitions

Input commands

Checker accepts the following three input commands which can interfere with each other:

1. Trigger
   This command initiates the acquisition and the inspection that follows it.

2. Job Change
   It changes the Checker's current running job to a chosen one. The user can choose any jobs from the job slots.

3. Retrain
   It retrains the appropriate sensors on the last acquired image.

Input command sources

Checker can get input commands from the following sources:

1. Discrete source (cable)

2. FFP protocols: PROFINET/Ethernet IP/Generic FFP

3. PC application (In run mode it could be only a Retrain command)

Multiple commands at the same time

This can happen only with FFP communications. Since 3.6 it is possible to send more than one input command in the same network package and they will be executed in a predefined order.

The possible combinations are:

1. Trigger & Job Change => 1. Trigger, 2. Job Change

2. Trigger & Retrain => 1. Trigger, 2. Retrain

3. Job Change & Retrain => 1. Job Change (Retrain is dropped)

4. Trigger & Job Change & Retrain => 1. Trigger, 2. Job Change (Retrain is dropped)

Important Note: Two discrete signals sent by the PLC in the same time are not handled as multiple commands but as overlapping ones (see below) where the signal processing sequence is non-deterministic and one of the commands will be dropped!

Command overlap

In the case of command overlap, the following cases can occur:

1) A new input command arrives when an input command is under execution.
2) An FFP packet with multiple input commands arrives when an input command is under execution.
3) A delayed command is started when a command is under execution.

In all the cases the new command(s) will be simply declined and the error result will be propagated to the parties in interest, if any.

These error results can be: Missed Acquisition, Job Load Failed, Retrain Failed.

Port Usage

The following table lists the ports Checker uses for communication. If you are using a firewall, you must leave these ports open.

Protocol	**Port**
TCP and UDP for general Checker communication	• UDP: 1069 • TCP: 50053
EtherNet/IP	• UDP: 2222 • TCP: 44818
PROFINET	UDP: 4096, 4097, 4098, and 34964 Note: PROFINET uses raw Ethernet frames without IP/TCP/UDP headers.
Generic FFP	• UDP: 7733 • TCP: 7733
FTP	The default is port 21 passive but this can be changed in the FTP setup.

Appendix A – Protocol Settings, modifying the Observer bit

In the Checker PC Application, you can enable different industrial protocols for data transfer and for controlling your device. You can set these protocols at Checker | Configure Checker | Protocol Settings. The Generic FFP protocol is always present as well if you want to control your device with that. In order to avoid sending confusing (duplicated, controversial) control commands for Checker, the Observer bit has been introduced that separates the observer mode from the controller mode. In observer mode, the controller is not able to control the Checker, only it can observe its state. The Observer bit is 1 for the Observer and 0 for the Controller.

If you select the same industrial protocol for the Protocol AND the Controller Protocol as well then the Observer bit will be 0 for the set industrial protocol and 1 for Generic FFP. If you explicitly set Generic FFP as the Controller protocol then the Observer bit will be 0 for it, making it the controller. See the table below for the available variations:

Protocol Settings	Generic FFP (Observer bit value)	Ethernet/IP (Observer bit value)	Profinet (Observer bit value)
Protocol: ProfiNET Controller Protocol: ProfiNET	Observer (1)	N/A	Controller (0)
Protocol: ProfiNET Controller Protocol: Generic FFP	Controller (0)	N/A	Observer (1)
Protocol: Ethernet/IP Controller Protocol: Ethernet/IP	Observer (1)	Controller (0)	N/A
Protocol: Ethernet/IP Controller Protocol: Generic FFP	Controller (0)	Observer (1)	N/A
Protocol: Generic FFP Controller Protocol: Generic FFP	Controller (0)	N/A	N/A
Protocol: Disabled Controller Protocol: Generic FFP	Controller (0)	N/A	N/A

CONTACT US

di-soric GmbH & Co. KG
Steinbeisstraße 6
73660 Urbach
Germany
Fon: +49 (0) 71 81 / 98 79 - 0
Fax: +49 (0) 71 81 / 98 79 - 179
info@di-soric.com
www.di-soric.com

INTERNATIONAL

AUSTRIA
di-soric Austria GmbH & Co. KG
Burg 39
4531 Kematen an der Krems
Austria
Fon: +43 (0) 72 28 / 72 366
Fax: +43 (0) 72 28 / 72 366 - 4
info.at@di-soric.com

FRANCE
di-soric SAS
19, Chemin du Vieux Chêne
38240 Meylan
France
Fon: +33 (0) 4 76 / 61 65 90
Fax: +33 (0) 4 76 / 61 65 98
info.fr@di-soric.com

SINGAPORE
di-soric Pte. Ltd.
33 Ubi Avenue 3, #03-47 Vertex
Singapore 408868
Singapore
Fon: +65 / 66 34 38 43
Fax: +65 / 66 34 38 44
info.sg@di-soric.com

Sensors | Lighting | Vision | ID

www.di-soric.com