COGNEX®

Checker® 4G Communications and Programming Guide

2/25/2011
Version 0.10

Legal Notices

The software described in this document is furnished under license, and may be used or copied only in accordance with the terms of such license and with the inclusion of the copyright notice shown on this page. Neither the software, this document, nor any copies thereof may be provided to, or otherwise made available to, anyone other than the licensee. Title to, and ownership of, this software remains with Cognex Corporation or its licensor. Cognex Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Cognex Corporation. Cognex Corporation makes no warranties, either express or implied, regarding the described software, its merchantability, non-infringement or its fitness for any particular purpose.

The information in this document is subject to change without notice and should not be construed as a commitment by Cognex Corporation. Cognex Corporation is not responsible for any errors that may be present in either this document or the associated software.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, nor transferred to any other media or language without the written permission of Cognex Corporation.

Portions of the hardware and software provided by Cognex may be covered by one or more of the U.S. and foreign patents listed below as well as pending U.S. and foreign patents. Such pending U.S. and foreign patents issued after the date of this document are listed on Cognex web site at http://www.cognex.com/patents.

VisionPro

5481712, 5495537, 5548326, 5583954, 5602937, 5640200, 5751853, 5768443, 5825913, 5850466, 5872870, 5901241, 5943441, 5978080, 5978521, 5987172, 6005978, 6039254, 6064388, 6075881, 6137893, 6141033, 6167150, 6215915, 6240208, 6324299, 6381366, 6381375, 6411734, 6421458, 6459820, 6490375, 6516092, 6563324, 6658145, 6687402, 6690842, 6697535, 6718074, 6748110, 6771808, 6804416, 6836567, 6850646, 6856698, 6920241, 6959112, 6963338, 6973207, 6975764, 6985625, 6993177, 6993192, 7006712, 7016539, 7043081, 7058225, 7065262, 7088862, 7164796, 7190834, 7242801, 7251366, 7313761, EP0713593, JP3522280, JP3927239

DataMan

5742037, 5943441, 6215915, 6236769, 6282328, 6381375, 6408109, 6457032, 6690842, 6941026, 7175090, 7181066, 7412106, 7427028, 7549582, 7604174, 7614563, 7617984, US-2005-0087601-A1, US-2006-0131418-A1, US-2006-0131419-A1, US-2006-0133757-A1, US-2007-0090193-A1, US-2007-0091332-A1, US-2007-0152064-A1, US-2007-0170259-A1, US-2008-0004822-A1, US-2008-0011855-A1, US-2008-0142604-A1, US-2008-0143838-A1, US-2008-0158365-A1, US-2009-0090781-A1, US-2009-0108073, US-2009-0121027-A1, US-2009-0166424-A1, US-2009-0294541-A1, WO06065619A1, EP1687752

CVL

5495537, 5548326, 5583954, 5602937, 5640200, 5717785, 5751853, 5768443, 5825483, 5825913, 5850466, 5859923, 5872870, 5901241, 5943441, 5949905, 5978080, 5987172, 5995648, 6002793, 6005978, 6064388, 6067379, 6075881, 6137893, 6141033, 6157732, 6167150, 6215915, 6240208, 6240218, 6324299, 6381366, 6381375, 6408109, 6411734, 6421458, 6457032, 6459820, 6490375, 6516092, 6563324, 6658145, 6687402, 6690842, 6718074, 6748110, 6751361, 6771808, 6798925, 6804416, 6836567, 6850646, 6856698, 6920241, 6959112, 6975764, 6985625, 6993177, 6993192, 7006712, 7016539, 7043081, 7058225, 7065262, 7088862, 7164796, 7190834, 7242801, 7251366, EP0713593, JP3522280, JP3927239

VGR

5495537, 5602937, 5640200, 5768443, 5825483, 5850466, 5859923, 5949905, 5978080, 5995648, 6002793, 6005978, 6075881, 6137893, 6141033, 6157732, 6167150, 6215915, 6324299, 6381375, 6408109, 6411734, 6421458, 6457032, 6459820, 6490375, 6516092, 6563324, 6658145, 6690842, 6748110, 6751361, 6771808, 6804416, 6836567, 6850646, 6856698, 6959112, 6975764, 6985625, 6993192, 7006712, 7016539, 7043081, 7058225, 7065262, 7088862, 7164796, 7190834, 7242801, 7251366

OMNIVIEW

6215915, 6381375, 6408109, 6421458, 6457032, 6459820, 6594623, 6804416, 6959112, 7383536

CVL Vision Library

5495537, 5548326, 5583954, 5602937, 5640200, 5717785, 5751853, 5768443, 5825483, 5825913, 5850466, 5859923, 5872870, 5901241, 5943441, 5949905, 5978080, 5987172, 5995648, 6002793, 6005978, 6064388, 6067379, 6075881, 6137893, 6141033, 6157732, 6167150, 6215915, 6240208, 6240218, 6324299, 6381366, 6381375, 6408109, 6411734, 6421458, 6457032, 6459820, 6490375, 6516092, 6563324, 6658145, 6687402, 6690842, 6718074, 6748110, 6751361, 6771808, 6798925, 6804416, 6836567, 6850646, 6856698, 6920241, 6959112, 6975764, 6985625, 6993177, 6993192, 7006712, 7016539, 7043081, 7058225, 7065262, 7088862, 7164796, 7190834, 7242801, 7251366, EP0713593, JP3522280, JP3927239

SMD 4

5995648, 5850466, 6751361, 6690842, 6563324, 6490375, 5949905, 5978080, 6137893, 6167150, 6075881, 6748110, 5859923, 6411734, 6324299, 6516092, 7190834, 6658145, 6836567, 6850646, 6975764, 6985625, 6993192, 7006712, 7043081, 7058225, 7065262, 7088862, 7164796, 7251366, 6856698, 6002793, 6005978, 6771808, 6804416, 7016539, 6959112, 5602937, 7242801, 5640200, 5495537, 5768443, 5825483, 6421458, 6459820, 6215915, 6381375, 6457032, 6157732, 6408109, 6141033, 6026176, 6442291, 6151406, 6396942, 6614926, 5371690, 5845007, 5943441, 6963338, 5805722, 5909504, 5933523, 5964844, 5974169, 5987172, 6078700, 6252986, 6278796, 6307210, 6408429, 6424734, 6526165, 6571006, 6639624, 6681039, 6748104, 6813377, 6853751, 6898333, 6950548, 6993177, 7139421, 5757956

BGA II and BGA III

5495537, 5602937, 5640200, 5768443, 5801966, 5825483, 5850466, 5859923, 5949905, 5978080, 5995648, 6002793, 6005978, 6026176, 6055328, 6075881, 6115042, 6118893, 6130959, 6137893, 6141009, 6141033, 6151406, 6157732, 6167150, 6215915, 6289117, 6324299, 6353676, 6381375, 6396942, 6408109, 6411734, 6421458, 6442291, 6457032, 6459820, 6490375, 6516092, 6563324, 6577775, 6614926, 6658145, 6690842, 6748110, 6751361, 6771808, 6804416, 6836567, 6850646, 6856698, 6959112, 6975764, 6985625, 6993192, 7006712, 7016539, 7043081, 7058225, 7065262, 7088862, 7164796, 7190834, 7242801, 7251366

Wire Bonder

5495537, 5532739, 5581632, 5602937, 5640199, 5640200, 5642158, 5676302, 5754679, 5757956, 5768443, 5825483, 5835622, 5850466, 5859923, 5861909, 5949905, 5978080, 5991436, 5995648, 6002793, 6005978, 6035066, 6061467, 6075881, 6137893, 6141033, 6157732, 6167150, 6215915, 6289492, 6324299, 6381375, 6408109, 6411734, 6421458, 6457032, 6459820, 6490375, 6516092, 6563324, 6658145, 6690842, 6748110, 6751361, 6771808, 6804416, 6836567, 6850646, 6856698, 6959112, 6975764, 6985625, 6993192, 7006712, 7016539, 7043081, 7058225, 7065262, 7088862, 7164796, 7171036, 7190834, 7242801, 7251366

The following are registered trademarks of Cognex Corporation:

acuReader® BGAII® Check it with Checker® Checker® Cognex Vision for Industry CVC-1000® CVL® DataMan® DisplayInspect® DVT® EasyBuilder® IDMax® In-SightIn-Sight 2000® In-Sight® (insignia with cross-hairs) MVS-8000® OmniView® PatFind® PatFlex® PatInspect® PatMax® PatQuick® SensorView® SmartLearn® SmartView® SMD4® UltraLight® Vision Solutions® VisionPro® VisionView®

The following are trademarks of Cognex Corporation:

3D-Locate™ 3DMax™ CheckPoint™ Cognex VSoC™ FFD™ iLearn™ InspectEdge™ Legend™ LineMax™ NotchMax™ ProofRead™ SmartAdvisor™ SmartSync™ SmartSystem™

Other product and company names mentioned herein are the trademarks, or registered trademarks, of their respective owners.

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

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.

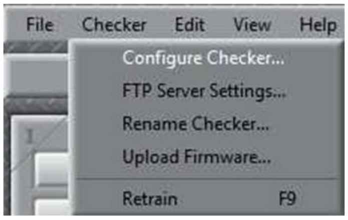
If no Link Local Addressing occurs, the Checker 4G sensor waits for a DHCP server to assign an address.

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.

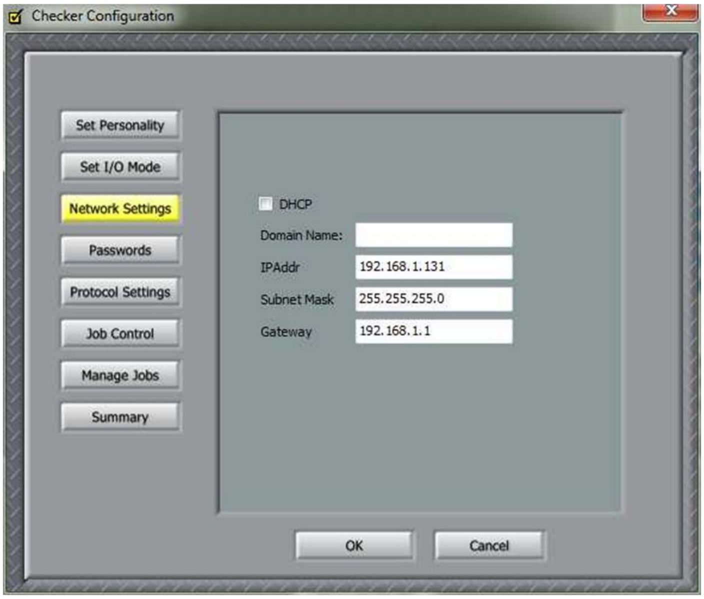
Static IP Address Assignment

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.

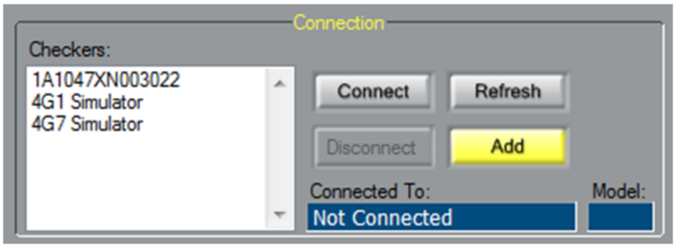


☐ DHCP
Domain Name:
IPAddr 192.168.1.131
Subnet Mask 255.255.255.0
Gateway 192.168.1.1

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.

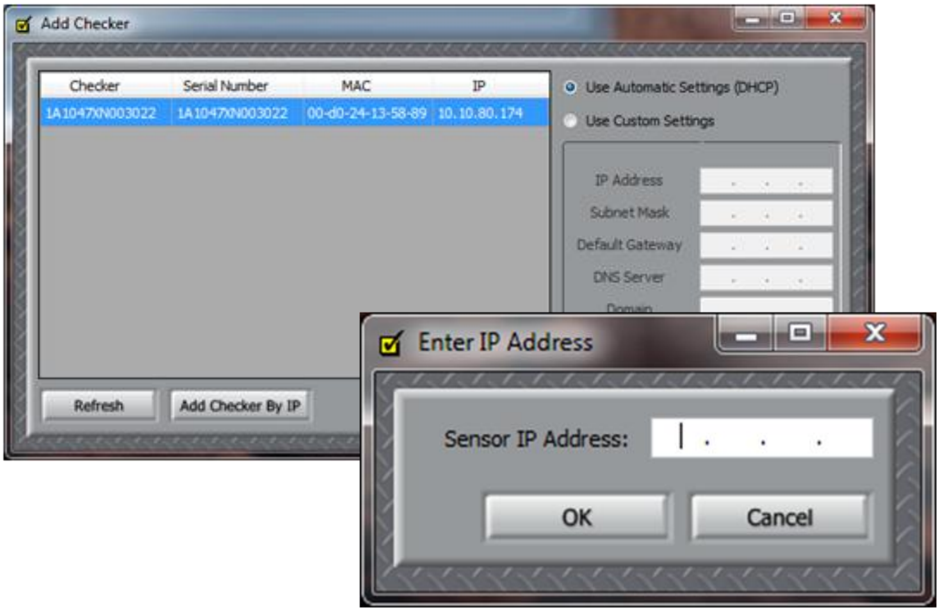
1. In the **Get Connected** step, click **Add** in the **Connection** frame.



Checkers:
1A1047XN003022
4G1 Simulator
4G7 Simulator

Connected To: Not Connected
Model:

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



Checker	Serial Number	MAC	IP
1A1047XN003022	1A1047XN003022	00-d0-24-13-58-89	10.10.80.174

● Use Automatic Settings (DHCP)

○ Use Custom Settings

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.

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

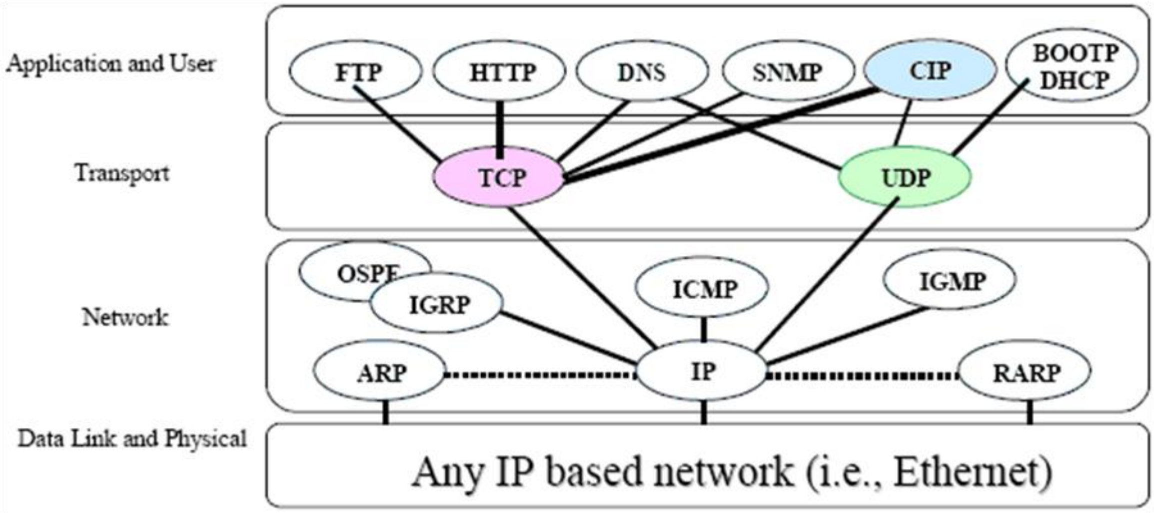
EtherNet/IP is a communication system suitable for use in industrial environments. EtherNet/IP allows industrial devices to exchange time-critical application information. These devices include simple I/O devices such as sensors/actuators, as well as complex control devices such as robots, programmable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

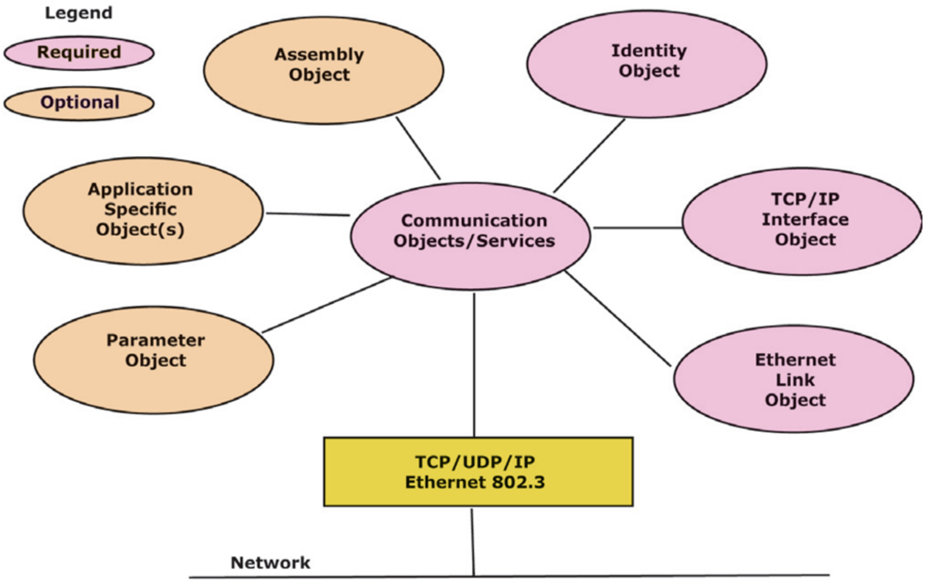
On Ethernet CIP straddles both UDP and TCP. CIP “explicit” messaging uses TCP to pass non-time critical point-to-point messages and commands from one device to another. CIP “implicit” messaging uses UDP for time-critical control and I/O.



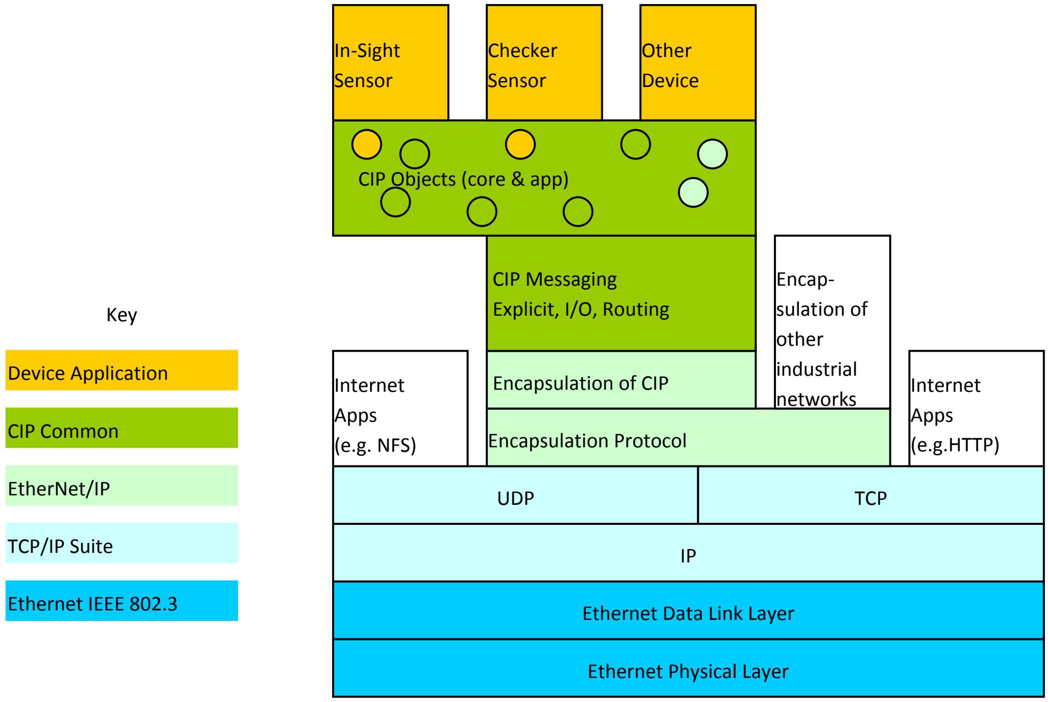
CIP is based on an object model. These objects encapsulate a device’s data and functionality (at least those portions you wish to expose to the outside world). The model provides a common way to access this data and functionality.

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)

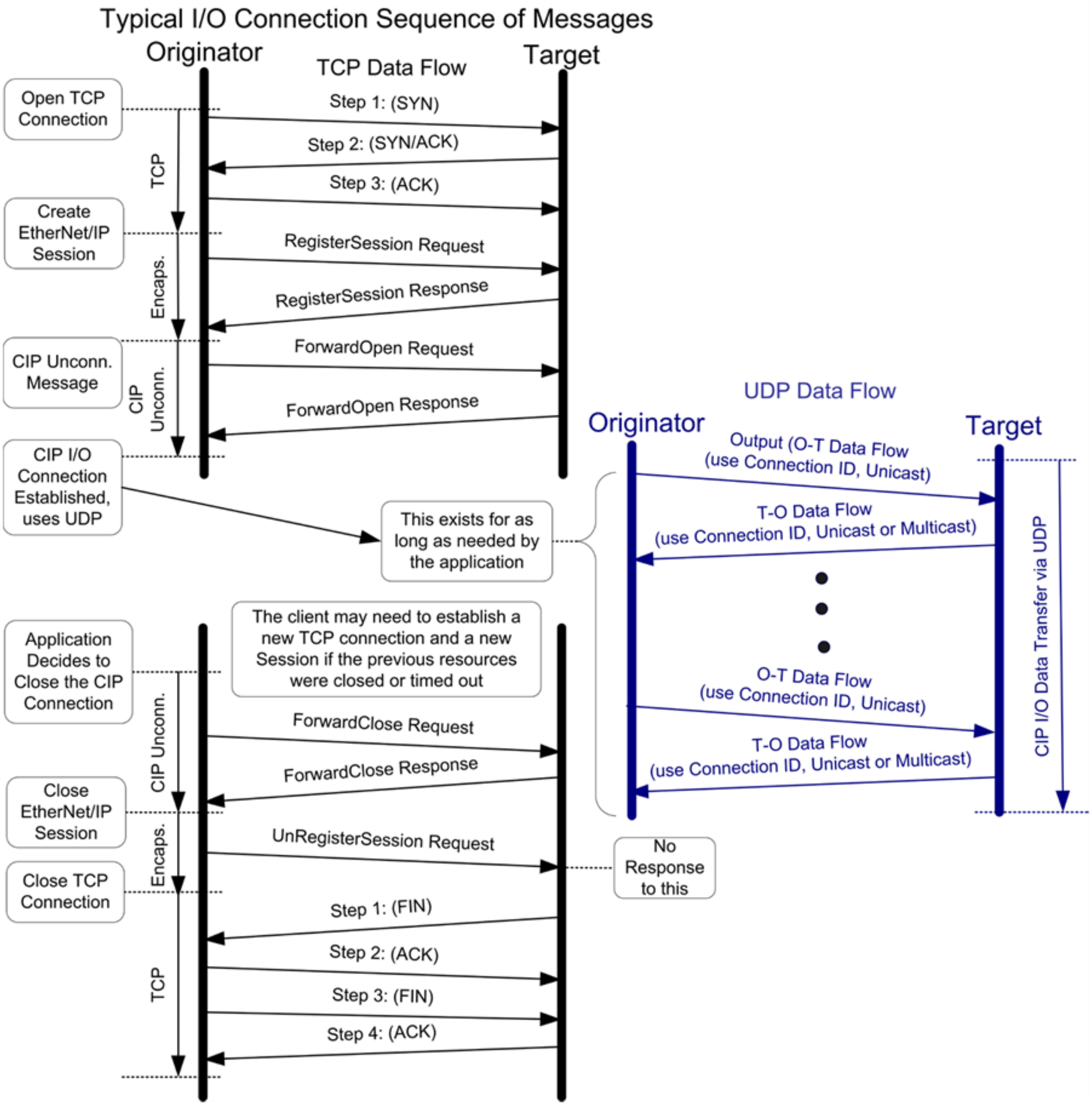
Implicit Messaging

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.

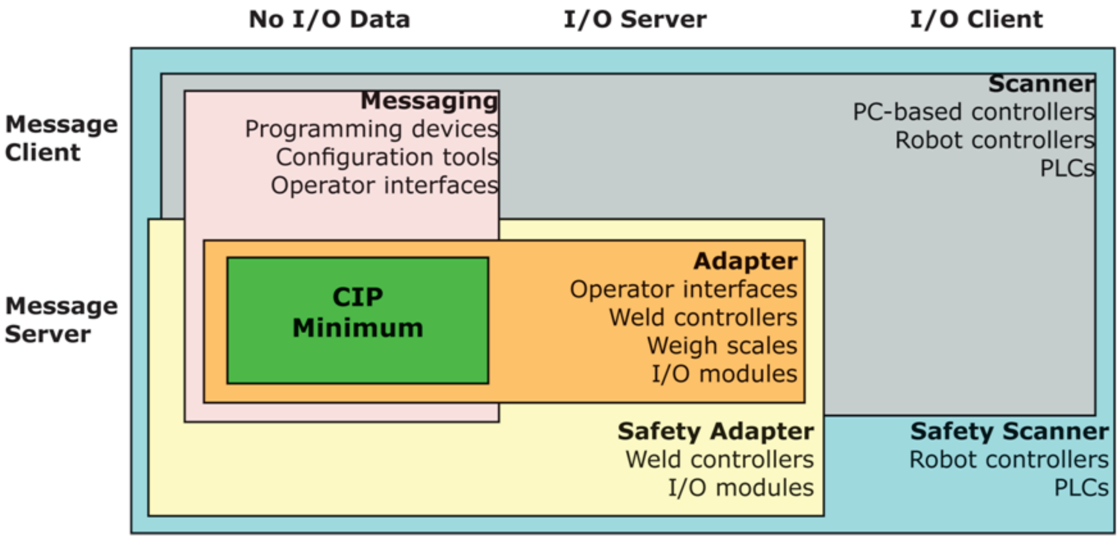

Typical I/O Connection Sequence of Messages
Originator
TCP Data Flow
Target
Open TCP Connection
Step 1: (SYN)
Step 2: (SYN/ACK)
Step 3: (ACK)
TCP
Create EtherNet/IP Session
RegisterSession Request
RegisterSession Response
Encaps.
CIP Unconn. Message
ForwardOpen Request
ForwardOpen Response
CIP Unconn.
CIP I/O Connection Established, uses UDP
This exists for as long as needed by the application
UDP Data Flow
Originator
Target
Output (O-T Data Flow (use Connection ID, Unicast)
T-O Data Flow (use Connection ID, Unicast or Multicast)
O-T Data Flow (use Connection ID, Unicast)
T-O Data Flow (use Connection ID, Unicast or Multicast)
CIP I/O Data Transfer via UDP
Application Decides to Close the CIP Connection
The client may need to establish a new TCP connection and a new Session if the previous resources were closed or timed out
ForwardClose Request
ForwardClose Response
CIP Unconn.
Close EtherNet/IP Session
UnRegisterSession Request
No Response to this
Encaps.
Close TCP Connection
Step 1: (FIN)
Step 2: (ACK)
Step 3: (FIN)
Step 4: (ACK)
TCP

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. For example, In-Sight is primarily classified as an I/O Adapter. It also has I/O Scanner capability which is used to connect to its I/O module.

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

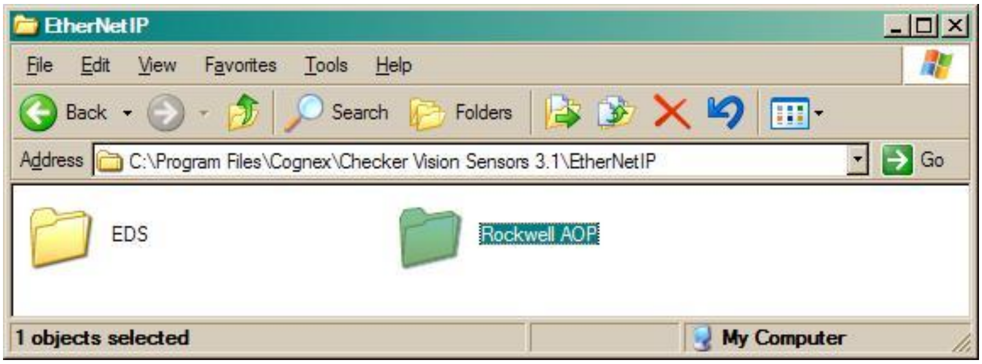
Checker 4G supports EtherNet/IP, an application level protocol based on the Common Industrial Protocol (CIP). EtherNet/IP provides an extensive range of messaging options and services for the transfer of data and I/O over Ethernet. All devices on an EtherNet/IP network present their data to the network 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:

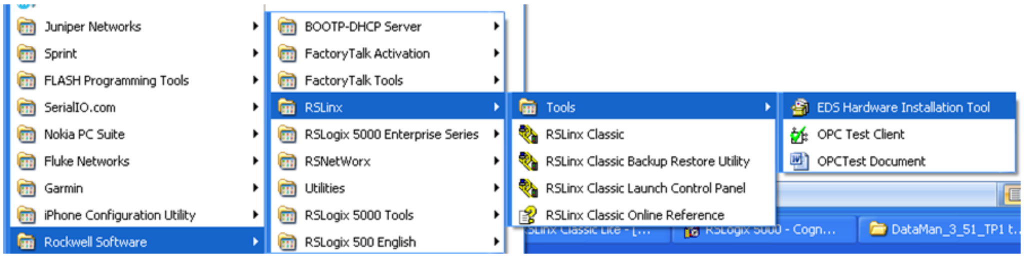
1. Verify that the Rockwell Software is on your PC.
2. Make sure you select the Add on Profile installation and the Samples installation. Add on Profile is only used with Rockwell ControlLogix or CompactLogix PLCs.
3. Install the Rockwell Add on Profiles by navigating to the following directory:



C:\Program Files\Cognex\Checker Vision Sensors 3.1\EtherNetIP

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

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



7. Run the ESD Install tool.

NOTE

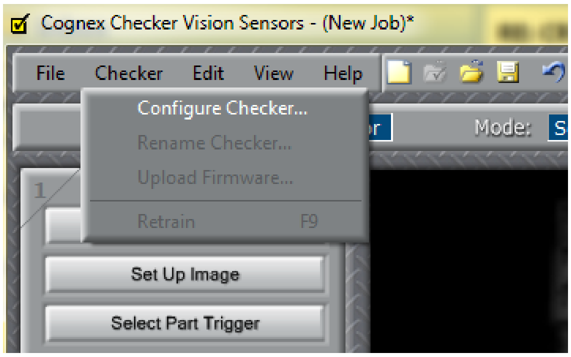
If you have an existing EDS file, uninstall it first, then install the latest version of the EDS.

8. 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.

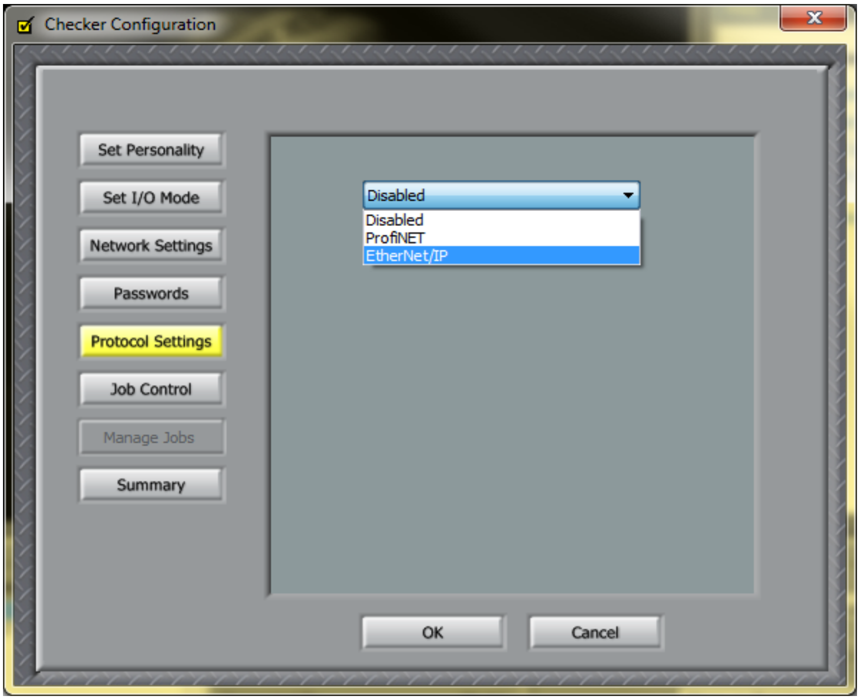
9. By default the Checker 4G has the EtherNet/IP protocol disabled. To enable the protocol, perform the following steps:

   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.



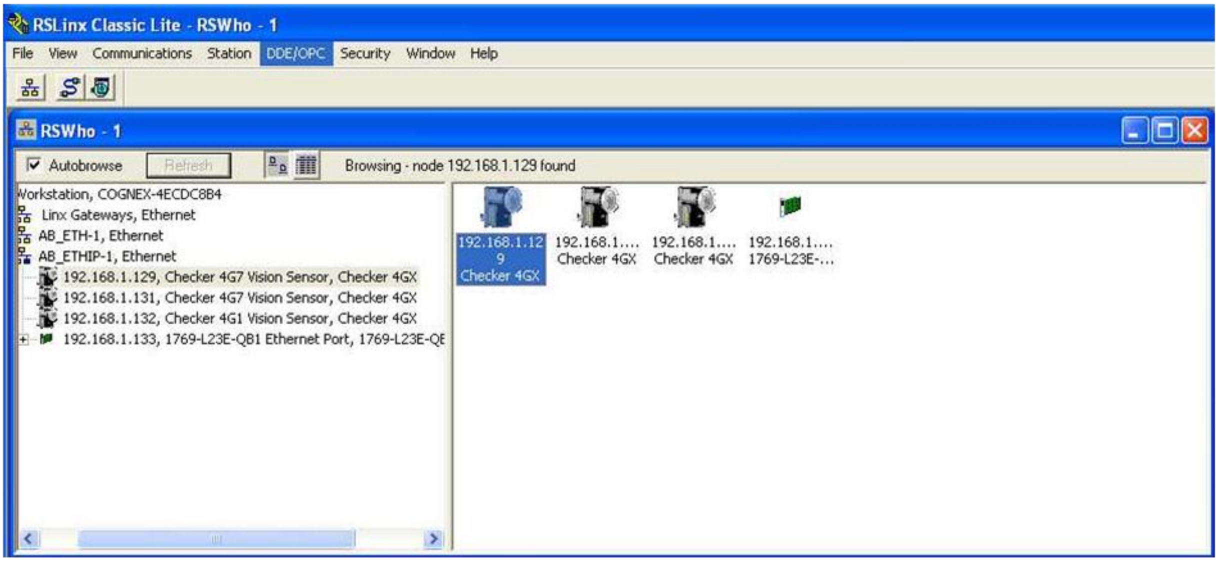
Disabled

Disabled
ProfiNET
EtherNet/IP

d. Click OK.

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

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


RSLinx Classic Lite - RSWho - 1
File View Communications Station DDE/OPC Security Window Help
RSWho - 1
Autobrowse
Refresh

☑ Autobrowse

Browsing - node 192.168.1.129 found

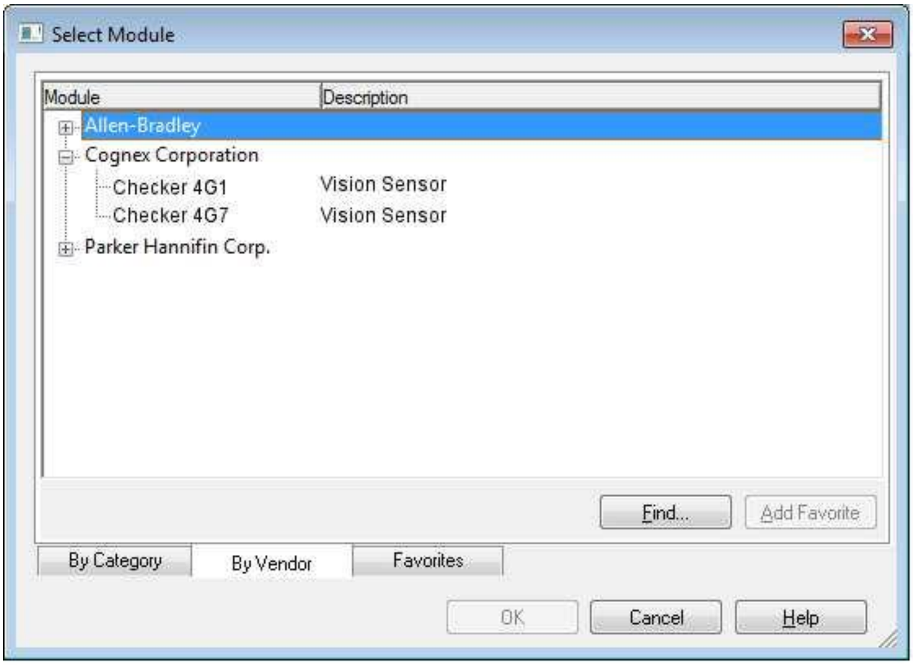
```
Workstation, COGNEX-4ECDC8B4
  Linx Gateways, Ethernet
  AB_ETH-1, Ethernet
  AB_ETHIP-1, Ethernet
    192.168.1.129, Checker 4G7 Vision Sensor, Checker 4GX
    192.168.1.131, Checker 4G7 Vision Sensor, Checker 4GX
    192.168.1.132, Checker 4G1 Vision Sensor, Checker 4GX
    192.168.1.133, 1769-L23E-QB1 Ethernet Port, 1769-L23E-QE
```

192.168.1.129 Checker 4GX
192.168.1.... Checker 4GX
192.168.1.... Checker 4GX
192.168.1.... 1769-L23E-...

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

11. 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.



Module	Description
Allen-Bradley	
Cognex Corporation	
Checker 4G1	Vision Sensor
Checker 4G7	Vision Sensor
Parker Hannifin Corp.	

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.

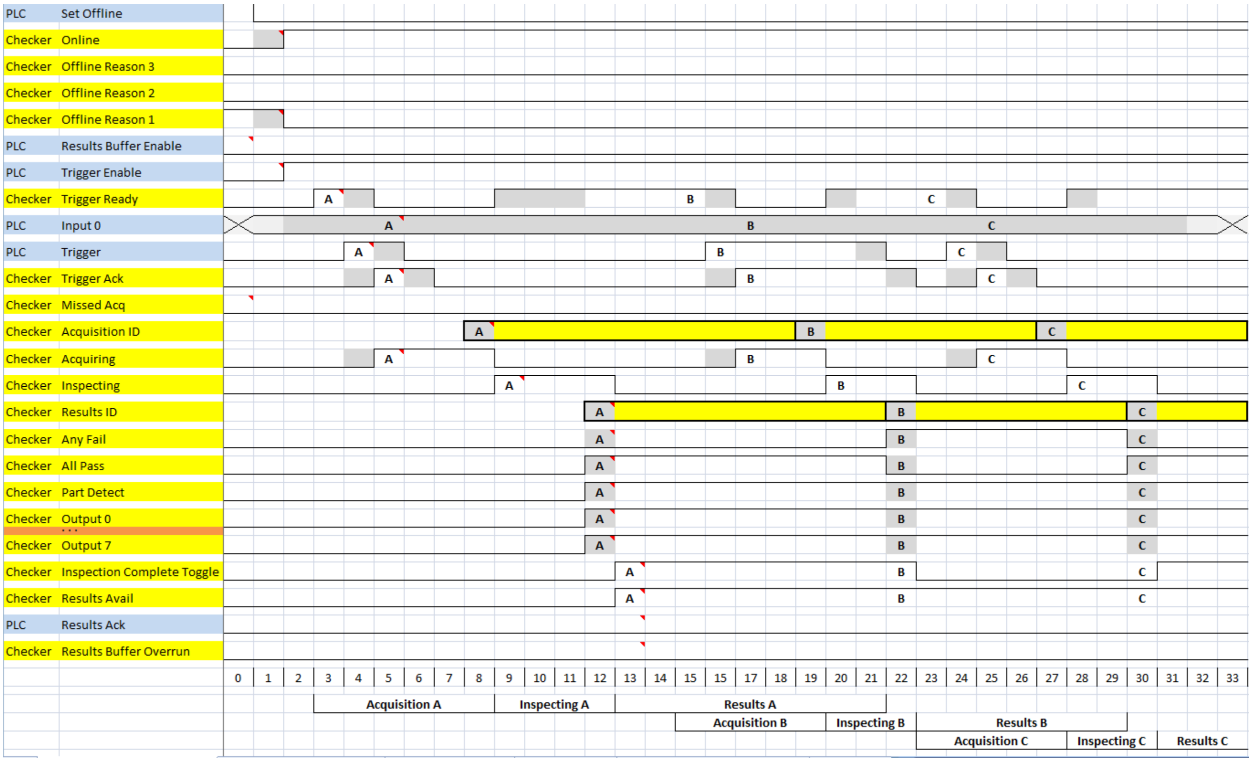
On startup the TriggerEnable attribute will be false. It must be set to true to enable triggering. When the device is ready to accept triggers, the Trigger Ready bit will be set to true.

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.

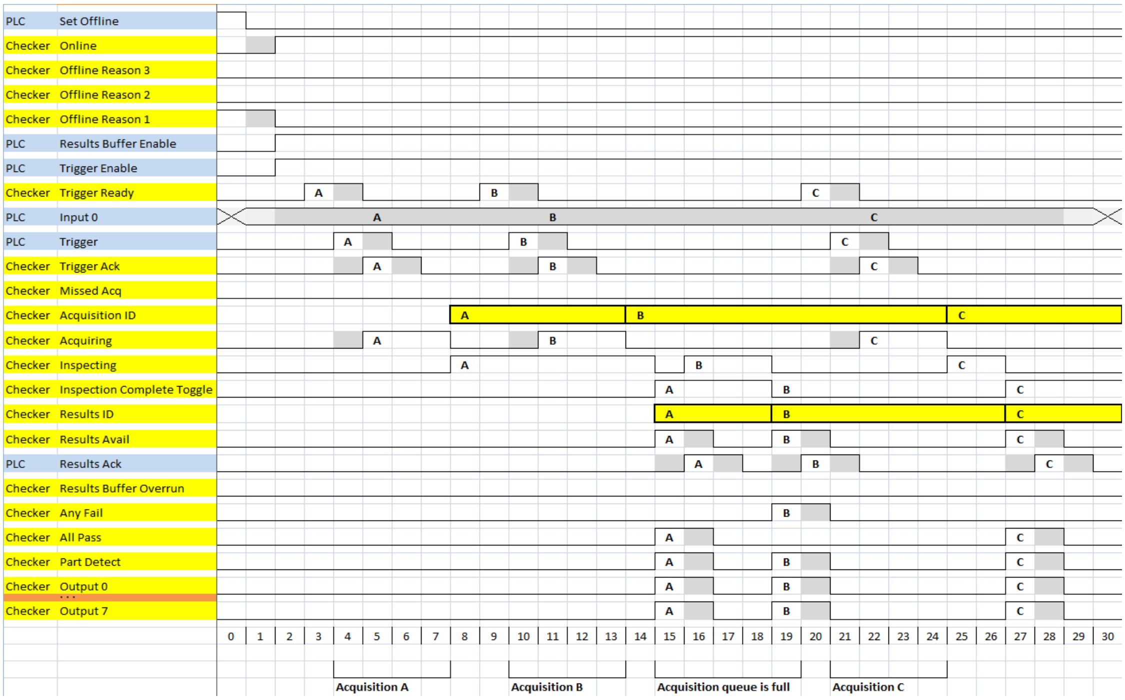
NOTE

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.

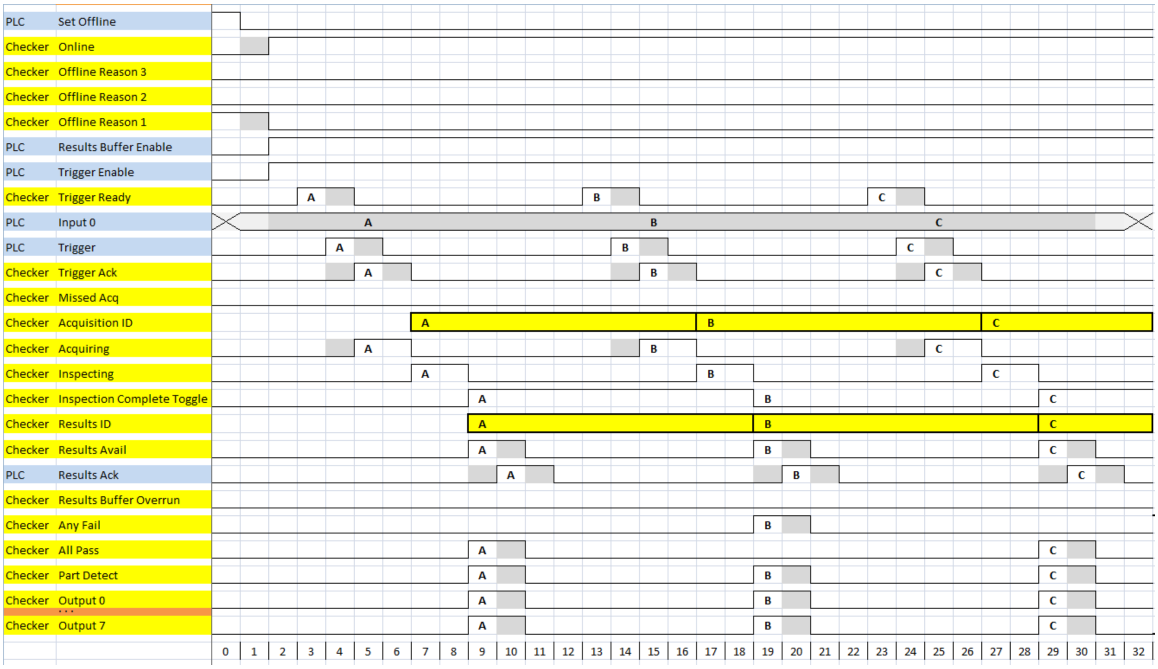
If the BufferResultsEnable attribute is set to true the new results are queued. When using internal triggering, the next trigger will overwrite the results. When using external triggering, the next external trigger will overwrite the result even if it is not used by the application.

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.

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
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).
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.



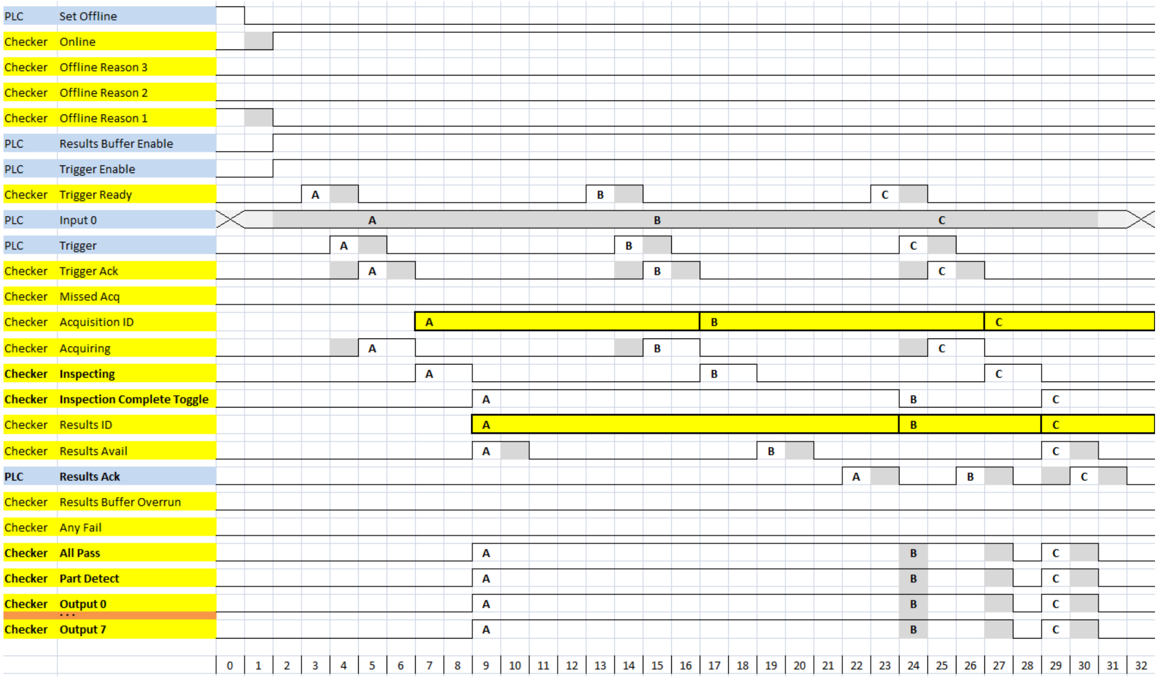
The states of the Any Fail, Any Pass, Part Detect, and Output 0 to 7 lines depend on the results of the inspection. The states shown are just examples.

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 means reserved for future use.

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	Reserved	Results Available	Results Buffer Overrun	Inspection Complete Toggle	Inspecting
	2	Reserved	Reserved	Reserved	Reserved	Reserved	Any Fail	All Pass	Part Detect
	3	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1	Output 0
	4	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	5	Reserved							
	6	Acquisition ID (16-bit integer)							
	7								
	8	Inspection Result ID (16-bit integer)							
	9								
	10	Reserved							
	…	Reserved							
	499	Reserved							

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 means reserved for future use.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
21	0	Set Offline	Reserved		Job Change	Results Available Ack	Buffer Results Enable	Trigger	Trigger Enable
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input 0
	2	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	3	Reserved							
	4	Job Number (8-bit integer)							
	5	Reserved							
	…	Reserved							
	499	Reserved							

Checker Implementation

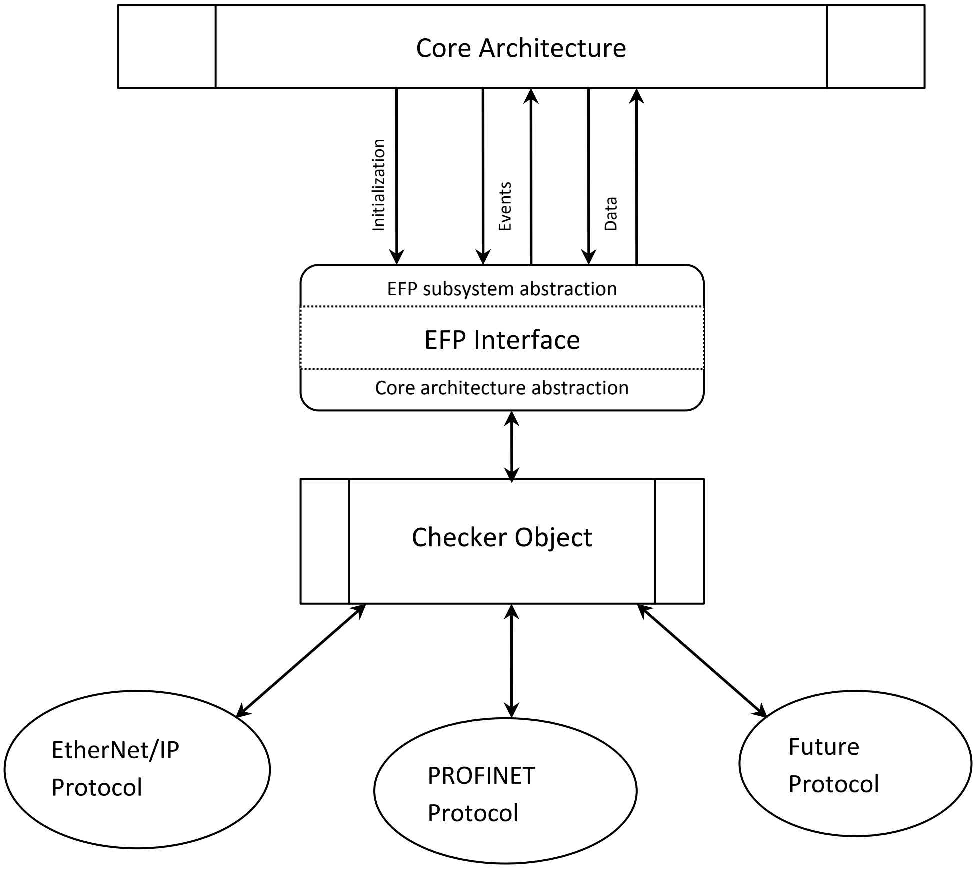
Ethernet Factory Protocol Subsystem

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.



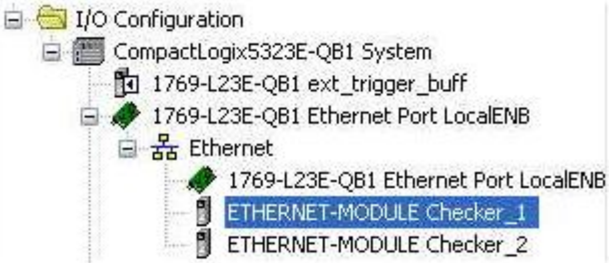
Establishing a Generic Implicit Messaging Connection

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:

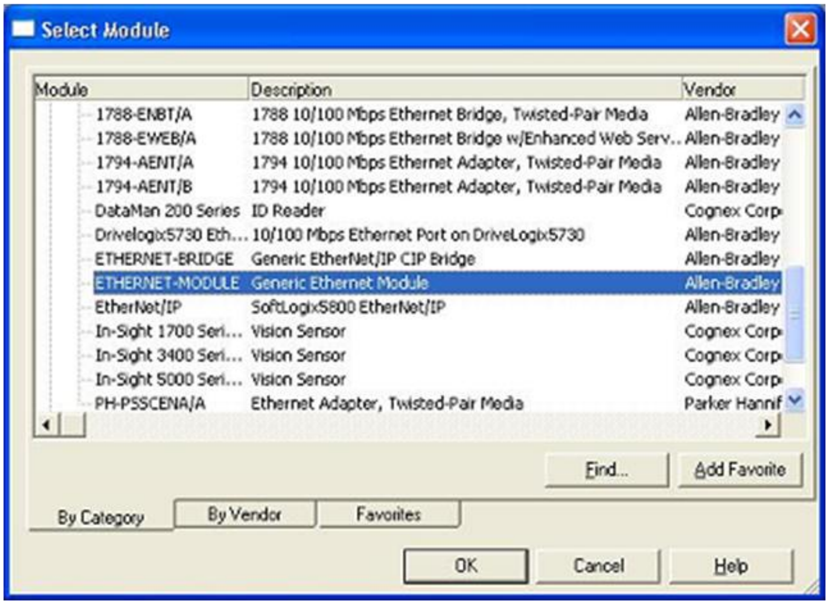
1. Open RSLogix5000 and load your project (or select **File**->**New…** to create a new one).

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:



I/O Configuration
CompactLogix5323E-QB1 System
1769-L23E-QB1 ext_trigger_buff
1769-L23E-QB1 Ethernet Port LocalENB
Ethernet
1769-L23E-QB1 Ethernet Port LocalENB
ETHERNET-MODULE Checker_1
ETHERNET-MODULE Checker_2

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

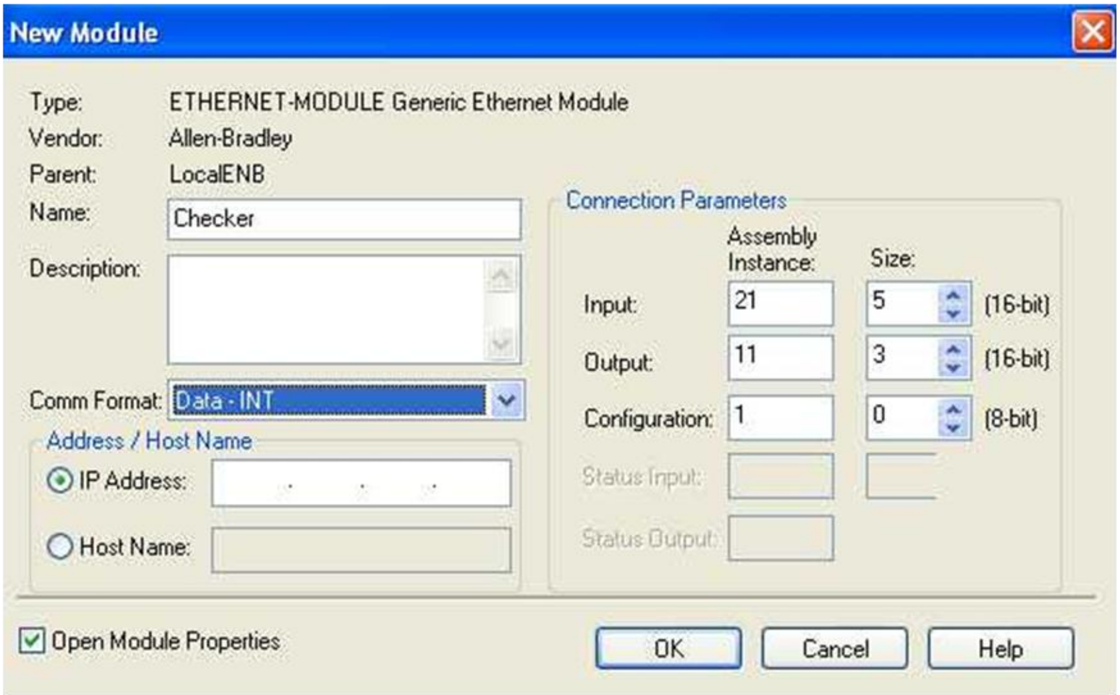


Select Module

Module	Description	Vendor
1788-ENBT/A	1788 10/100 Mbps Ethernet Bridge, Twisted-Pair Media	Allen-Bradley
1788-EWEB/A	1788 10/100 Mbps Ethernet Bridge w/Enhanced Web Serv...	Allen-Bradley
1794-AENT/A	1794 10/100 Mbps Ethernet Adapter, Twisted-Pair Media	Allen-Bradley
1794-AENT/B	1794 10/100 Mbps Ethernet Adapter, Twisted-Pair Media	Allen-Bradley
DataMan 200 Series	ID Reader	Cognex Corp
Drivelogix5730 Eth...	10/100 Mbps Ethernet Port on DriveLogix5730	Allen-Bradley
ETHERNET-BRIDGE	Generic EtherNet/IP CIP Bridge	Allen-Bradley
ETHERNET-MODULE	Generic Ethernet Module	Allen-Bradley
EtherNet/IP	SoftLogix5800 EtherNet/IP	Allen-Bradley
In-Sight 1700 Seri...	Vision Sensor	Cognex Corp
In-Sight 3400 Seri...	Vision Sensor	Cognex Corp
In-Sight 5000 Seri...	Vision Sensor	Cognex Corp
PH-PSSCENA/A	Ethernet Adapter, Twisted-Pair Media	Parker Hannif

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).



Type: ETHERNET-MODULE Generic Ethernet Module

Vendor: Allen-Bradley

Parent: LocalENB

Name: Checker

Description:

Comm Format: Data - INT

Address / Host Name

● IP Address: . . .

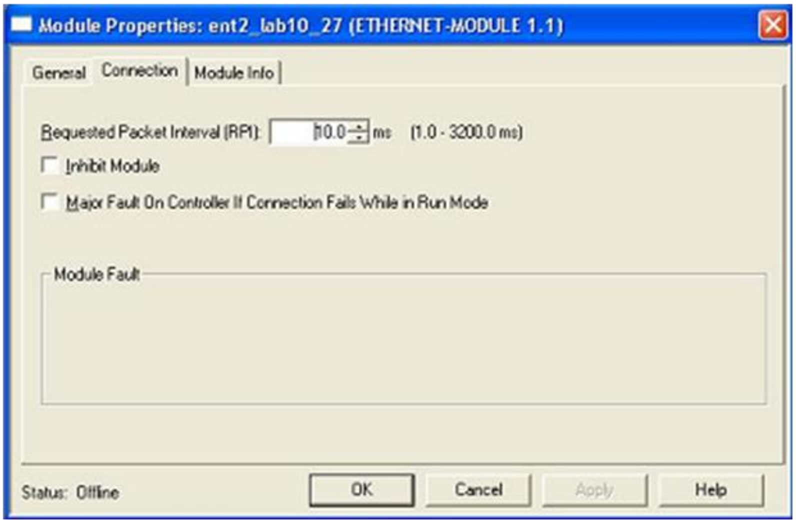
○ Host Name:

Connection Parameters

	Assembly Instance:	Size:	
Input:	21	5	(16-bit)
Output:	11	3	(16-bit)
Configuration:	1	0	(8-bit)
Status Input:			
Status Output:			

☑ Open Module Properties

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.



Requested Packet Interval (RPI): 10.0 ms (1.0 - 3200.0 ms)

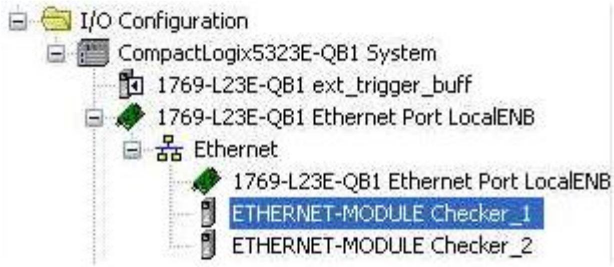
□ Inhibit Module

□ Major Fault On Controller If Connection Fails While in Run Mode

Module Fault

Status: Offline

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

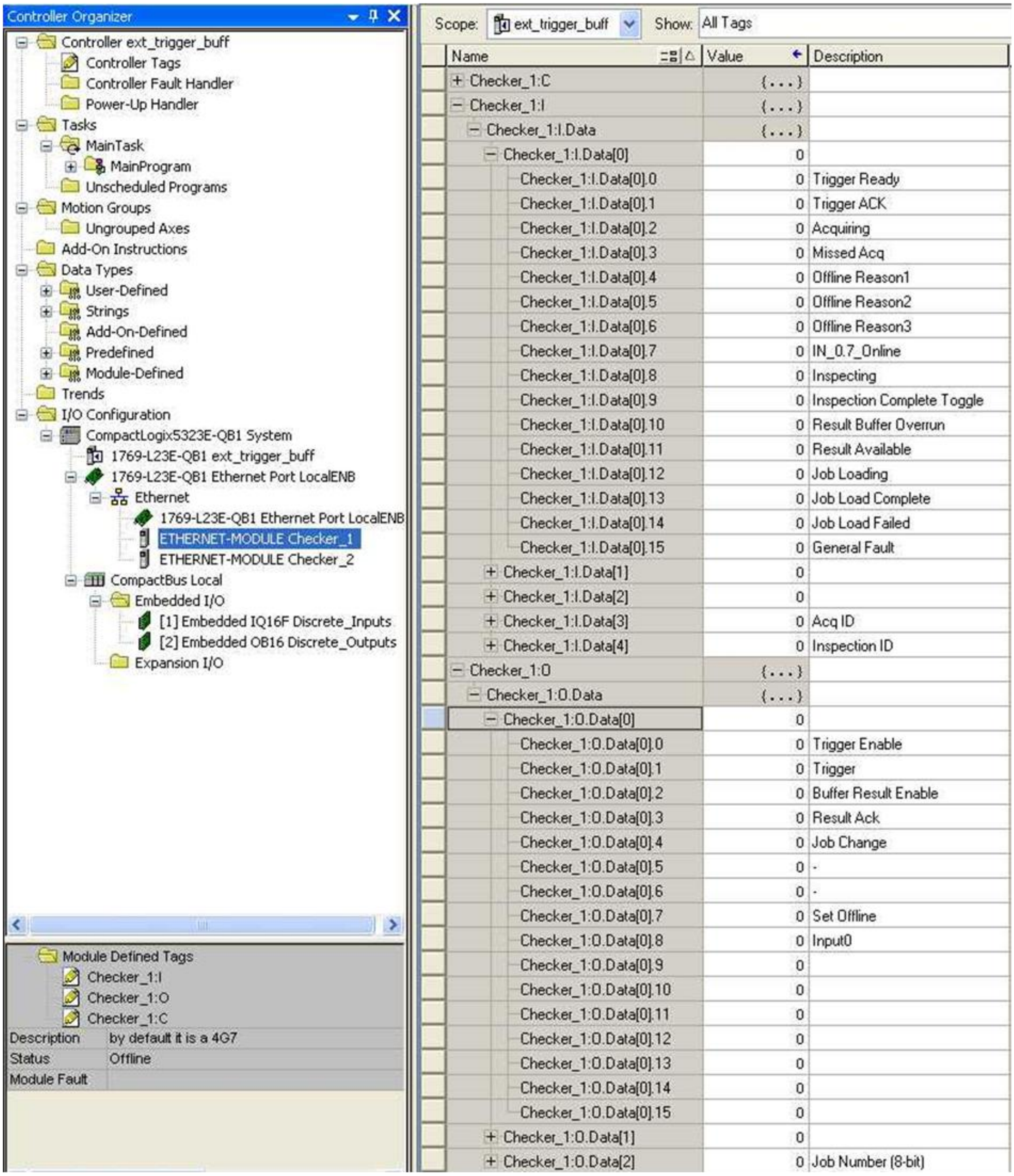


```
I/O Configuration
  CompactLogix5323E-QB1 System
    1769-L23E-QB1 ext_trigger_buff
    1769-L23E-QB1 Ethernet Port LocalENB
      Ethernet
        1769-L23E-QB1 Ethernet Port LocalENB
        ETHERNET-MODULE Checker_1
        ETHERNET-MODULE Checker_2
```

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.



Controller Organizer

- Controller ext_trigger_buff
  - Controller Tags
  - Controller Fault Handler
  - Power-Up Handler
- Tasks
  - MainTask
    - MainProgram
  - Unscheduled Programs
- Motion Groups
  - Ungrouped Axes
- Add-On Instructions
- Data Types
  - User-Defined
  - Strings
  - Add-On-Defined
  - Predefined
  - Module-Defined
- Trends
- I/O Configuration
  - CompactLogix5323E-QB1 System
    - 1769-L23E-QB1 ext_trigger_buff
    - 1769-L23E-QB1 Ethernet Port LocalENB
      - Ethernet
        - 1769-L23E-QB1 Ethernet Port LocalENB
        - ETHERNET-MODULE Checker_1
        - ETHERNET-MODULE Checker_2
    - CompactBus Local
      - Embedded I/O
        - [1] Embedded IQ16F Discrete_Inputs
        - [2] Embedded OB16 Discrete_Outputs
      - Expansion I/O

Module Defined Tags
- Checker_1:I
- Checker_1:O
- Checker_1:C

Description	by default it is a 4G7
Status	Offline
Module Fault	

Scope: ext_trigger_buff Show: All Tags

Name	Value	Description
+ Checker_1:C	{...}	
− Checker_1:I	{...}	
− Checker_1:I.Data	{...}	
− Checker_1:I.Data[0]	0	
Checker_1:I.Data[0].0	0	Trigger Ready
Checker_1:I.Data[0].1	0	Trigger ACK
Checker_1:I.Data[0].2	0	Acquiring
Checker_1:I.Data[0].3	0	Missed Acq
Checker_1:I.Data[0].4	0	Offline Reason1
Checker_1:I.Data[0].5	0	Offline Reason2
Checker_1:I.Data[0].6	0	Offline Reason3
Checker_1:I.Data[0].7	0	IN_0.7_Online
Checker_1:I.Data[0].8	0	Inspecting
Checker_1:I.Data[0].9	0	Inspection Complete Toggle
Checker_1:I.Data[0].10	0	Result Buffer Overrun
Checker_1:I.Data[0].11	0	Result Available
Checker_1:I.Data[0].12	0	Job Loading
Checker_1:I.Data[0].13	0	Job Load Complete
Checker_1:I.Data[0].14	0	Job Load Failed
Checker_1:I.Data[0].15	0	General Fault
+ Checker_1:I.Data[1]	0	
+ Checker_1:I.Data[2]	0	
+ Checker_1:I.Data[3]	0	Acq ID
+ Checker_1:I.Data[4]	0	Inspection ID
− Checker_1:O	{...}	
− Checker_1:O.Data	{...}	
− Checker_1:O.Data[0]	0	
Checker_1:O.Data[0].0	0	Trigger Enable
Checker_1:O.Data[0].1	0	Trigger
Checker_1:O.Data[0].2	0	Buffer Result Enable
Checker_1:O.Data[0].3	0	Result Ack
Checker_1:O.Data[0].4	0	Job Change
Checker_1:O.Data[0].5	0	-
Checker_1:O.Data[0].6	0	-
Checker_1:O.Data[0].7	0	Set Offline
Checker_1:O.Data[0].8	0	Input0
Checker_1:O.Data[0].9	0	
Checker_1:O.Data[0].10	0	
Checker_1:O.Data[0].11	0	
Checker_1:O.Data[0].12	0	
Checker_1:O.Data[0].13	0	
Checker_1:O.Data[0].14	0	
Checker_1:O.Data[0].15	0	
+ Checker_1:O.Data[1]	0	
+ Checker_1:O.Data[2]	0	Job Number (8-bit)

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.

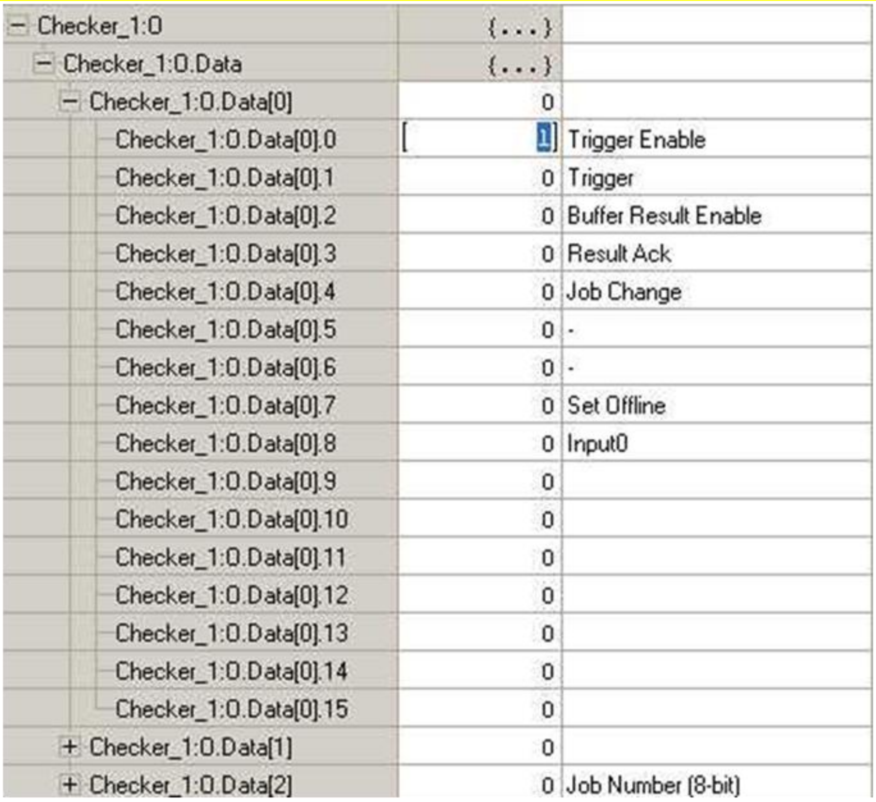
To access individual data items within an Assembly Object you must manually select the correct tag offset and data subtype (if necessary) within the tag array that the Generic profile provided. This can be awkward and error prone since it requires you to manually reference the vendor documentation which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

A deliberate effort has been made to make the Checker PROFINET communication model closely match the Cognex In-Sight family. Customers with In-Sight experience should find working with Checker familiar and comfortable.

Enabling PROFINET involves the following main steps:

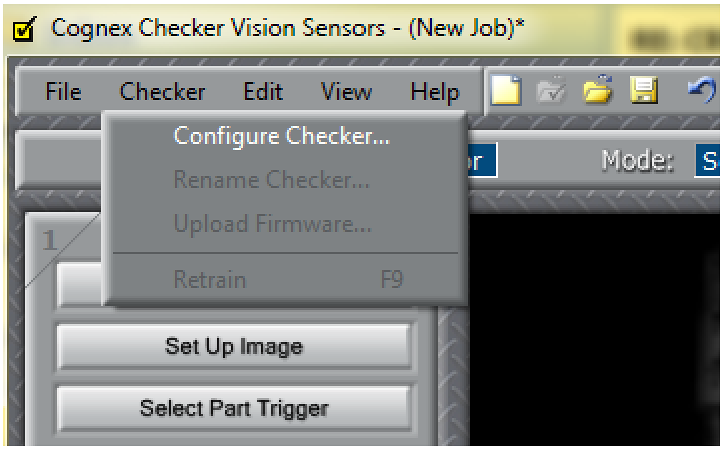
- Make sure you have the Siemens Step 7 programming software (SIMATIC) installed.
- Set up the Siemens Software tool so that it recognizes your Checker device.

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

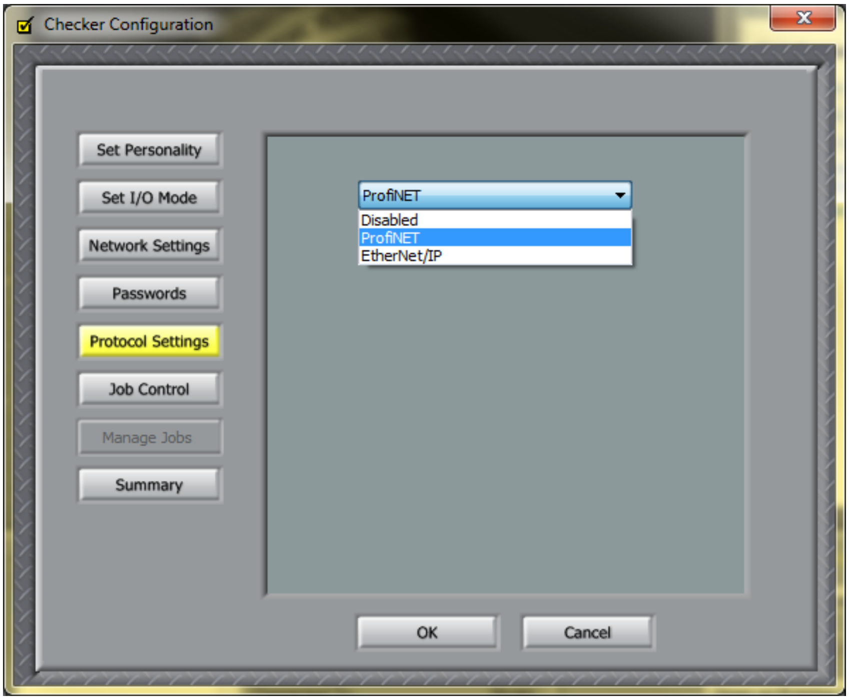
Enabling PROFINET in the Checker GUI

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.

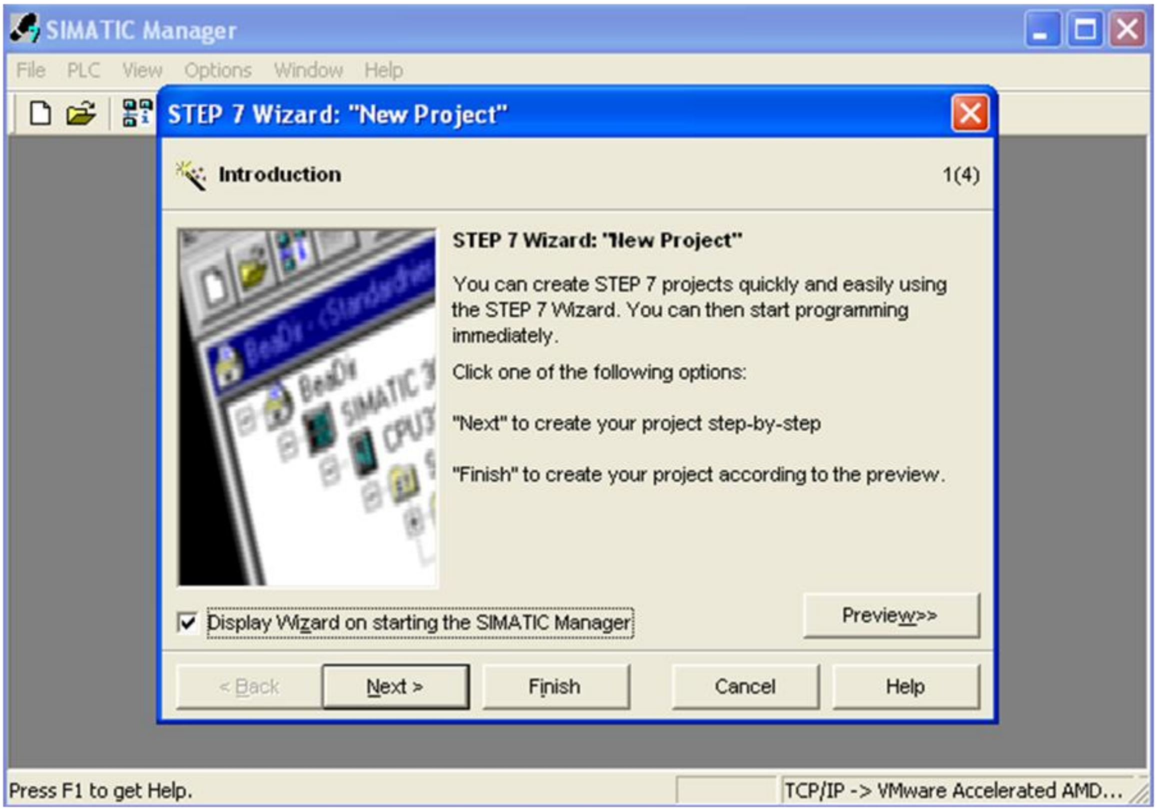
For the changes to take effect, the sensor automatically reboots.

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.



SIMATIC Manager

File PLC View Options Window Help

STEP 7 Wizard: "New Project"

Introduction 1(4)

STEP 7 Wizard: "New Project"

You can create STEP 7 projects quickly and easily using the STEP 7 Wizard. You can then start programming immediately.

Click one of the following options:

"Next" to create your project step-by-step

"Finish" to create your project according to the preview.

☑ Display Wizard on starting the SIMATIC Manager

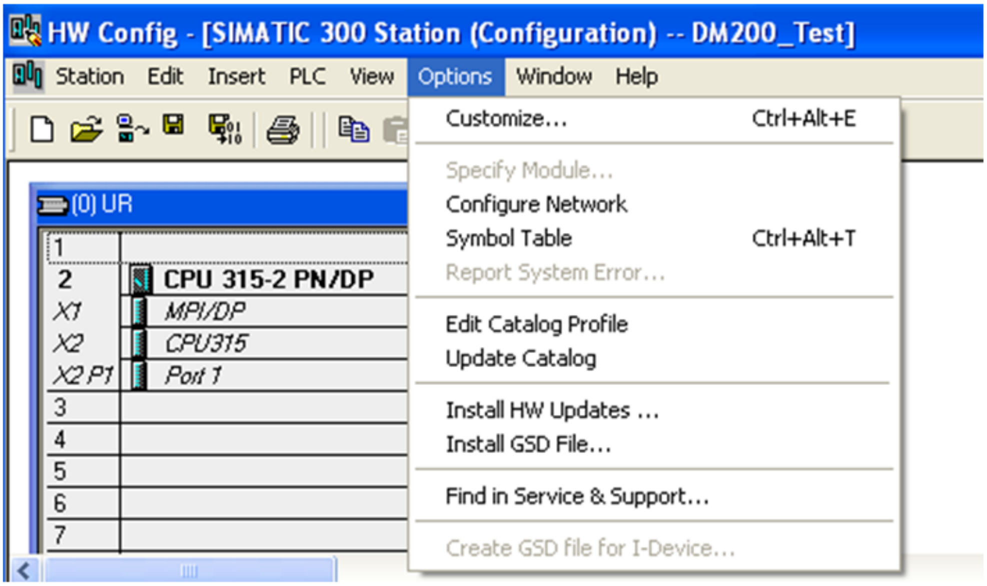
Preview>>

< Back | Next > | Finish | Cancel | Help

Press F1 to get Help. TCP/IP -> VMware Accelerated AMD...

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…**.



HW Config - [SIMATIC 300 Station (Configuration) -- DM200_Test]

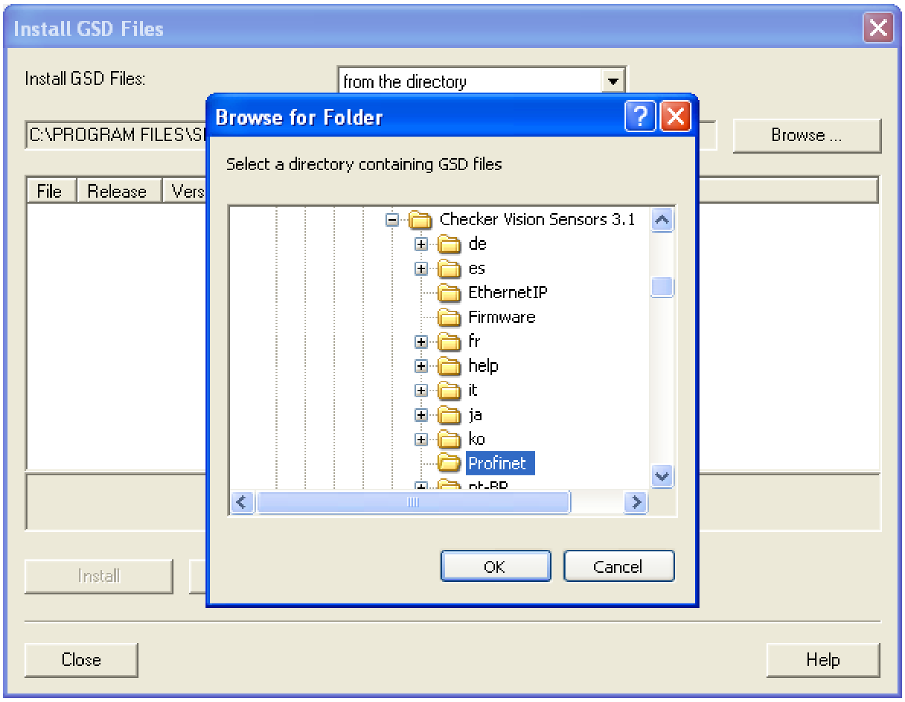
Station Edit Insert PLC View Options Window Help

Options menu	
Customize...	Ctrl+Alt+E
Specify Module...	
Configure Network	
Symbol Table	Ctrl+Alt+T
Report System Error...	
Edit Catalog Profile	
Update Catalog	
Install HW Updates ...	
Install GSD File...	
Find in Service & Support...	
Create GSD file for I-Device...	

(0) UR

1	
2	CPU 315-2 PN/DP
X1	MPI/DP
X2	CPU315
X2 P1	Port 1
3	
4	
5	
6	
7	

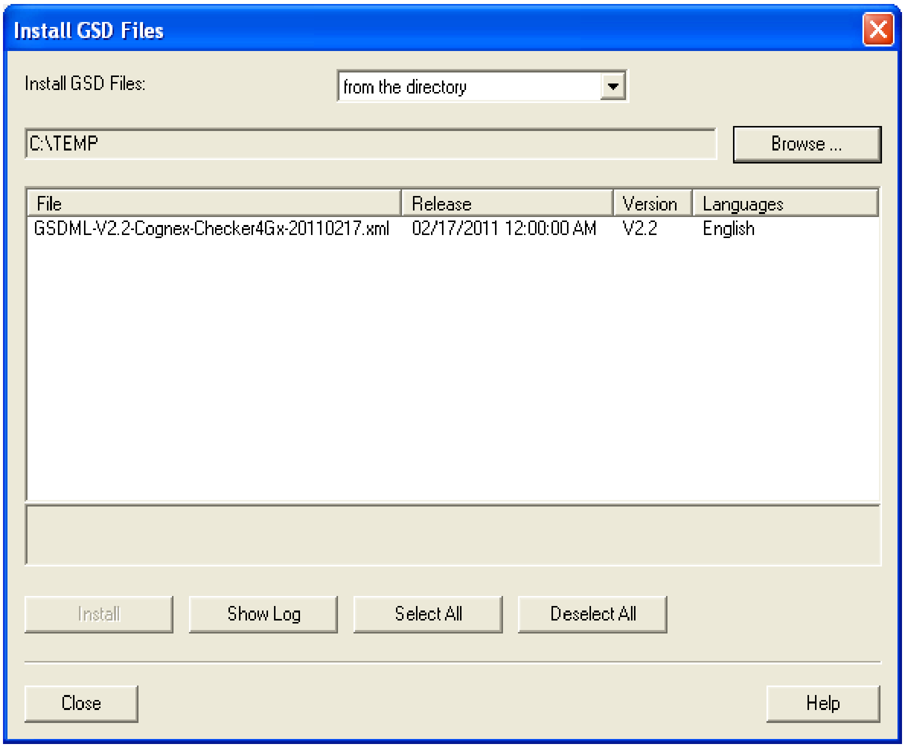
5. Browse to the location where the GSD file was installed (or the location where you saved the GSD file if it was downloaded from the web).



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.



Install GSD Files: from the directory

C:\TEMP

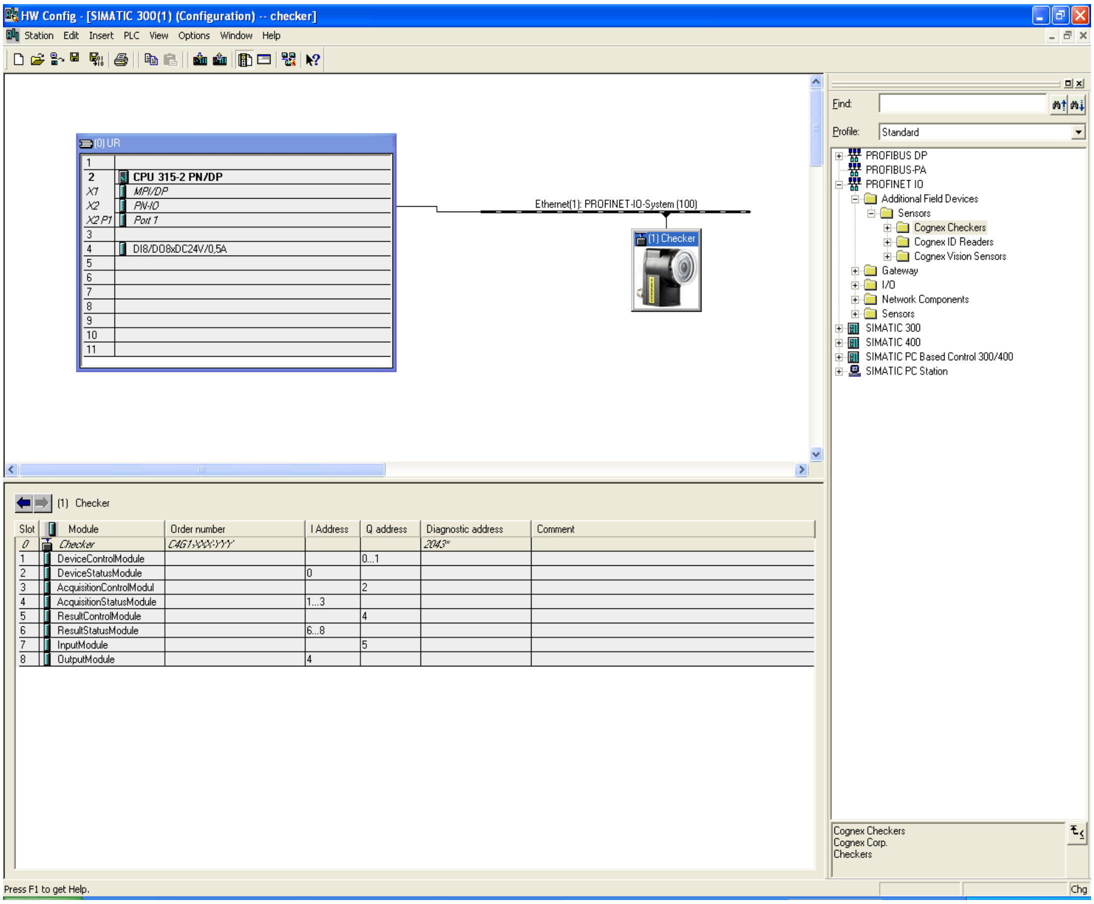
File	Release	Version	Languages
GSDML-V2.2-Cognex-Checker4Gx-20110217.xml	02/17/2011 12:00:00 AM	V2.2	English

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 “Cognex Checkers” 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.



(0) UR

1	
2	**CPU 315-2 PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2 P1	*Port 1*
3	
4	DI8/DO8xDC24V/0,5A
5	
6	
7	
8	
9	
10	
11	

(1) Checker

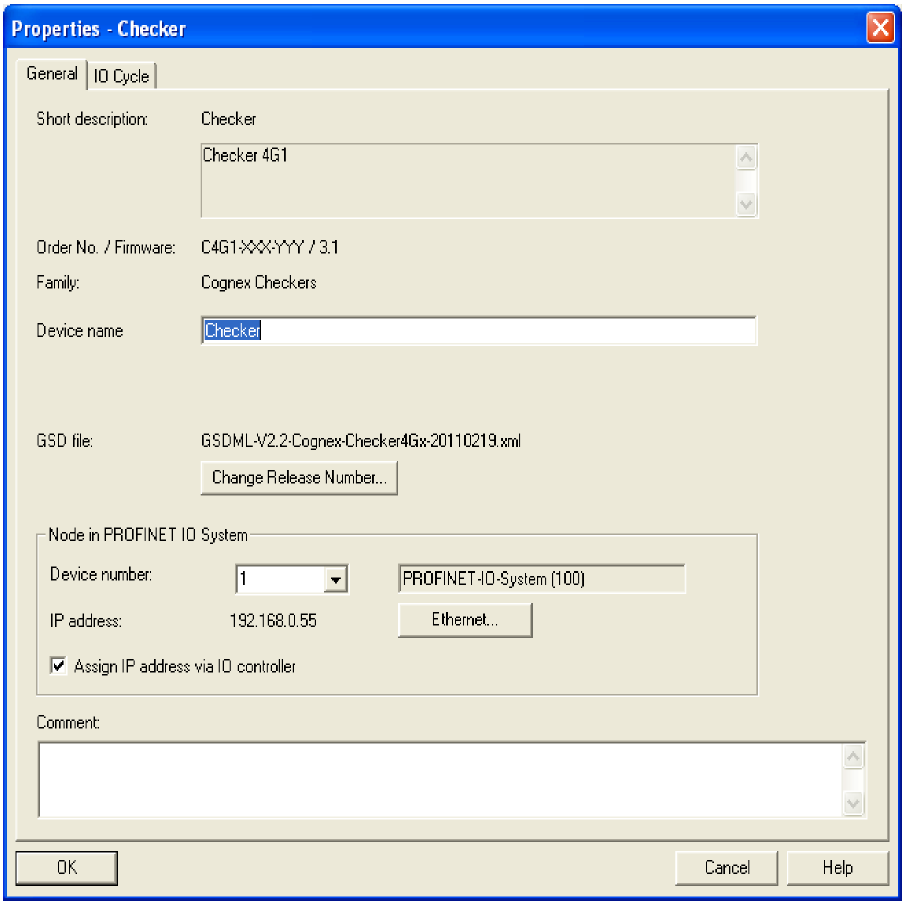
Slot	Module	Order number	I Address	Q address	Diagnostic address	Comment
0	*Checker*	*C4G1-XXX-YYY*			*2043**	
1	DeviceControlModule			0...1		
2	DeviceStatusModule		0			
3	AcquisitionControlModul			2		
4	AcquisitionStatusModule		1...3			
5	ResultControlModule			4		
6	ResultStatusModule		6...8			
7	InputModule			5		
8	OutputModule		4			

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.


Properties - Checker

General | IO Cycle

Short description: Checker

Checker 4G1

Order No. / Firmware: C4G1-XXX-YYY / 3.1

Family: Cognex Checkers

Device name Checker

GSD file: GSDML-V2.2-Cognex-Checker4Gx-20110219.xml

Node in PROFINET IO System

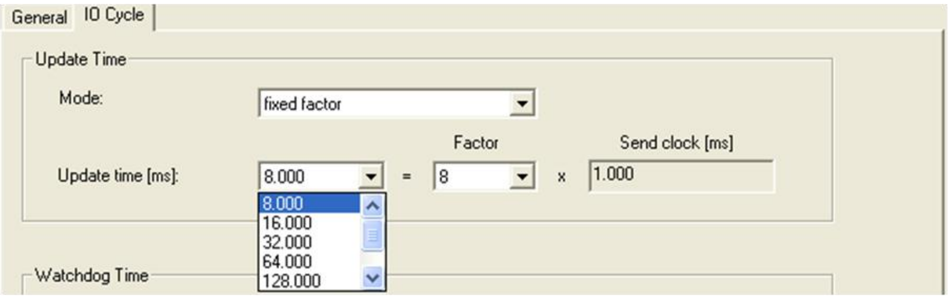
Device number: 1 PROFINET-IO-System (100)

IP address: 192.168.0.55

☑ Assign IP address via IO controller

Comment:

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


General | IO Cycle

Update Time

Mode: fixed factor

Factor: 8

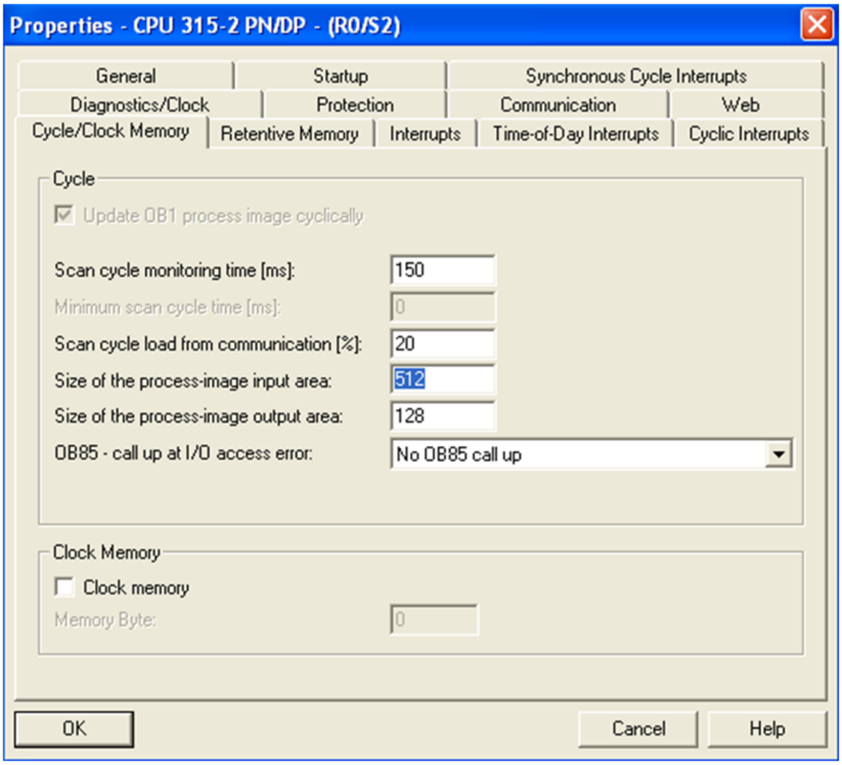
Send clock [ms]: 1.000

Update time [ms]: 8.000 = 8 x 1.000

Watchdog Time

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.



☑ Update OB1 process image cyclically

Scan cycle monitoring time [ms]: 150

Minimum scan cycle time [ms]: 0

Scan cycle load from communication [%]: 20

Size of the process-image input area: 512

Size of the process-image output area: 128

OB85 - call up at I/O access error: No OB85 call up

□ Clock memory

Memory Byte: 0

NOTE

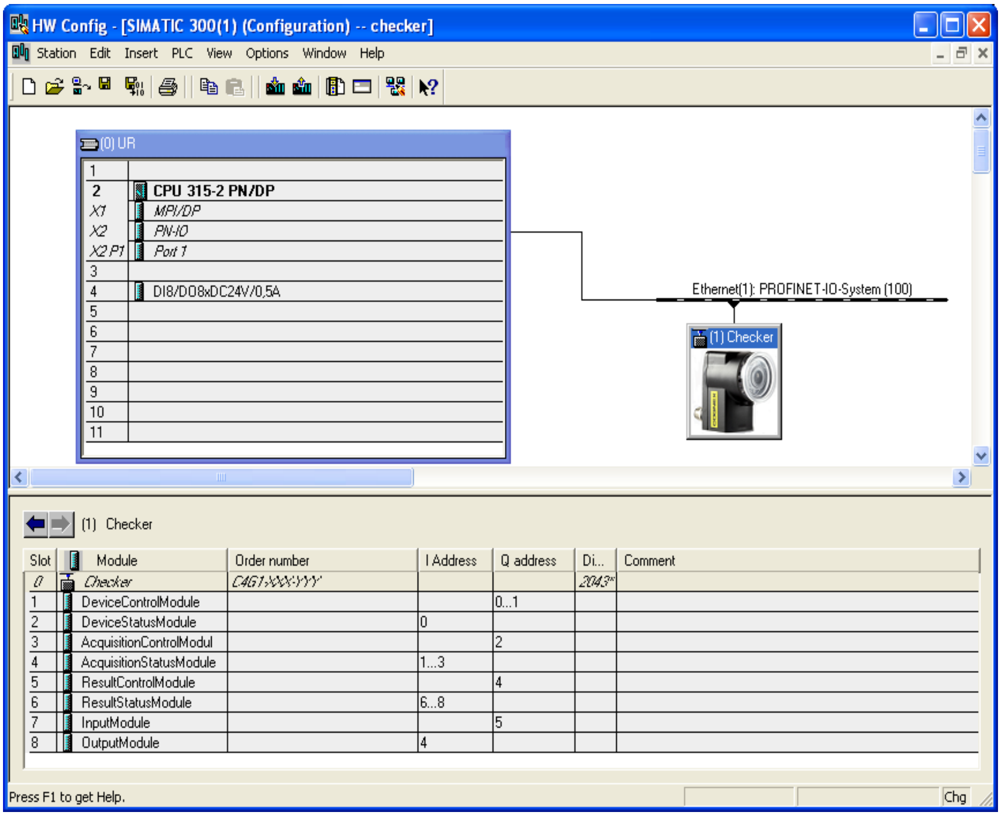
Expanding the process image can have a performance impact on the PLC scan cycle time. If your scan time is critical, use the minimal acceptable module sizes and manually remap them down lower in the process image.

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


HW Config - [SIMATIC 300(1) (Configuration) -- checker]
Station Edit Insert PLC View Options Window Help
(0) UR
Ethernet(1): PROFINET-IO-System (100)
(1) Checker
Press F1 to get Help.
Chg

1		
2	CPU 315-2 PN/DP	
X1	MPI/DP	
X2	PN-IO	
X2 P1	Port 1	
3		
4	DI8/DO8xDC24V/0,5A	
5		
6		
7		
8		
9		
10		
11		

(1) Checker

Slot	Module	Order number	I Address	Q address	Di...	Comment
0	Checker	C4G1-XXX-YYY			2043*	
1	DeviceControlModule			0...1		
2	DeviceStatusModule		0			
3	AcquisitionControlModul			2		
4	AcquisitionStatusModule		1...3			
5	ResultControlModule			4		
6	ResultStatusModule		6...8			
7	InputModule			5		
8	OutputModule		4			

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 suspends all 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 is filled in.
2 – 7	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: 1 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.

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.

Bit	Name	Description
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
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 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 tread 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 byte

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.

Bit	Name	Description
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 Any 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: 1 byte

Bit	Name	Description
0	Input 0	This bit is used as Input in the Ladder Logic of the Checker GUI.

Output Module

Slot number: 8

Total Module size: 1 byte

Bit	Name	Description
0 - 7	Output 0-7	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.

Operation

Acquisition Sequence

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

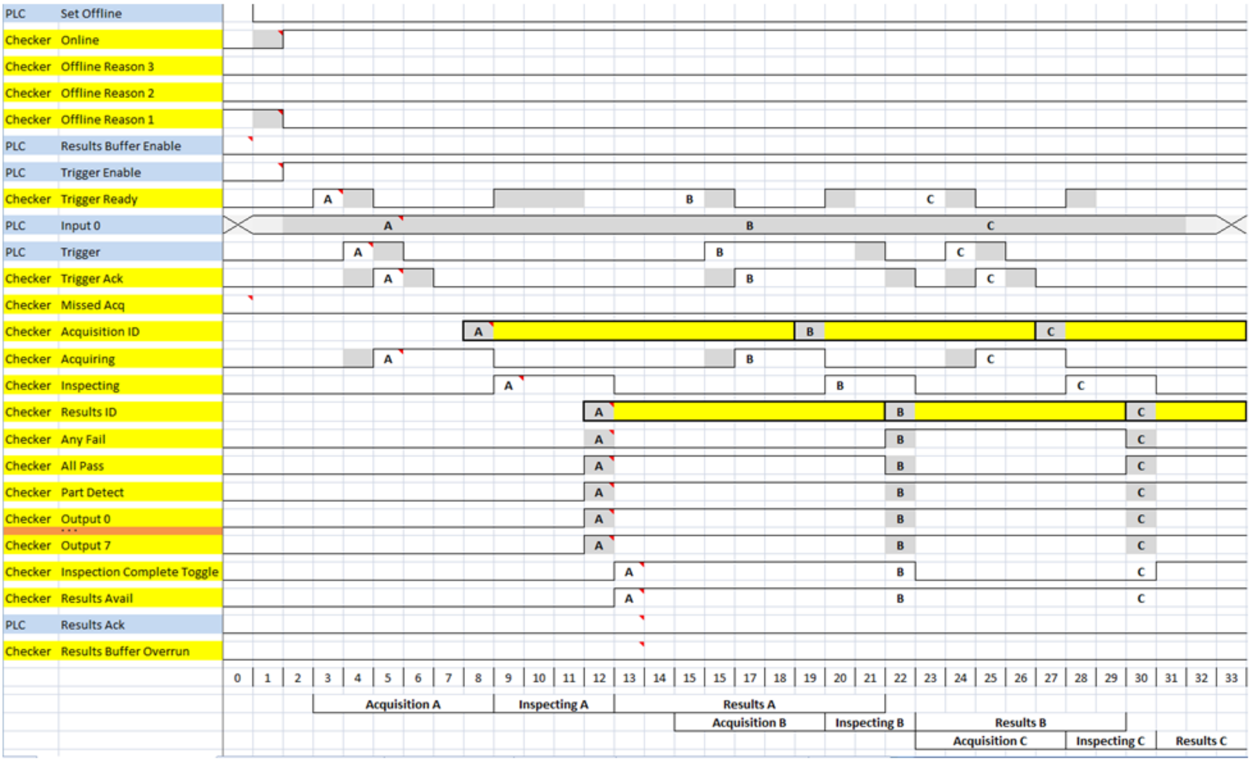
On startup the TriggerEnable attribute will be false. It must be set to true to enable triggering. When the device is ready to accept triggers, the Trigger Ready bit in the AcqStatusRegister will be set to true.

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.
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.

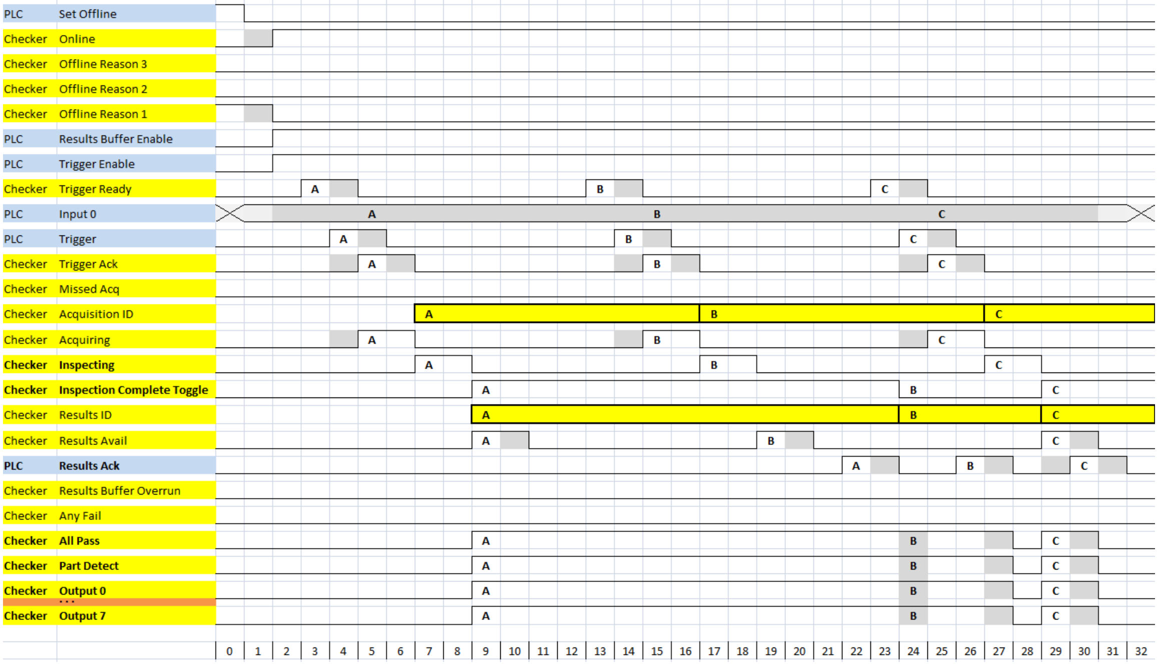
Results Buffering

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 is 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	

Trigger and Get Results

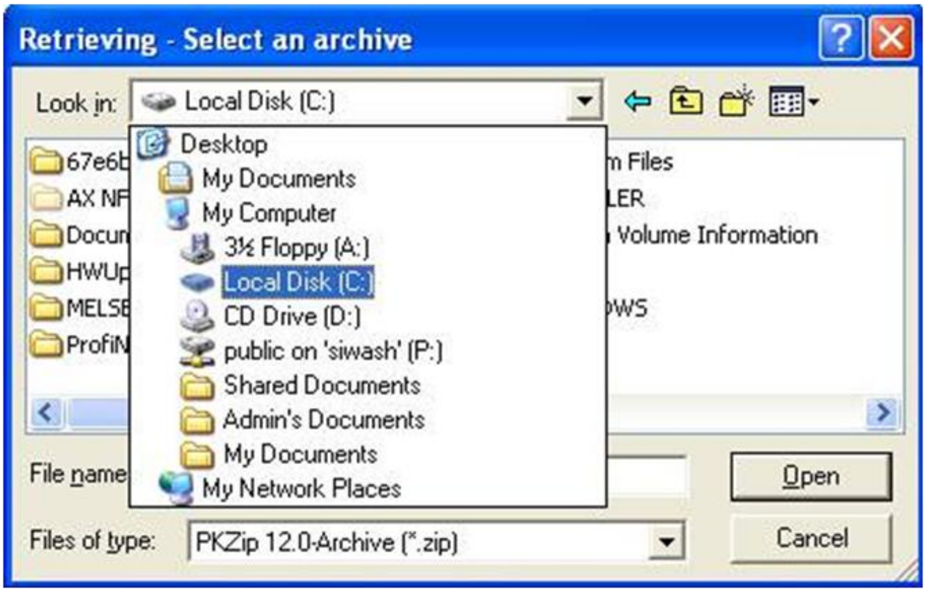
Run the sample program “Chkr_E_2” for the complete example program. Perform the following steps to install the program:

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.

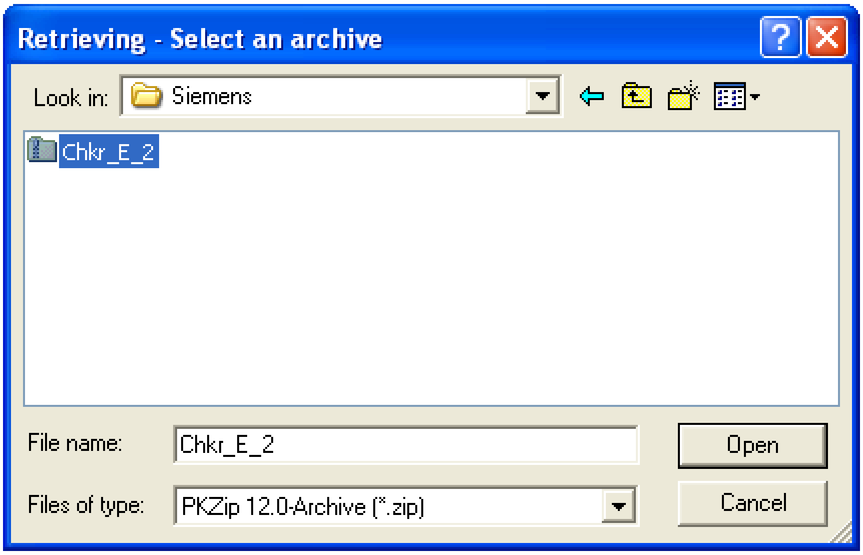


Retrieving - Select an archive

Look in: Local Disk (C:)

Files of type: PKZip 12.0-Archive (*.zip)

5. Look for the Siemens folder and select the zip file.



Retrieving - Select an archive

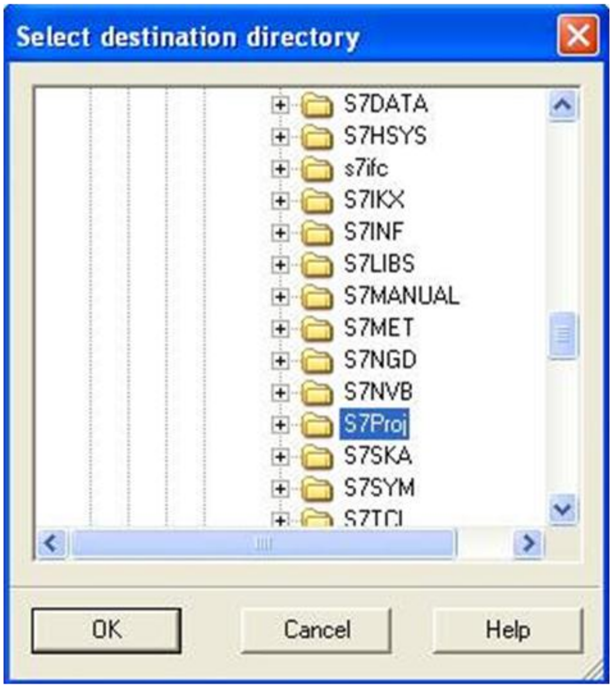
Look in: Siemens

Chkr_E_2

File name: Chkr_E_2

Files of type: PKZip 12.0-Archive (*.zip)

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



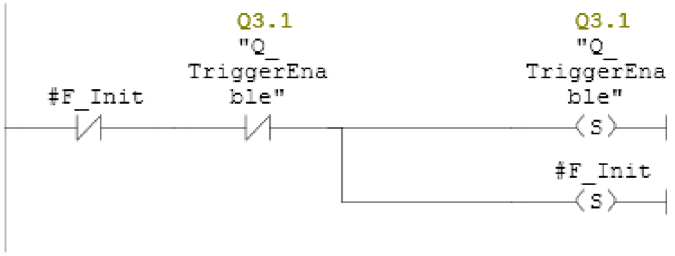
Select destination directory

S7DATA
S7HSYS
s7ifc
S7IKX
S7INF
S7LIBS
S7MANUAL
S7MET
S7NGD
S7NVB
S7Proj
S7SKA
S7SYM
S7TCI

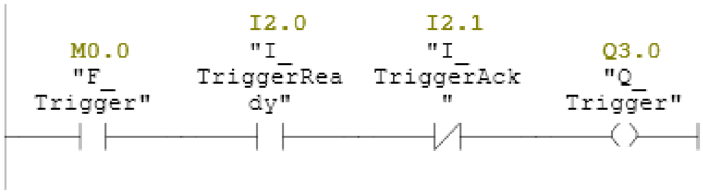
The Siemens software extracts the sample archive and makes it available.

Reduced to the basics the process of external triggering consists of the following:

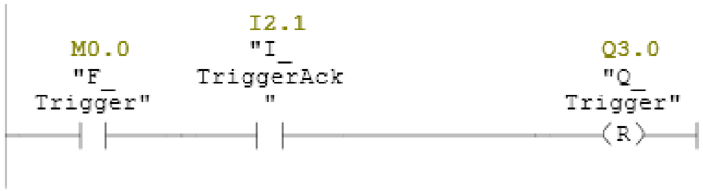
1. Enable the external trigger on the sensor:



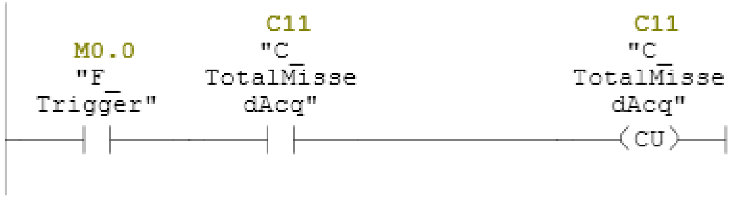
2. Set the trigger signal to start the acquisition.



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



4. Check for any Missed Acquisition (you must not detect any).


M0.0
"F_
Trigger"
C11
"C_
TotalMisse
dAcq"
C11
"C_
TotalMisse
dAcq"
CU