COGNEX

Advanced machine vision made easy

DataMan® Industrial Protocols Manual

2026 July 30

Revision: 26.2.0.35

Legal Notices

The software described in this documentation is provided "as is" and "as available," without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall Cognex Corporation and its global affiliates (collectively, “Cognex”) be liable for any claims, damages, or other liability arising from, out of, or in connection with the software or your use of the software.

Your use of this software is governed by the full terms and conditions of the Cognex Software License Agreement. For complete licensing details and legal terms, please visit https://www.cognex.com/terms-and-conditions.

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

Companies, names, and data used in examples herein are fictitious unless otherwise noted. 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 U.S. and foreign patents, as well as pending U.S. and foreign patents listed on the Cognex web site at: cognex.com/patents.

In the event of any inconsistency between the English language version of this documentation and any other language version, the English language version shall take precedence.

Copyright © 2026. Cognex Corporation. All Rights Reserved.

The following are registered U.S. trademarks of Cognex Corporation:

2DMAX, 3D LIGHTBURST TECHNOLOGY, ADVANTAGE, ALIGNPLUS, ASSEMBLYPLUS, COGNEX, COGNEX VISION FOR INDUSTRY, CVL, DATAMAN, DVT, EASYBUILDER, EDGE INTELLIGENCE, HOTBARS, IDMAX, IN-SIGHT, LIGHTBURST, MX, OMNIVIEW, PATFIND, PATFLEX, PATINSPECT, PATMAX, PATMAX REDLINE, PATQUICK, POWERGRID, QUICKSORT, ULTRALIGHT, VISION SOLUTIONS, VISIONPRO, VISIONVIEW

The following are trademarks of the Cognex Corporation:

1DMAX, 3D SYMBOLIC LIGHT, 3D-LOCATE, 3DMAX, ONEVISION, PINPOINT, RANGEMAX, SLX, VIDI, OCRMAX, INSPECTEDGE, VISION SOLUTIONS

Portions copyright © Microsoft Corporation. All rights reserved.

Portions copyright © MadCap Software, Inc. All rights reserved.

Other product and company trademarks identified herein are the trademarks of their respective owners.

Table of Contents

Legal Notices 2
Table of Contents 3
Symbols 5
About This Manual 6
Industrial Network Protocols 7
EtherNet/IP 8
DMCC 8
Reader Configuration Code 8
DataMan Setup Tool 9
Preparing to Use Ethernet/IP 9
EtherNet/IP Object Model 13
EtherNet/IP Operation 20
Rockwell ControlLogix Examples 24
PROFINET 38
DMCC 38
Reader Configuration Code 38
Setup Tool 39
Preparing to Use PROFINET 39
PROFINET Object Model 43
PROFINET Operation 48
SIMATIC Examples 53
TIA Portal Examples 65
Removing Outdated GSD Files from TIA Portal 75
PROFINET Class B 83
iQ Sensor Solution 86
DMCC 86
Reader Configuration Code 86
Setup Tool 87
Overview 87
Discovering DataMan Readers on GX Work2 88
Monitoring the Status of the DataMan Reader 88
iQSS Error Codes 89
CC-Link IE Field Basic 90
DMCC 90
Reader Configuration Code 90
DataMan Setup Tool 90
CC-Link IE Field Basic Modules 91
CC-Link IE Field Basic Operation 95
SLMP Protocol 98
DMCC 98
Reader Configuration Code 98
SLMP Protocol Scanner 99
Preparing to Use SLMP 99
SLMP Network Configuration 100

SLMP Operation 107
SLMP Examples 113
ModbusTCP 116
DMCC 116
Reader Configuration Code 116
Setup Tool 117
Modbus TCP Handler 117
Preparing to Use Modbus TCP 117
Modbus TCP Interface Model 120
Modbus TCP Operation 125
Modbus TCP Examples 131
EtherCAT 134
File Over EtherCAT (FoE) Function 134
DMCC 134
Reader Configuration Code 135
Setup Tool 135
Preparing to Use EtherCAT 136
EtherCAT Object Model 136
EtherCAT Operation 141
Industrial Protocols for the Wireless DataMan 148
Protocol Operation 148
Ethernet Address 148
PLC Triggering 148
SoftEvents 148
DMCC 148
Offline Buffering 149
Status of Industrial Protocols 149

Symbols

The following symbols indicate safety precautions and supplemental information:

WARNING: This symbol indicates a hazard that could cause death, serious personal injury or electrical shock.

CAUTION: This symbol indicates a hazard that could result in property damage.

Note: This symbol indicates additional information about a subject.

Tip: This symbol indicates suggestions and shortcuts that might not otherwise be apparent.

About This Manual

The DataMan reader acquires images by several methods when connected to a network:

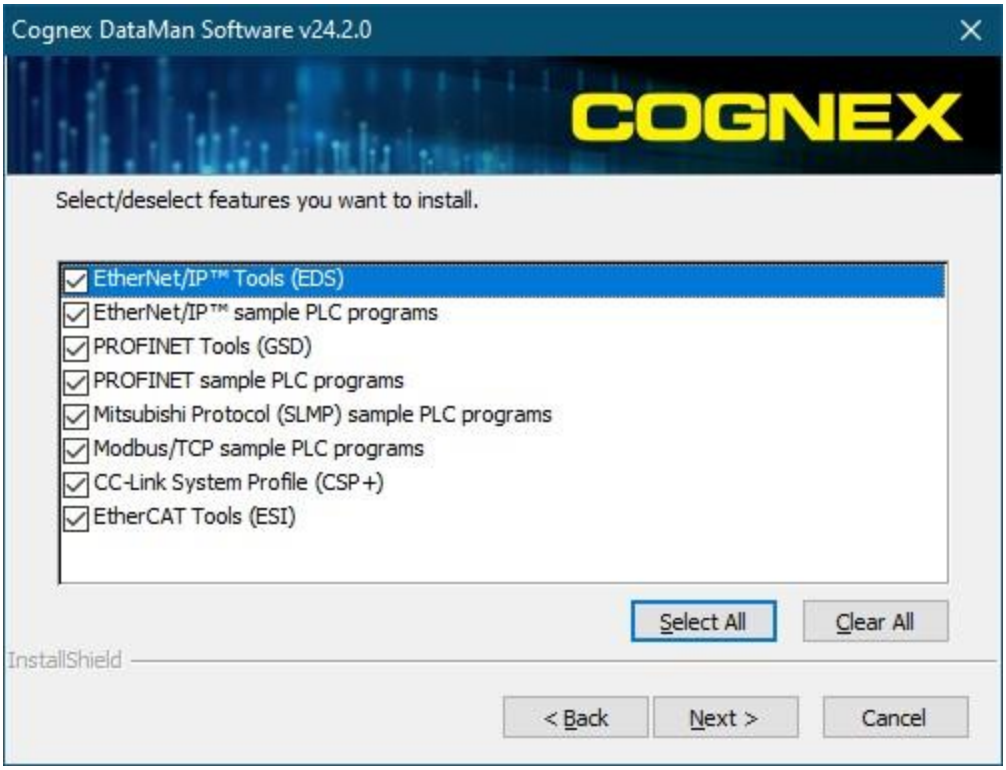
- using the DataMan Setup Tool
- trigger bits
- through a DMCC command
- manipulating objects through industrial protocols

The *DataMan Industrial Protocols Manual* provides information on how to integrate the DataMan reader into your particular environment using industrial protocols.

Industrial Network Protocols

DataMan uses industrial network protocols that are based on standard Ethernet protocols. These protocols, EtherNet/IP, PROFINET, SLMP Protocol, and Modbus/TCP, are enhanced to provide more reliability than standard Ethernet.

When you install the DataMan Setup Tool, the Installer prompts you to download the tools and sample programs associated with industrial protocols.



EtherNet/IP

DataMan 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 transferring 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 called Assemblies.

By default the DataMan reader has the EtherNet/IP protocol disabled. The protocol can be enabled via DMCC, scanning a configuration code, or in the DataMan Setup Tool.

Note: If you have a wireless DataMan reader, see section *Industrial Protocols for the Wireless DataMan* on page 148

DMCC

The following commands can be used to enable or disable EtherNet/IP on a reader. The commands can be issued via RS-232 or Telnet connection.

Note: Use a third party Telnet client such as PuTTY to communicate with your DataMan reader.

Enable:

```
||>SET ETHERNET-IP.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET ETHERNET-IP.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

Reader Configuration Code

Scan the following reader configuration codes to enable/disable EtherNet/IP on your corded reader.

Note: You must reboot the reader for the change to take effect.

Enable:

Disable:

Scan the following reader configuration codes to enable/disable EtherNet/IP on your DataMan 8000 Base Station.

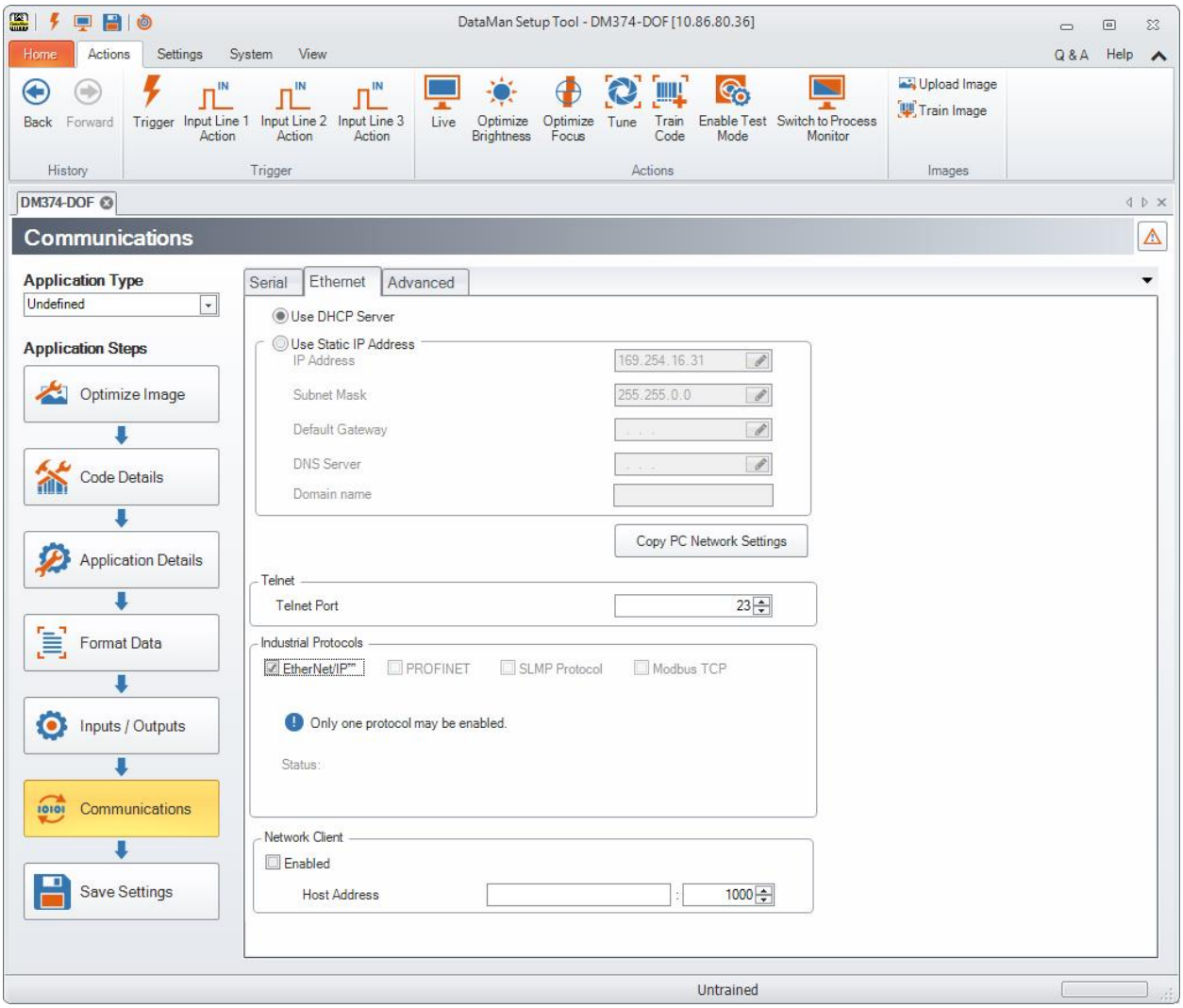
Note: You must reboot the reader for the change to take effect.

Enable:

Disable:

DataMan Setup Tool

In the **Ethernet** tab of the **Communications** application step in DataMan Setup Tool, check **EtherNet/IP** to enable this industrial protocol.



Make sure to save the new selection by clicking the **Save Settings** button in the upper toolbar before disconnecting from the reader.

ⓘ **Note**: You must reboot the reader for the change to take effect.

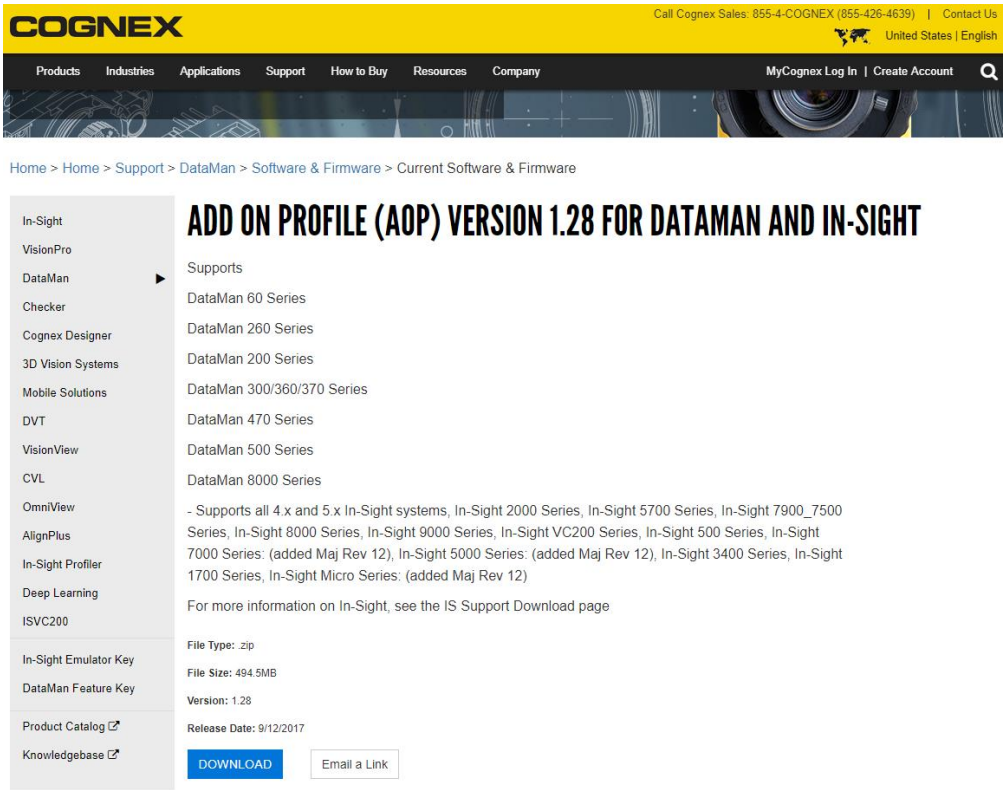
Preparing to Use Ethernet/IP

Perform these steps to start using EtherNet/IP:

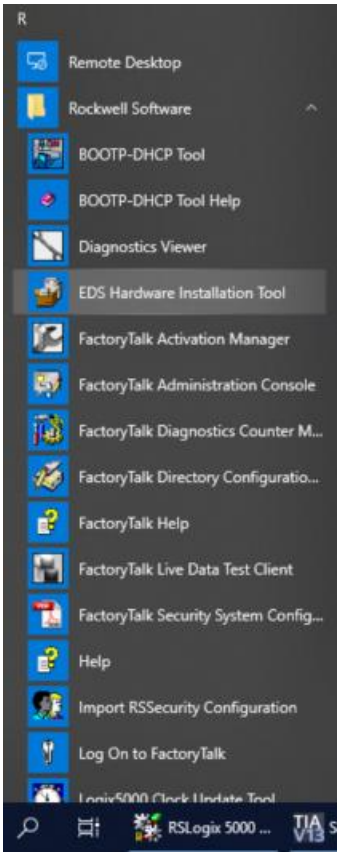
- Install the Rockwell Software tool.
- Set up the Rockwell Software tool so that it recognizes your DataMan reader.
- Install the DataMan Electronic Data Sheet (EDS) for the DataMan reader.

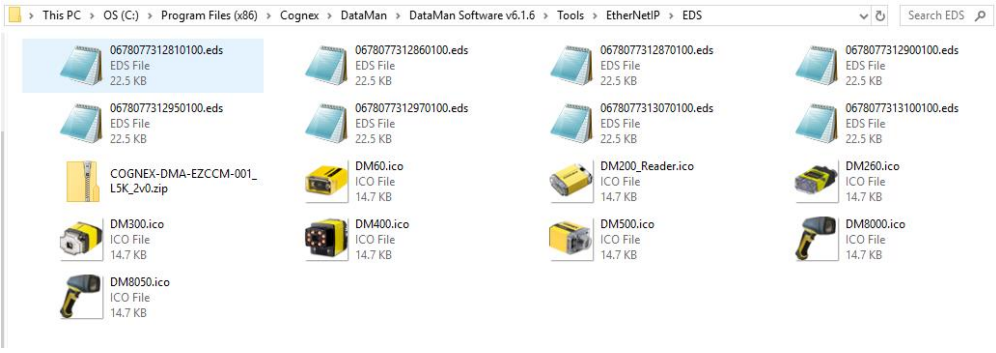
Perform the following steps to set up EtherNet/IP:

1. Make sure that you select the **Add-on Profile** (AOP) installation and the **Samples** installation. The Add-on Profile is only used with Rockwell ControlLogix or CompactLogix PLCs.
2. Install the Rockwell Add-on Profiles by navigating to the following directory on Cognex.com: https://support.cognex.com/en/downloads/dataman/software-firmware.
3. In the search box, type **Add-on Profile**. Click the file and download it from the following page:



4. From the Start menu, go to Programs -> Rockwell Software -> EDS Hardware Installation Tool.



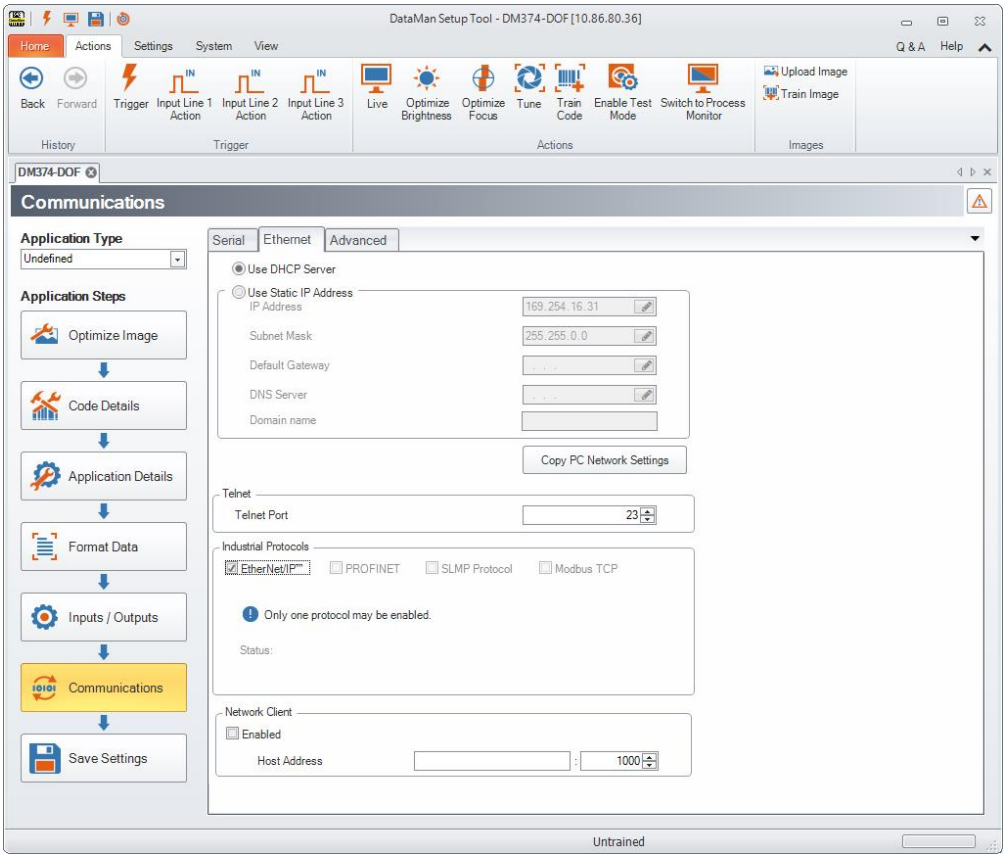


5. Run the EDS Installation Tool.

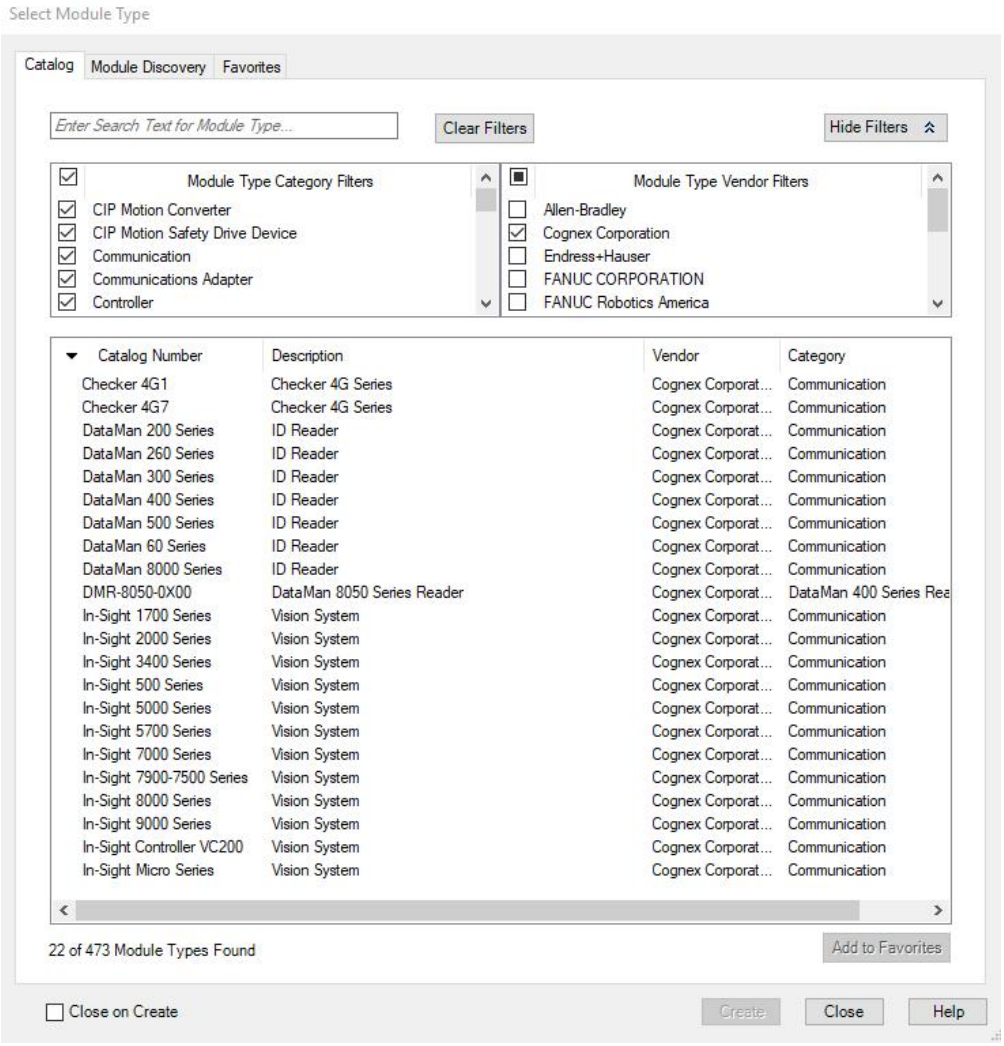
> **Note**: If you have an existing EDS file, uninstall it first, then install the latest version of the EDS.

6. Run the DataMan Setup Tool and update the DataMan firmware.
7. Check if the DataMan firmware is up-to-date by clicking, in the DataMan Setup Tool, **View** and then **System Info**.

8. Make sure that in the DataMan Setup Tool, **EtherNet/IP** is enabled.



9. In order for the changes to take effect, save your settings and cycle power. Go to **System** and click **Save Settings**.
10. Reboot your reader.
11. Your reader is visible now in the RSLinx.



Note: If your reader is visible, but the icon is a question mark, repeat the EDS Installation.

12. Open one of the sample jobs and integrate your reader into your program using the **Add-on Profile**.

EtherNet/IP Object Model

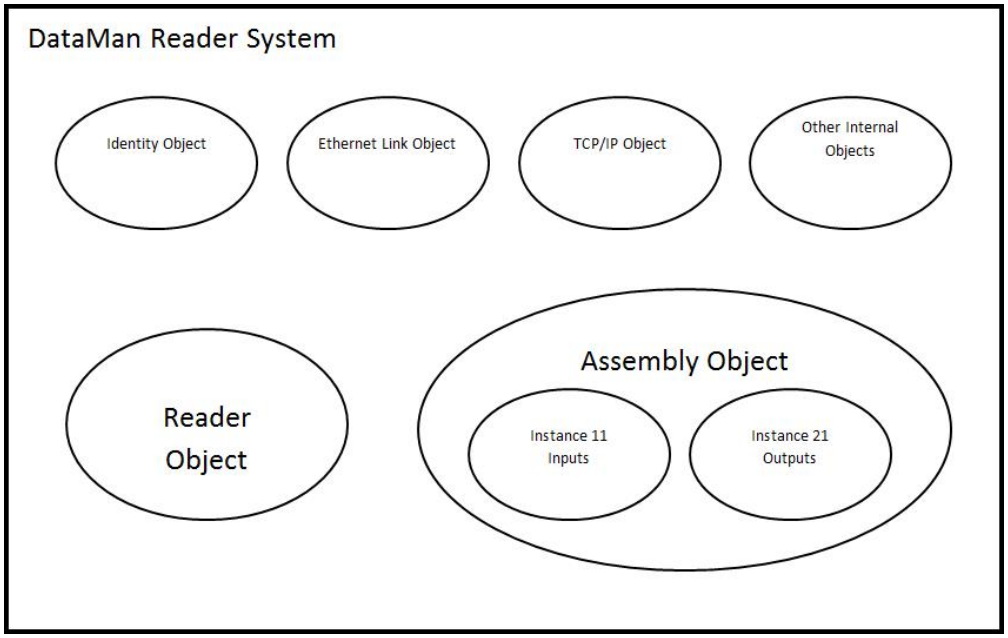
The ID Reader Object is a vendor specific object class. This means that it is not part of the CIP common (public) architecture but an extension. It is a custom object that Cognex has added to the EtherNet/IP architecture on the DataMan reader. All the data and functionality of this object model are available in the reader. This includes triggering, status, events, errors, and result data.

The ID Reader Object is identified by its vendor specific class code.

Reader ID Reader Object Class Code: 0x79

Objects are made up of attributes (data) and services (functionality). These can be defined at the class level (common to all instances of the class) or the instance level (unique to an individual instance). The CIP specification defines common attributes and services that apply to all objects (often these are optional). Vendors may also define their own attributes and services for their vendor specific classes.

The ID Reader Object attributes and services can be individually accessed via explicit messaging. In addition, a number of the ID Reader Object attributes are exposed in the DataMan assembly objects which allow them to be accessed as a group via implicit messaging.



Attributes

The DataMan ID Reader Object (Class Code: 0x79) has the following attributes:

Attribute ID	Access Rule	Name	Data Type	Description
0x9	Set	AcqTriggerEnable	BOOL	0 = EtherNet/IP triggering is disabled 1 = EtherNet/IP triggering is enabled
0xA	Set	AcqTrigger	BOOL	Acquire an image when this attribute changes from 0 to 1.
0xB	Get	AcqStatusRegister	BYTE	Bit0: Trigger Ready Bit1: Trigger Ack Bit2: Acquiring Bit3: Missed Acquisition Bit4-7: Reserved
0xC	Set	UserData	ARRAY of BYTE	User defined data that can be used as an input to the acquisition/decode.
0xD	Set	BufferResultsEnable	BOOL	When true, it enables buffering of the decode results. The reader can buffer up to 50 and the Base Station can buffer up to 500 sets of read results.
0xE	Get	DecodeStatusRegister	BYTE	Bit0: Decoding Bit1: Decode completed (toggle) Bit2: Results buffer overrun Bit3: Results available Bit4: Reserved Bit5: Reserved Bit6: Reserved Bit7: General fault indicator

Attribute ID	Access Rule	Name	Data Type	Description
0xF	Set	ResultsAck	BOOL	Acknowledges that the client received the decode results.
0x10	Get	DecodeResults	STRUCT of	The last decode results.
		ResultsID	UINT	Decode results identifier. Corresponds to the TriggerID of the decoded image.
		ResultCode	UINT	Decode result summary code value. Bit0: 1=Read, 0=No read Bit1: 1=Validated, 0=Not Validated Bit2: 1=Verified, 0=Not Verified Bit3: 1=Acquisition trigger overrun Bit4: 1=Acquisition buffer overrun Bit 5: Reserved Bit 6: 1=result is a DMCC response, 0=result is a read result. For more information, see *DMCC Result Feedback from PLC* on page 23. Bit 7-15: Reserved (future use)
		ResultExtended	UINT	Extended result information.
		ResultLength	UINT	Current number of result data bytes.
		ResultData	ARRAY of BYTE	Result data from last decode.
0x12	Set	SoftEvents	BYTE	SoftEvents act as virtual inputs (execute action on 0 to 1 transition) Bit0: Train code Bit1: Train match string Bit2: Train focus Bit3: Train brightness Bit4: Un-Train Bit5: Reserved (future use) Bit6: Execute DMCC command Bit7: Set match string
0x15	Get	TriggerID	UINT	Trigger identifier. ID of the next trigger to be issued.

<table>
<tr><th>Attribute ID</th><th>Access Rule</th><th>Name</th><th>Data Type</th><th>Description</th></tr>
<tr><td>0x16</td><td>Set</td><td>UserDataOption</td><td>UINT</td><td>Optional user data information (16-bit Integer). Used by the “Set Match String” SoftEvent to assign the match string to the specified symbology. Depending on your PLC software, you can move an integer value or a binary to this field:
<table>
<tr><th>Integer</th><th>Binary</th><th>Description</th></tr>
<tr><td>0</td><td>0000 0000 0000 0000</td><td>Assign string to all smybologies.</td></tr>
<tr><td>1</td><td>0000 0000 0000 0001</td><td>Assign string to 2D codes.</td></tr>
<tr><td>2</td><td>0000 0000 0000 0010</td><td>Assign string to QR codes.</td></tr>
<tr><td>3</td><td>0000 0000 0000 0011</td><td>Assign string to 1D/stacked/postal codes.</td></tr>
</table></td></tr>
<tr><td>0x17</td><td>Set</td><td>UserDataLength</td><td>UINT</td><td>Current number of user data bytes.</td></tr>
<tr><td>0x18</td><td>Get</td><td>SoftEventAck</td><td>BYTE</td><td>Acknowledgment of SoftEvents.
Bit0: Train code ack
Bit1: Train match string ack
Bit2: Train focus ack
Bit3: Train brightness ack
Bit4: Un-Train ack
Bit5: Reserved (future use)
Bit6: Execute DMCC command ack
Bit7: Set match string ack</td></tr>
</table>

SoftEvents

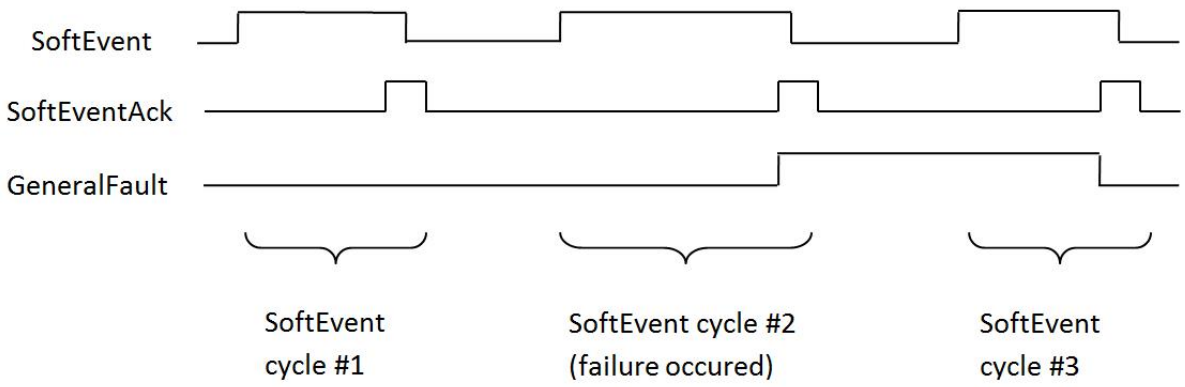
SoftEvents act as “virtual” inputs. When the value of a SoftEvent changes from 0 to 1, the action associated with the event is executed. When it is done, the corresponding SoftEventAck bit changes from 1 to 0 to mark completion.

The SoftEvent and SoftEventAck form a logical handshake. After SoftEventAck changes to 1 the original SoftEvent is set back to 0. When that occurs, SoftEventAck is automatically set back to 0.

The “ExecuteDMCC” and “SetMatchString” SoftEvent actions require user supplied data. This data must be written to the UserData and UserDataLength area of the Input Assembly prior to invoking the SoftEvent. Since both of these SoftEvents depend on the UserData, only one can be invoked at a time.

General Fault Indicator

When a communication related fault occurs, the “GeneralFault” bit changes from 0 to 1. The only fault conditions supported are SoftEvent operations. If a SoftEvent operation fails, the fault bit will be set. The fault bit remains set until either the next successful SoftEvent operation or until TriggerEnable is set to 0 and then back to 1.



Services

The ID Reader Object supports the following Common CIP services:

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

The ID Reader Object supports the following vendor specific services:

Service Code	Service Name	Description
0x32	Acquire	Triggers a single acquisition.
0x34	SendDMCC	Sends a DMCC command to the reader.
0x35	GetDecodeResults	Gets the content of the DecodeResults attribute.

Acquire Service

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

SendDMCC Service

The SendDMCC Service sends a DMCC command string to the reader. The request data consists of the DMCC command string that is to be sent to the reader. The reply data contains the string result of the DMCC command. Additionally, the service provides a numeric result status for the call. Most of these result codes relate to the basic success/failure of the service execution. However, the service also maps the actual DMCC status codes. This allows the PLC to interpret the service request without having to parse the actual DMCC return string.

DMCC commands transferred through the Industrial Ethernet protocols are automatically routed to the Wi-Fi reader. The commands cannot be executed while the Wi-Fi reader is powered off, hibernating, or out-of-range.

Service Return Code	Description	DMCC Return Code
0	Success - No error	0
1	Bad Command	-
4	No Answer - System too busy	-
100	Unidentified error	100

Service Return Code	Description	DMCC Return Code
101	Command invalid	101
102	Parameter invalid	102
103	Checksum incorrect	103
104	Parameter rejected/altered due to reader state	104

Note: The DMCC command string must be in the CIP STRING2 format (16-bit integer indicating the string length in characters followed by the actual string characters, no terminating null required).

GetDecodeResults Service

The GetDecodeResults service reads data from the DecodeResults attribute of the ID Reader Object. This service takes parameters indicating the “size” (number of bytes to read) and the “offset” (offset into the DecodeResults attribute to begin reading). This gives the service the flexibility to be used with PLCs that have different restrictions on the amount of data allowed in an explicit message. It also allows you to access very large codes that cannot be completely transferred with implicit messaging (assembly object).

GetDecodeResults Request Data Format:

Name	Type	Description
Size	UINT	The number of bytes of the DecodeResults attribute to read.
Offset	UINT	The offset into the DecodeResults attribute. This specifies the first byte of the DecodeResults attribute to begin reading (0 based offset).

Assembly Object

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

DataMan is an I/O adapter class device. The convention for adapters is that Input Assemblies produce (transmit) data for another device (that is, DataMan to PLC) and Output Assemblies consume (receive) data from another device (that is, PLC to DataMan). DataMan acts as an I/O module for another device such as a PLC.

Assembly objects use implicit messaging. They are 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 (for example, 100ms or 200ms).

DataMan reader have a single input assembly and single output assembly. These assemblies combine selected attributes (data) of the DataMan ID Reader 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 ID Reader Object. However, using the assembly objects is much more efficient, thus they are the primary means of runtime communication between a DataMan reader and a PLC.

Input Assembly

The Input assembly provides status information, process state, and decode results.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
11	0	Reserved				Missed Acq	Acquiring	Trigger Ack	Trigger Ready
	1	General Fault	Reserved			Results Available	Results Buffer Overrun	Decode Complete Toggle	Decoding[1]
	2	SoftEvent Ack 7	SoftEvent Ack 6	SoftEvent Ack 5	SoftEvent Ack 4	SoftEvent Ack 3	SoftEvent Ack 2	SoftEvent Ack 1	SoftEvent Ack 0
	3-5	Reserved							
	6	Trigger ID (16-bit integer)							
	7								
	8	Result ID (16-bit integer)							
	9								
	10	Result Code (16-bit integer)							
	11								
	12	Result Extended (16-bit integer)							
	13								
	14	Result Data Length (16-bit integer)							
	15								
	16	Result Data 0							
	...								
	499	Result Data 483							

Output Assembly

The Output assembly contains control signals, software event signals, and any user data required for the trigger and decode.

[1] The value of the **Decoding** bit is identical to the value of the **Acquiring** bit. Both bits are true if an acquisition, or series of acquisitions of the same trigger, is in progress.

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
21	0	Reserved				Results Ack	Buffer Results Enable	Trigger	Trigger Enable
	1	SoftEvent 7	SoftEvent 6	SoftEvent 5	SoftEvent 4	SoftEvent 3	SoftEvent 2	SoftEvent 1	SoftEvent 0
	2	Reserved							
	3								
	4	User Data Option (16-bit integer)							
	5								
	6	User Data Length (16-bit integer)							
	7								
	8	User Data 0							
	...								
	499	User Data 491							

EtherNet/IP Operation

This section describes the acquisition sequence and decode result sequences.

Acquisition Sequence

The DataMan reader can be triggered to acquire images by several methods. It can be done either implicitly through the Assembly object, or done explicitly through the ID Reader object. When using explicit messaging, you can either:

- access the Acquire Service in a single step, or
- directly manipulate the ID Reader object attributes (AcqTrigger and AcqStatusRegister), or
- use DMCC commands.

The ID Reader attributes are discussed in this section. These same values can also be accessed through the assembly objects. On startup, the AcqTriggerEnable attribute is False. Set the attribute to True to enable triggering. When the reader is ready to accept triggers, the Trigger Ready bit in the AcqStatusRegister is set to True.

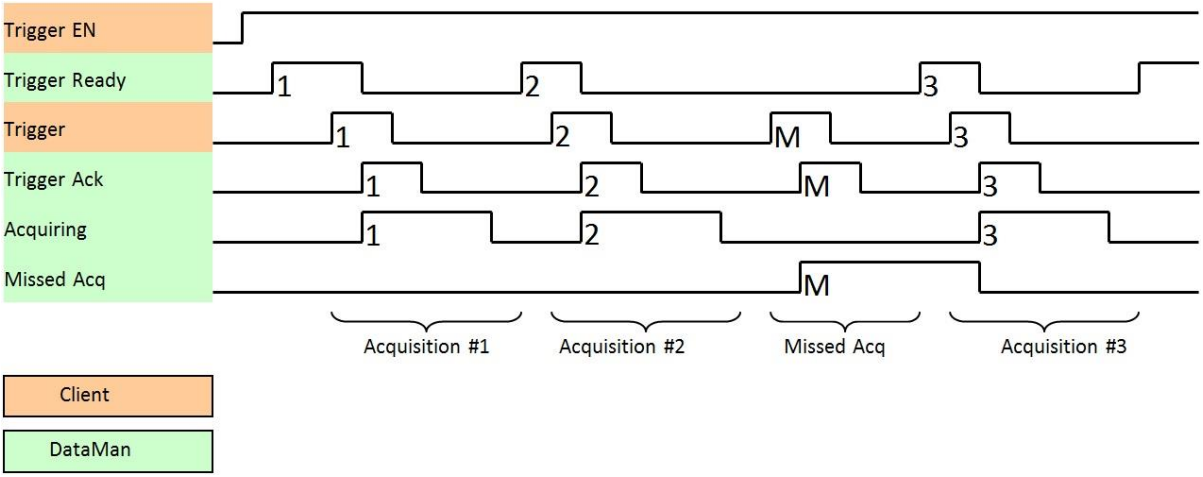
While the AcqStatusRegister “Trigger Ready” bit is True, each time the ID Reader object sees the AcqTrigger attribute change from 0 to 1, it initiates an image acquisition. When setting the Trigger Ready bit to True through the assembly objects, make sure that the attribute is held in the new state until 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 ID Reader object.

During an acquisition, the Trigger Ready bit in the AcqStatusRegister is cleared and the Acquiring bit is set to True. When the acquisition is completed, the Acquiring bit is cleared. The Trigger Ready bit is again set to True once the reader is ready to begin a new image acquisition.

The reader allows acquisitions to overlap with the decoding of previous acquisitions. The Trigger Ready bit is set high after the acquisition is complete, while decoding may still be in progress. The Decoding bit is deprecated and only mirrors the behavior of the Acquiring bit. If trigger queuing is active or other trigger sources can interfere, the Trigger Ready bit may not be reliable. As result, conflicting trigger overruns may occur, which are reported in Bit 3 of the ResultCode attribute of the ID Reader object.

In certain cases, you can cancel an acquisition by clearing the Trigger signal before the read operation is finished. This allows you to cancel reads in Presentation and Manual mode if no code is in the field of view. To ensure that a read is not

unintentionally canceled, make sure that the PLC holds the Trigger signal True until both TriggerAck and ResultsAvailable are True (or DecodeComplete toggles state).



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

Decode / Result Sequence

After an image is acquired, it is decoded. When the decoding is complete and a result is ready to be delivered, the reader toggles the *DecodeComplete* bit of the *DecodeStatusRegister*.

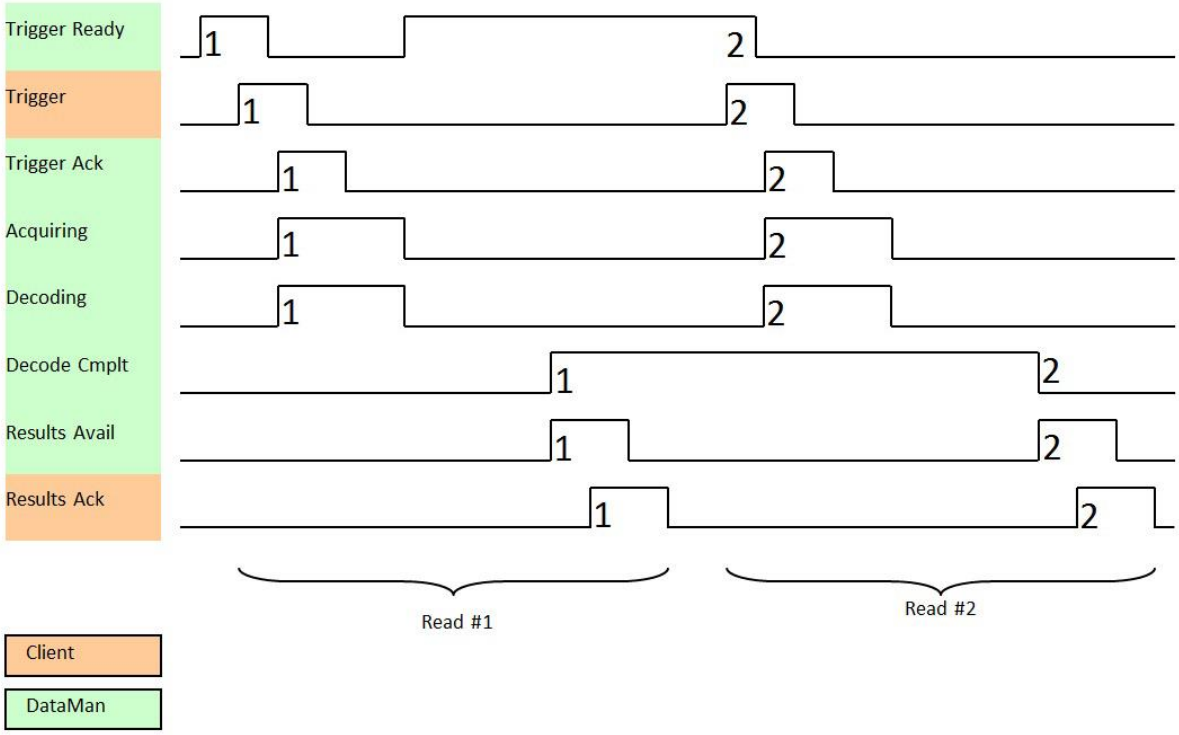
Decode results are reported asynchronously to the ID Reader Object. If BufferResultsEnable is set to False, then the decode results are immediately placed into DecodeResults, and the Results Available bit is set to True.

Note: The only way to ensure that you are not losing results is to use BufferResults. Make sure that your application can read each DecodeResult in time if you do not use BufferResultsEnable.

If the BufferResultsEnable attribute is set to True, the new results are queued. The earlier decode results remain in the DecodeResults attribute until they are acknowledged by the client setting the ResultsAck attribute to True. After the Results Available bit is cleared, the client sets the ResultsAck attribute back to False to allow the next queued results to be placed in to the DecodeResults attribute. This is a necessary handshake to ensure that the PLC receives the results, even if short gaps occur between results.

Behavior of DecodeStatusRegister

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
1	Decoding	Set when acquiring and decoding an image. The value of the Decoding bit is always the same as the Acquiring bit.	Set when acquiring and decoding an image. The value of the Decoding bit is always the same as the Acquiring bit.
2	Decode Complete	Toggled on completion of an image decode.	Toggled on completion of an image decode.
3	Results Buffer Overflow	Remains set to zero.	Set when decode results could not be queued because the client failed to acknowledge a previous result. Cleared when the decode result is successfully queued.
4	Results Available	Set when new results are placed in the DecodeResults attribute. Stays set until the results are acknowledged by setting ResultsAck to true.	Set when new results are placed in the DecodeResults attribute. Stays set until the results are acknowledged by setting ResultsAck to true.



Results Buffering

The Results Buffering feature allows a queue for decode results. When enabled, the feature queues a finite number of result data sets until the PLC (client) has time to read them. This smooths out data flow if the PLC slows down for short periods or if there are surges in read activity.

If the feature is enabled, the reader overlaps acquisition and decode operations. This can lead to faster over all trigger rates. When reads occur faster than results can be sent out (the queue is full), the most recent results are discarded until space becomes available in the results queue.

If the feature is disabled, the more recent results overwrite the earlier results. The most recent results are kept. Earlier results are discarded if the PLC does not read them fast enough.

If the queue overflows while the feature is disabled:

- The difference between the TriggerID and ResultID values becomes greater than 1. This difference represents the number of reads that occurred but could not be queued because the queue was full.
- The number of lost reads equals TriggerID - ResultID - 1.
- After the next read, the ResultID value returns to the typical operating value of TriggerID - 1.

Result Fragmentation

The Result Fragmentation feature splits decode result strings into fragments if the strings are too large for the Results Data field. The fragments are then pushed to the buffer like normal results. Then, the PLC reads the fragments one by one.

Result Fragmentation depends on Result Buffering. This means that results are only fragmented if Result Buffering is enabled.

Note: For Ethernet/IP, you have to enable Result Fragmentation manually.

Enabling Result Fragmentation

To enable or disable Result Fragmentation, you have the following options:

- Use the FFP.RESULT-FRAG DMCC.
- Use SoftEvent 6 to enable the Results Buffering and Result Fragmentation features from the PLC.

Operation of Result Fragmentation

The PLC does not automatically reassemble the result string. Once the PLC reads a result fragment, you have to copy the fragment to a new user-defined variable. After this, you have to set and clear the ResultAck bit, which updates the Decode Results with the next fragment. This process continues until all fragments have been sent to the PLC.

In contrast, if Results Buffering is disabled, large decode result strings are cut off to fit into the Results Data field, which means that the end of the string gets lost.

Identifying Decode Result Chunks

Decode result fragments look like normal results. To identify result fragments, look at the Result ID:

- The Result ID of the result fragments are the same.
- The Result ID of the last result fragment is different from the Result ID of the next result.

Independent of whether results fragmentation is enabled, the Result Length always represents the number of characters to be read from the Results Data plus the remaining characters that did not fit into the Results Data buffer. With results fragmentation enabled, the remaining characters will be found in follow-up fragments, the remaining characters are discarded.

You can identify the first and last fragments of a result as follows:

- The first fragment has a different Result ID from the previous Result ID.
- The Result Length of the last fragment is lower or equal to the Result Data buffer size.
- All fragments of a result share the same Result ID.

Maximum Result Data Length

Without results fragmentation, result data is truncated to fit into the results data buffer size. Depending on the protocol, the result buffer size can be fixed or configurable.

The Result Fragmentation feature allows processing of up to 50 times longer results, depending on the results buffering queue size, with an absolute limit of 65535 characters.

The maximum number of result characters for EtherNet/IP is 470 bytes (fixed).

DMCC Result Feedback from PLC

The DMCC Result Feedback feature allows the reader to send DMCC responses back to the PLC.

By default, this feature is off. To enable the feature, use the `FFP.DMCC-RESULTS` DMCC command. Changing this configuration does not require you to reboot or reconnect reader.

When you enable the feature, the reader sends DMCC responses to the PLC using the same mechanism it uses to send read results. To distinguish DMCC results from read results, bit 6 is of the ResultCode field is set to 1. The following Input Assembly attributes contain DMCC information instead of read results:

Byte	Name	Description

8-9	ResultID	Contains a DMCC result code: • 0: no error • 100: unidentified error • 101: invalid command • 102: invalid parameter • 103: incorrect checksum • 104: parameter rejected/altered due to reader state • 105: reader unavailable (offline)
10-11	ResultCode	Bit 6 is set to 1 to indicate a DMCC response.
14-15	ResultLength	Actual number of bytes of DMCC result data contained in the *ResultData* field (16-bit Integer).
16...	ResultData	DMCC result data (array of bytes).

You can use DMCC Result Feedback in combination with Results Fragmentation.

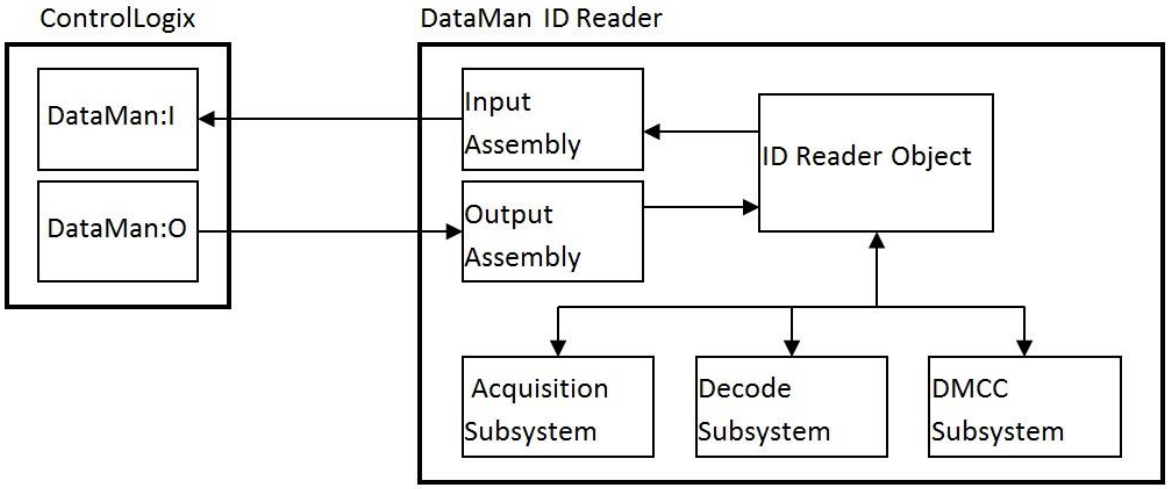
Rockwell ControlLogix Examples

Implicit messages transmit time-critical application specific I/O data and can be point-to-point or multicast. Explicit messages require a response from the receiving device. As a result, explicit messages are better suited for operations that occur less frequently. An instruction to send a DMCC command is an example of an explicit message.

Implicit Messaging

EtherNet/IP implicit messaging allows the inputs and outputs of a DataMan reader to be mapped into tags in the ControlLogix PLC. Once these connections are established, the data is transferred cyclically at a user defined interval (10ms, 50ms, 100ms, and so on).

The figure below represents Ethernet-based I/O through EtherNet/IP:



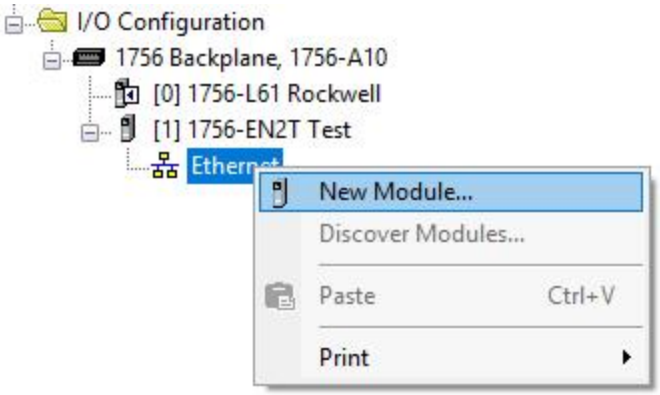
The *Input Assembly* and *Output Assembly* map various attributes (data) from the ID Reader object: The Input Assembly is the collection of reader data values sent to the PLC (PLC inputs); and the Output Assembly is the collection of data values received by the reader from the PLC (PLC outputs).

Establishing an Implicit Messaging Connection

To setup an EtherNet/IP implicit messaging connection between a reader and a ControlLogix controller, the reader must first be added to the ControlLogix I/O Configuration tree. The most efficient method is to use the Add-on Profile. This example assumes that the Add-on Profile has already been installed. If you do not have the Add-on Profile, see section Using the Generic EtherNet/IP Profile.

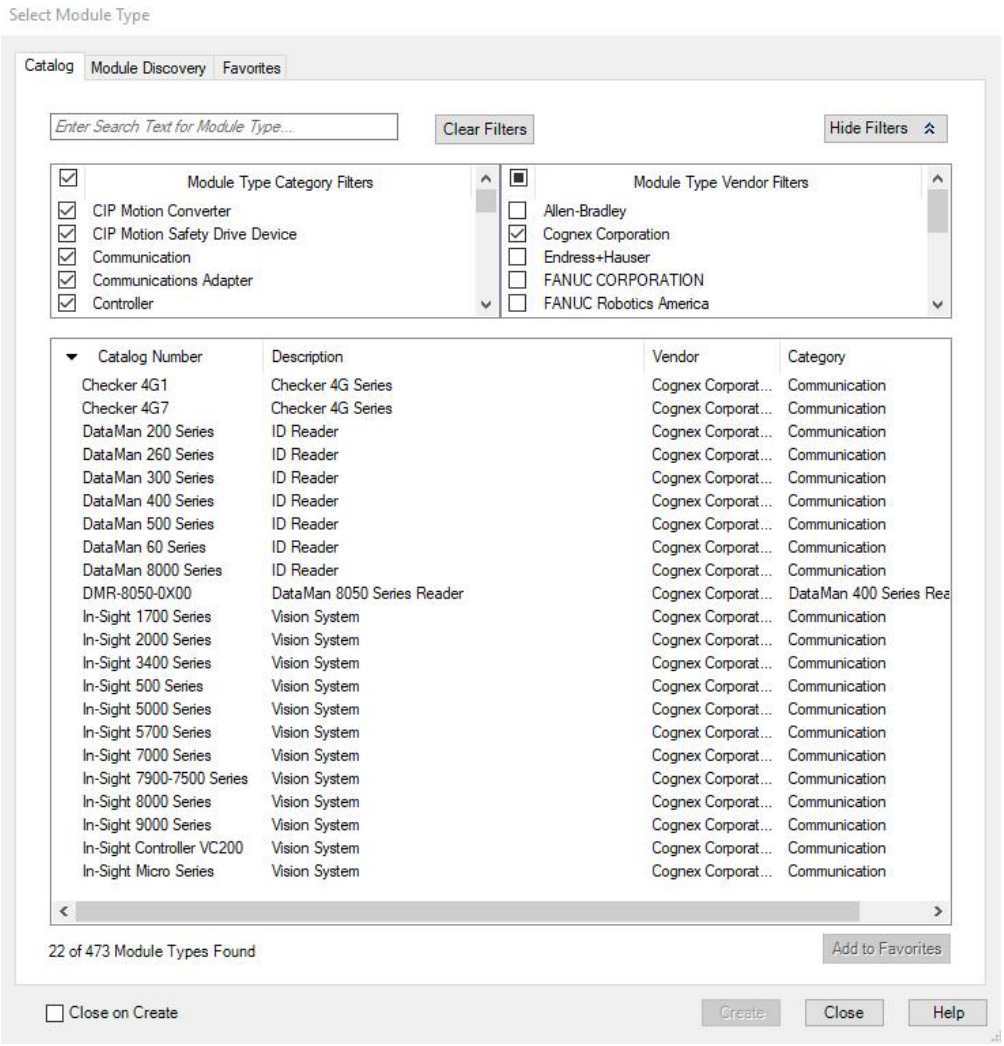
To establish an implicit messaging connection with a ControlLogix PLC:

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



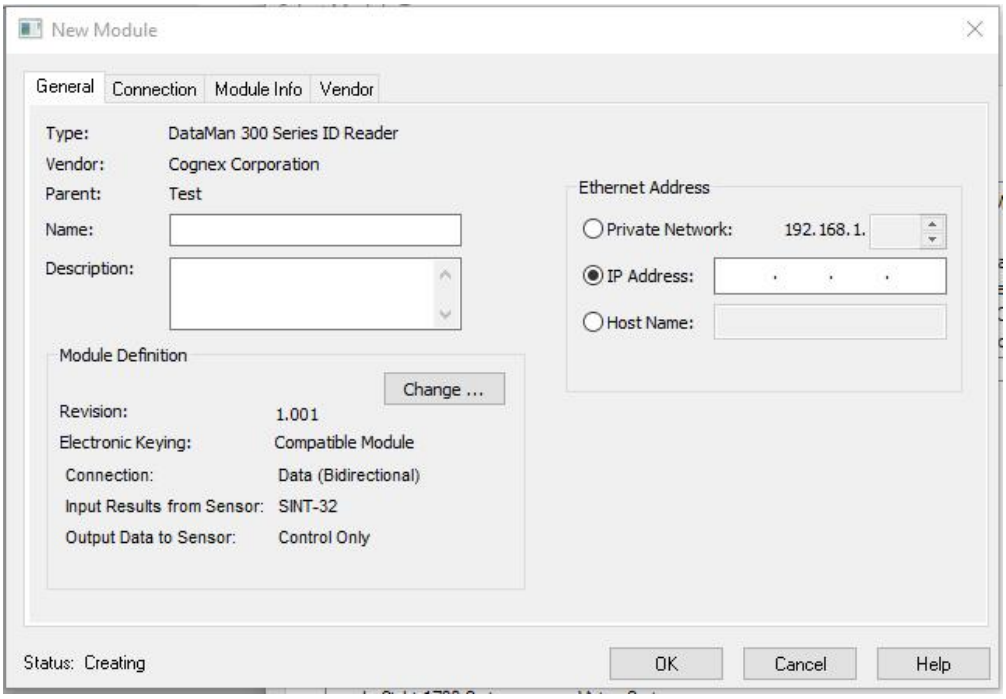
2. From the Select Module dialog, choose your model of DataMan ID Reader from the list.

> **Note**: This option is only available after you install the DataMan Add-On Profile.

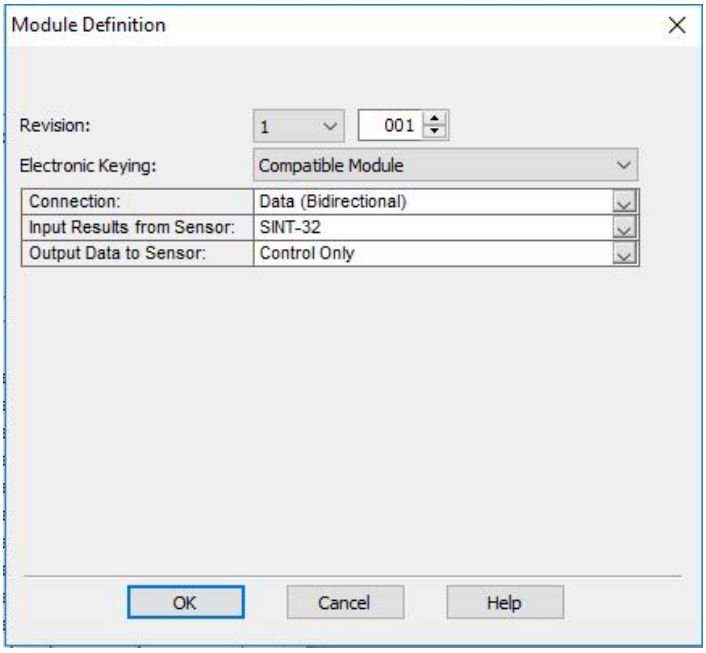


Note: The rest of the steps is identical regardless of which DataMan model is selected.

3. After selecting the device, the configuration dialog for the DataMan ID Reader system is displayed. Give the module a name and enter the reader's IP address. The default is a bidirectional (send/receive) connection consisting of control, status, and 32 bytes of result data with keying disabled. To change this default connection, click the **Change...** button. If no change is required, skip the next step.



4. Clicking the **Change...** button brings up the **Module Definition** dialog. This dialog is used to alter the connection configuration. You can change:
    - DataMan revision
    - Electronic keying
    - Connection type (bidirectional/receive-only)
    - Amount of data received (from the reader)
    - Amount of data sent (to the reader)



Electronic Keying: Defines the level of module type checking that is performed by the PLC before a connection is established.

Exact Match – All of the parameters must match or the connection will be rejected.

- Vendor
- Product Type
- Catalog Number
- Major Revision
- Minor Revision

Compatible Module – To prevent the inserted module from rejecting the connection:

- The Module Types have to match
- Catalog Number has to match
- Major Revision has to match
- The Minor Revision of the module has to be equal to or greater than the one specified in the software.

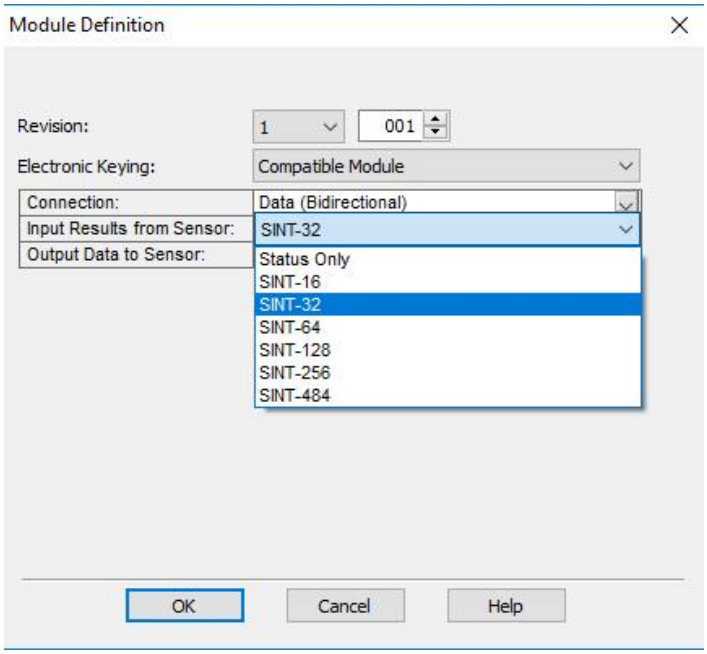
Disable Keying – The controller does not employ keying at all.

Connection: Defines the type of data flow.

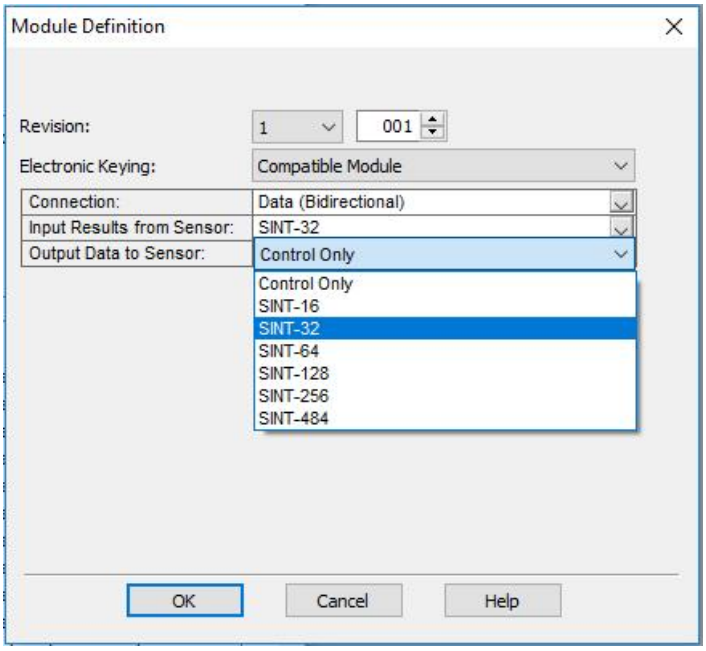
Data (Bidirectional) – The connection sends data to the reader and receives data from the reader.

Input (Results only) – The connection only receives data from the reader. If more than one PLC needs to receive data from the same reader, choose the Input connection option.

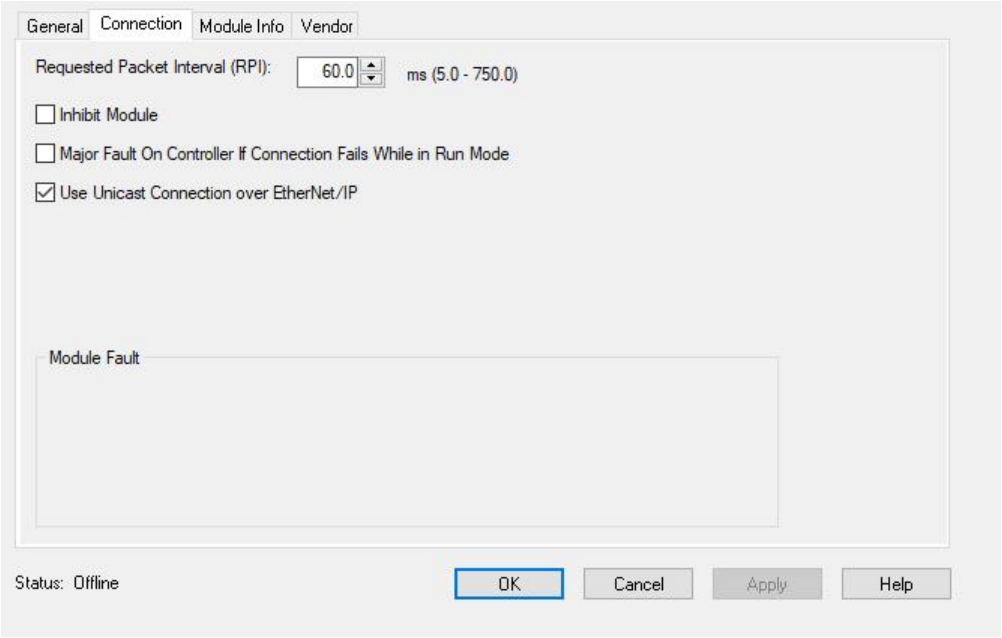
Input Results from Sensor: Defines the amount of data received on the connection from the reader. The minimum amount is the Status data only. The connection can be configured to also receive read result data. The amount of result data received is defined in fixed increments (16 bytes, 32 bytes, 64 bytes, and so on). Select the size to return no more than the largest code size to be read by the application. Setting the size larger wastes network bandwidth and diminishes performance.



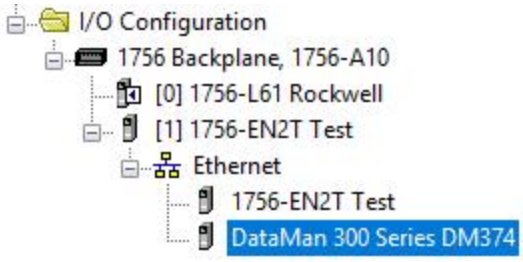
Output Data to Sensor: Defines the amount of data transmitted on the connection (to the reader). The minimum amount is the Control data only. The connection can be configured to also send user data. The amount of user data sent is defined in fixed increments (16 bytes, 32 bytes, 64 bytes, and so on). To enable User Data output, right-click the DataMan module and then go to Properties -> Change -> Output Data to Sensor.



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, do not set this rate lower than required by a given application, or lower than half the expected maximum read rate of the user application. Setting it lower wastes bandwidth and does not improve processing performance.
6. Select the “Connection” tab of the “New Module” dialog to set the rate.

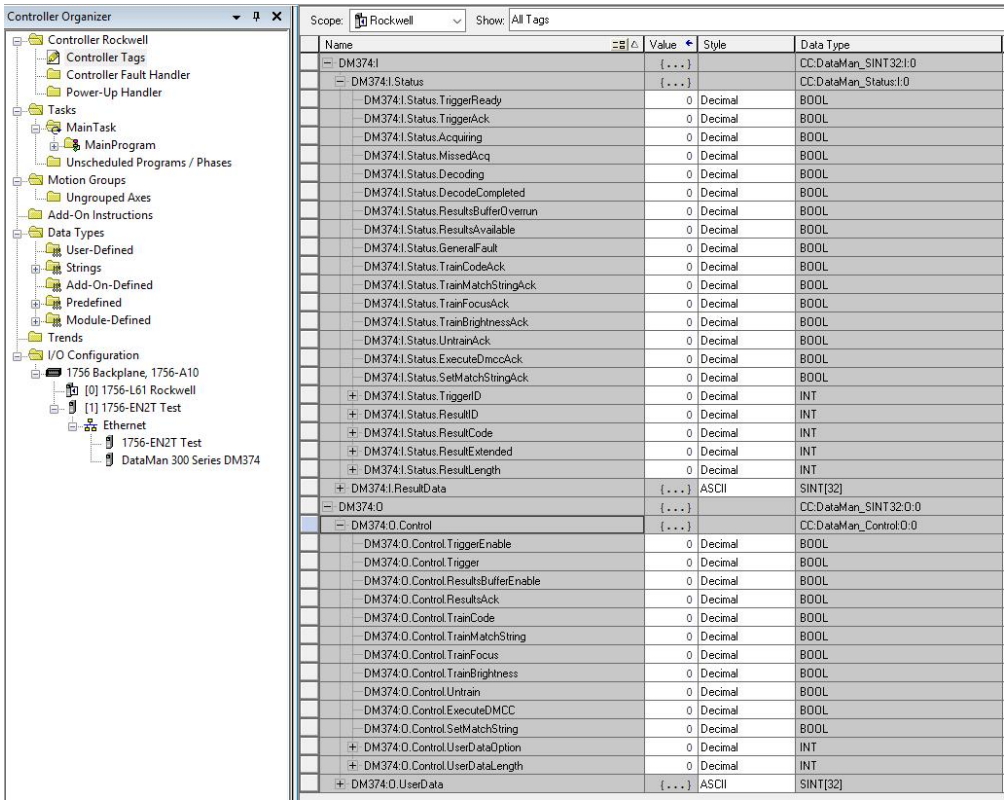


7. After adding the module to ControlLogix, the I/O tree should appear as follows:



8. When the DataMan module is added to the I/O tree, RSLogix 5000 creates tags that map to the reader Input and Output Data (that is, the Input and Output Assembly Objects in the DataMan). 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 DataMan Module that you added to the I/O Configuration in the previous steps.



The tags are organized in two groups: Status and Control. The Status group represents all the data being received from the reader. The Control group represents all the data being sent to the reader.

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

> **Note**: Based on the configured RPI, there is a time delay between the DataMan and the PLC tag values. Take this time delay into account when writing all PLC ladders.

Accessing Implicit Messaging Connection Data

One aspect of the Add-on Profile is that it automatically generates ControlLogix tags representing the connection data.

The generated tags are divided into two groups: Status and Control. The Status group represents all the data being received from the reader. The Control group represents all the data being sent to the reader.

Status tag group is the data the ControlLogix receives from the reader:

DM374:I	{...}		CC:DataMan_SINT32:I:0
DM374:I.Status	{...}		CC:DataMan_Status:I:0
DM374:I.Status.TriggerReady	0	Decimal	BOOL
DM374:I.Status.TriggerAck	0	Decimal	BOOL
DM374:I.Status.Acquiring	0	Decimal	BOOL
DM374:I.Status.MissedAcq	0	Decimal	BOOL
DM374:I.Status.Decoding	0	Decimal	BOOL
DM374:I.Status.DecodeCompleted	0	Decimal	BOOL
DM374:I.Status.ResultsBufferOverrun	0	Decimal	BOOL
DM374:I.Status.ResultsAvailable	0	Decimal	BOOL
DM374:I.Status.GeneralFault	0	Decimal	BOOL
DM374:I.Status.TrainCodeAck	0	Decimal	BOOL
DM374:I.Status.TrainMatchStringAck	0	Decimal	BOOL
DM374:I.Status.TrainFocusAck	0	Decimal	BOOL
DM374:I.Status.TrainBrightnessAck	0	Decimal	BOOL
DM374:I.Status.UntrainAck	0	Decimal	BOOL
DM374:I.Status.ExecuteDmccAck	0	Decimal	BOOL
DM374:I.Status.SetMatchStringAck	0	Decimal	BOOL
DM374:I.Status.TriggerID	0	Decimal	INT
DM374:I.Status.ResultID	0	Decimal	INT
DM374:I.Status.ResultCode	0	Decimal	INT
DM374:I.Status.ResultExtended	0	Decimal	INT
DM374:I.Status.ResultLength	0	Decimal	INT
DM374:I.ResultData	{...}	ASCII	SINT[32]

- **TriggerReady**: Indicates when the reader can accept a new trigger. This tag is True when the Control tag “TriggerEnable” has been set , and the sensor is not acquiring an image.
- **TriggerAck**: Indicates when the reader has been triggered (that is, the Control tag “Trigger” has been set to True). This tag stays set until the Trigger tag is cleared.
- **Acquiring**: Indicates when the <MadCap:variable name="Global/Global.Product Line" /> reader is acquiring an image either by setting the Trigger bit or by an external trigger.
- **MissedAcq**: Indicates when the DataManreader misses an acquisition trigger. This happens when trigger on through the Trigger On bit of Ethernet/IP fails. It is cleared when the next successful acquisition occurs.
- **Decoding**: Deprecated. Same behavior as **Acquiring**.
- **DecodeCompleted**: Tag value is toggled (1 to 0 or 0 to 1) when a decode is completed.
- **ResultsBufferOverrun**: Indicates when the reader discards a set of decode results because the results queue is full. Cleared when the next set of results are successfully queued.
- **ResultsAvailable**: Indicates when a set of decode results are available (that is, the ResultID, ResultCode, ResultLength and ResultsData tags contain valid data).
- **GeneralFault**: Indicates when a fault has occurred. Currently only used by Soft Event operations.
- **TrainCodeAck**: Indicates that the SoftEvent “TrainCode” is complete.
- **TrainMatchStringAck**: Indicates that the SoftEvent “TrainMatchString” has completed.
- **TrainFocusAck**: Indicates that the SoftEvent “TrainFocus” has completed.
- **TrainBrightnessAck**: Indicates that the SoftEvent “TrainBrightness” has completed.
- **UnTrainAck**: Indicates that the SoftEvent “UnTrain” has completed.
- **ExecuteDmccAck**: Indicates that the SoftEvent “ExecuteDMCC” has completed.
- **SetMatchStringAck**: Indicates that the SoftEvent “SetMatchString” has completed.

- **TriggerID**: Value of the next trigger to be issued. Used to match triggers issued with corresponding result data received later.
- **ResultID**: The value of TriggerID when the trigger that generated these results was issued. Used to match TriggerID's with result data.
- **ResultCode**: Indicates success/failure of this set of results.
    - **Bit 0** ,1=read 0=no read
    - **Bit 1** ,1=validated 0=not validated (or validation not in use)
    - **Bit 2** ,1=verified 0=not verified (or verification not in use)
    - **Bit 3** ,1=acquisition trigger overrun
    - **Bit 4** ,1=acquisition buffer overflow (not the same as result buffer overflow).
    - **Bits 5-15** , reserved (future use)
- **ResultExtended**: Unused.
- **ResultLength**: Number of bytes of result data contained in the ResultData tag.
- **ResultData**: Decode result data. Control tag group is the data sent from the ControlLogix to the reader:

DM374:0	{...}		CC:DataMan_SINT32:0:0
DM374:0.Control	{...}		CC:DataMan_Control:0:0
DM374:0.Control.TriggerEnable	0	Decimal	BOOL
DM374:0.Control.Trigger	0	Decimal	BOOL
DM374:0.Control.ResultsBufferEnable	0	Decimal	BOOL
DM374:0.Control.ResultsAck	0	Decimal	BOOL
DM374:0.Control.TrainCode	0	Decimal	BOOL
DM374:0.Control.TrainMatchString	0	Decimal	BOOL
DM374:0.Control.TrainFocus	0	Decimal	BOOL
DM374:0.Control.TrainBrightness	0	Decimal	BOOL
DM374:0.Control.Untrain	0	Decimal	BOOL
DM374:0.Control.ExecuteDMCC	0	Decimal	BOOL
DM374:0.Control.SetMatchString	0	Decimal	BOOL
DM374:0.Control.UserDataOption	0	Decimal	INT
DM374:0.Control.UserDataLength	0	Decimal	INT
DM374:0.UserData	{...}	ASCII	SINT[32]

- **TriggerEnable**: Setting this tag enables EtherNet/IP triggering. Clearing this field disables the EtherNet/IP triggering.
- **Trigger**: Setting this tag triggers an acquisition if:
    - The TriggerEnable tag is set.
    - No acquisition/decode is currently in progress.
    - The device is ready to trigger.
- **ResultsBufferEnable**: When set, the decode results are queued. Results are made available each time the PLC acknowledges the current results. The Decode ID, Decode Result and Decode ResultsData fields are held constant until the ResultsAck field acknowledged and set them. The reader responds to the acknowledgment by clearing the ResultsValid bit. Once the ResultsAck field is cleared, the next set of decode results are posted.
- **ResultsAck**: The ResultsAck tag is used to acknowledge that the PLC read the latest results. When ResultsAck is set, the ResultsAvailable tag will be cleared. If results buffering is enabled, the next set of results are made available when the ResultsAck tag is cleared again.
- **TrainCode**: Changing this tag from 0 to 1 invokes the train code operation.
- **TrainMatchString**: Changing this tag from 0 to 1 invokes the train match string operation.
- **TrainFocus**: Changing this tag from 0 to 1 invokes the train focus operation.

- **TrainBrightness**: Changing this tag from 0 to 1 invokes the train brightness operation.
- **Untrain**: Changing this tag from 0 to 1 invokes the un-train operation.
- **ExecuteDMCC**: Changing this tag from 0 to 1 invokes the DMCC operation. A valid DMCC command string must be written to UserData prior to invoking this SoftEvent.
- **SetMatchString**: Changing this tag from 0 to 1 invokes the set match string operation. The match string data must be written to UserData prior to invoking this SoftEvent.
- **UserDataOption**: Optional user data information. For more information, see the PROFINET documentation.
- **UserDataLength**: Number of bytes of user data contained in the UserData tag.
- **UserData**: This data is sent to the reader to support acquisition and/or decode.

> **Note**:
> Configure the DataMan module in RSLogix 5000 to manually add **UserData** to the output assembly.

Perform the following steps on a CompactLogix or ControlLogix PLC:

1. Right click the DataMan module and select **Properties**.
2. Under **Module Definition**, click **Change**.
3. The Module Definition window pops up. Under the **Output Data to Sensor** drop-down menu, select **SINT-484**.
4. Click **OK**. RSLogix 500 is now updating the Module Definition.

The output assembly controller tags now list UserData as part of the output assembly.

Verifying Implicit Messaging Connection Operation

After the reader is added as an I/O device in a ControlLogix project and the project is downloaded to the controller, the I/O connection needs to be established. Once a successful connection is established, cyclic data transfers are initiated at the requested RPI.

To verify a proper I/O connection, follow these steps:

1. Download the project to the ControlLogix controller.
2. When the download completes, the project I/O indicator is **I/O OK**, indicating that the I/O connection is successfully completed.

   To verify the correct, two-way transfer of I/O data, go to the controller tags in RSLogix and change the state of the TriggerEnable bit from 0 to 1:

⊟ DM374:O	{...}		CC:DataMan_SINT32:O:0
⊟ DM374:O.Control	{...}		CC:DataMan_Control:O:0
DM374:O.Control.TriggerEnable	1	Decimal	BOOL
DM374:O.Control.Trigger	0	Decimal	BOOL
DM374:O.Control.ResultsBufferEnable	0	Decimal	BOOL
DM374:O.Control.ResultsAck	0	Decimal	BOOL
DM374:O.Control.TrainCode	0	Decimal	BOOL
DM374:O.Control.TrainMatchString	0	Decimal	BOOL
DM374:O.Control.TrainFocus	0	Decimal	BOOL
DM374:O.Control.TrainBrightness	0	Decimal	BOOL
DM374:O.Control.Untrain	0	Decimal	BOOL
DM374:O.Control.ExecuteDMCC	0	Decimal	BOOL
DM374:O.Control.SetMatchString	0	Decimal	BOOL
⊞ DM374:O.Control.UserDataOption	0	Decimal	INT
⊞ DM374:O.Control.UserDataLength	0	Decimal	INT

3. The TriggerReady tag changes to 1. Triggering is now enabled.
4. Whenever the Trigger tag is changed from 0 to 1, the reader acquires an image.

> **Note**: The current TriggerID value is 1. Make sure that the results of the next trigger to be issued come back with a corresponding ResultID of 1.

5. After the acquisition/decode has completed, the DecodeCompleted tag will toggle, and the ResultsAvailable tag goes to 1. This example shows a successful read (ResultCode bit 0 = 1) and the read has returned 16 bytes of data (ResultLength=16). The data is in the ResultData tag.

Name	Value	Style	Data Type
DM374:I	{...}		CC:DataMan_SINT32:I:0
DM374:I.Status	{...}		CC:DataMan_Status:I:0
DM374:I.Status.TriggerReady	1	Decimal	BOOL
DM374:I.Status.TriggerAck	0	Decimal	BOOL
DM374:I.Status.Acquiring	0	Decimal	BOOL
DM374:I.Status.MissedAcq	0	Decimal	BOOL
DM374:I.Status.Decoding	0	Decimal	BOOL
DM374:I.Status.DecodeCompleted	1	Decimal	BOOL
DM374:I.Status.ResultsBufferOverrun	0	Decimal	BOOL
DM374:I.Status.ResultsAvailable	1	Decimal	BOOL
DM374:I.Status.GeneralFault	0	Decimal	BOOL
DM374:I.Status.TrainCodeAck	0	Decimal	BOOL
DM374:I.Status.TrainMatchStringAck	0	Decimal	BOOL
DM374:I.Status.TrainFocusAck	0	Decimal	BOOL
DM374:I.Status.TrainBrightnessAck	0	Decimal	BOOL
DM374:I.Status.UntrainAck	0	Decimal	BOOL
DM374:I.Status.ExecuteDmccAck	0	Decimal	BOOL
DM374:I.Status.SetMatchStringAck	0	Decimal	BOOL
DM374:I.Status.TriggerID	297	Decimal	INT
DM374:I.Status.ResultID	296	Decimal	INT
DM374:I.Status.ResultCode	1	Decimal	INT
DM374:I.Status.ResultExtended	0	Decimal	INT
DM374:I.Status.ResultLength	16	Decimal	INT
DM374:I.ResultData	{...}	ASCII	SINT[32]
DM374:O	{...}		CC:DataMan_NODATA:O:0
DM374:O.Control	{...}		CC:DataMan_Control:O:0
DM374:O.Control.TriggerEnable	0	Decimal	BOOL
DM374:O.Control.Trigger	0	Decimal	BOOL
DM374:O.Control.ResultsBufferEnable	0	Decimal	BOOL
DM374:O.Control.ResultsAck	0	Decimal	BOOL
DM374:O.Control.TrainCode	0	Decimal	BOOL

Explicit Messaging

Unlike implicit messaging, explicit messages are sent to a specific device that always sends a reply to that message. As a result, explicit messages are better suited for operations that occur infrequently. Explicit messages can be used to read and write the attributes (data) of the ID Reader Object. They can also be used for acquiring images, sending DMCC commands, and retrieving result data.

Issuing DMCC Commands

One of the more common explicit messages sent to a DataMan ID reader is an instruction to execute a DMCC command. Explicit messages are sent from ControlLogix to a reader using MSG instructions. Use the SendDMCC service to invoke DMCC messages with explicit messaging.

The CIP STRING2 format is required for transmission across EtherNet/IP: a 16-bit length value followed by actual string characters, no null terminator. Logix stores strings in a slightly different format: a 32-bit length value followed by actual string

characters, no null terminator. Therefore, some of the sample ladder involves converting to/from the two different string formats.

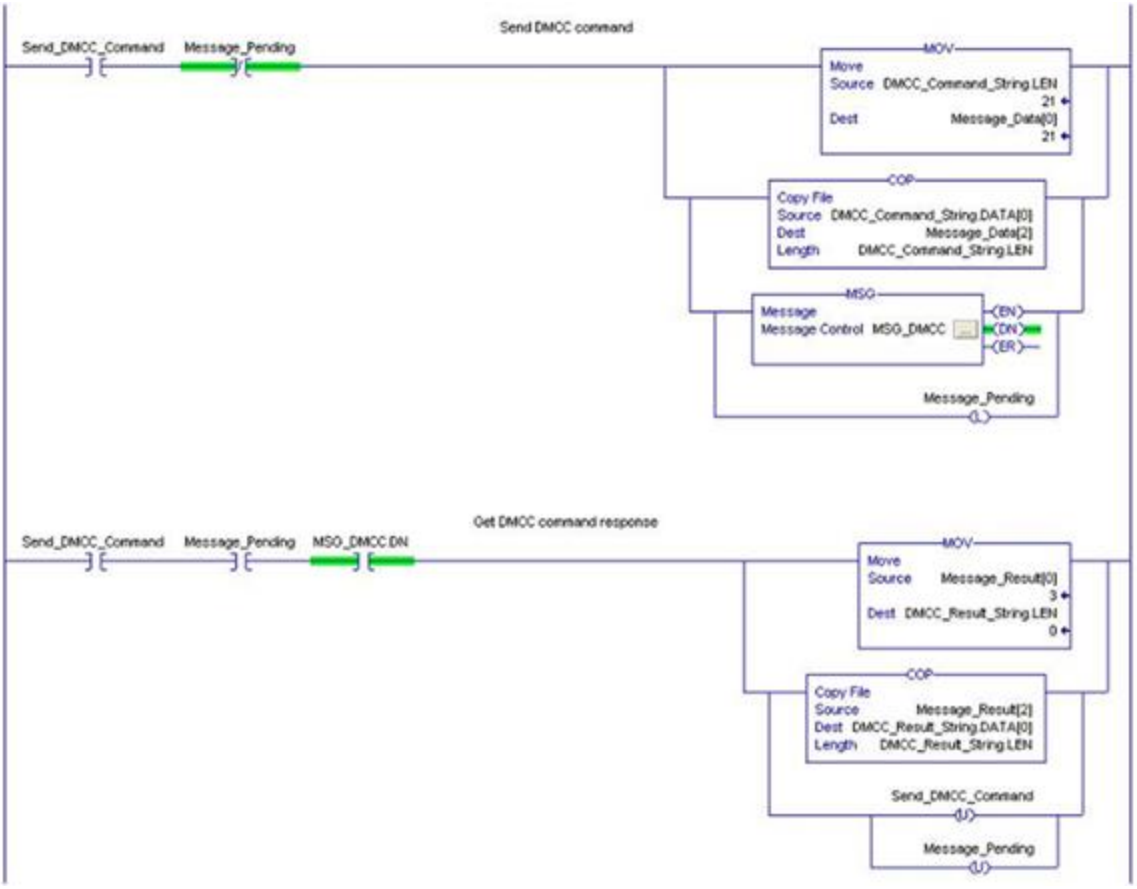
Note: The instruction to send a DMCC is intended as a demonstration of DataMan explicit messaging behavior. This same operation could be written in a much more efficient ladder but would be less useful as a learning tool.

1. Add the following tags to the ControlLogix Controller Tags dialog:

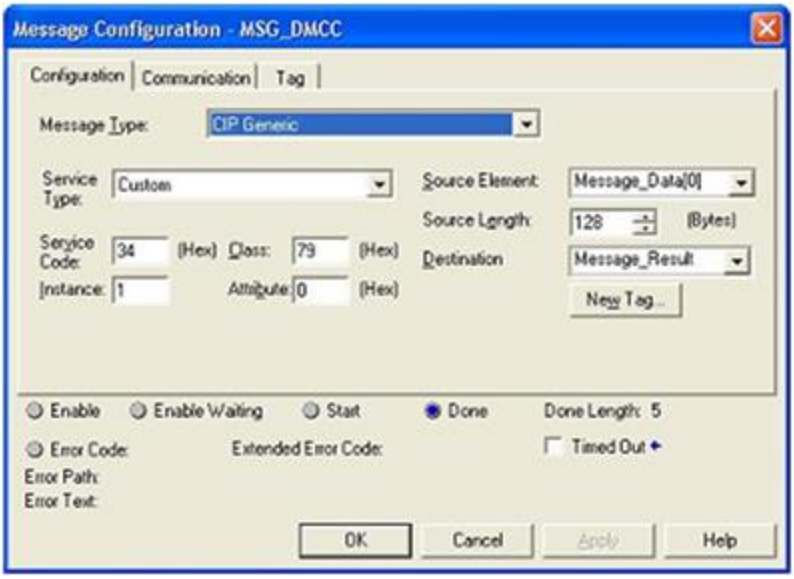
Name	Data Type	Style
Send_DMCC_Command	BOOL	Decimal
⊞ DMCC_Command_String	STRING	
⊞ DMCC_Result_String	STRING	
⊞ Message_Data	SINT[128]	Decimal
⊞ Message_Result	SINT[128]	Decimal
Message_Pending	BOOL	Decimal
⊞ MSG_DMCC	MESSAGE	

- **Send_DMCC_Command**: Boolean flag used to initiate the command.
- **DMCC_Command_String**: String containing the DMCC command to execute.
- **DMCC_Result_String**: String receiving the DMCC command results.
- **Message_Data**: Temp buffer holding the data to send via the MSG instruction.
- **Message_Result**: Temp buffer holding the data received via the MSG instruction.
- **Message_Pending**: Boolean flag used to indicate that a message is in process.
- **MSG_DMCC**: Data structure required by the Logix MSG instruction.

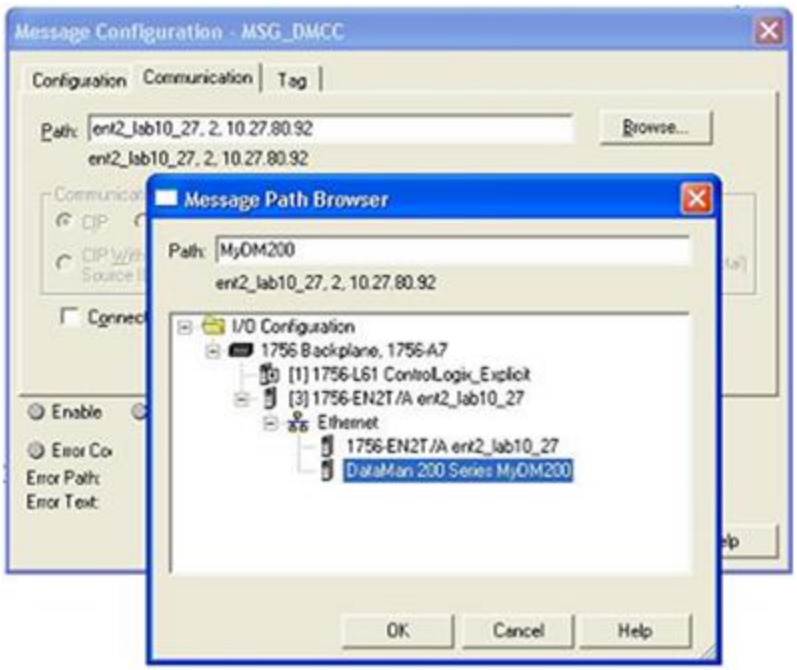
2. Add the following two rungs to the MainRoutine of your ControlLogix project:



3. Edit the MSG instruction. Configure it for “CIP Generic”, service 0x34 “SendDMCC”, class 0x79 “ID Reader Object” and instance 1. Set the source to “Message_Data” and the destination to “Message_Result”.



4. On the MSG instruction **Communication** tab, browse for and select the DataMan which you added to the project I/O Configuration tree. This tells Logix where to send the explicit message.



5. Download the project to the ControlLogix and place in “Run Mode”.

To operate:

1. Place a DMCC command in the “DMCC_Command_String” tag. For example “||>GET TRIGGER.TYPErl”. Note the rl at the end of the string. This is how Logix represents a CRLF.
2. Toggle the “Send_DMCC_Command” tag to 1.
3. When the “Send_DMCC_Command” tag goes back to 0 execution is complete. The DMCC command results can be found in “DMCC_Result_String”.

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.

DataMan supports PROFINET I/O. This is one of the two “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 Component Based Automation (CBA), is not supported.

By default, the DataMan has the PROFINET protocol disabled. The protocol can be enabled via DMCC, scanning a parameter code or in the DataMan Setup Tool.

Note: If you have a wireless reader, see *Industrial Protocols for the Wireless DataMan* on page 148

DMCC

The following commands can be used to enable/disable PROFINET. The commands can be issued via RS-232 or Telnet connection.

Note: Use a third party Telnet client such as PuTTY to communicate with your DataMan reader.

Enable:

```
||>SET PROFINET.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET PROFINET.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

Reader Configuration Code

Scanning the following reader configuration codes enables/disables PROFINET for your corded reader.

Note: You must reboot the reader for the change to take effect.

Enable:

Disable:

Scanning the following reader configuration codes enables/disables PROFINET for your DataMan 8000 Base Station.

Note: You must reboot the reader for the change to take effect.

Enable: Disable:

Setup Tool

1. Open Setup Tool to discover the reader.

   Note: The reader must be in the subnet.

2. If the reader appears as misconfigured, select the reader, and click **Repair & Support** to change the misconfigured IP address to the desired static IP.

   Note: Having a properly configured IP address is mandatory to further connect to the reader via Setup Tool.

3. Connect to the reader by clicking **Connect**.
4. Navigate to **Communications** application.
5. Enable PROFINET under **Ethernet** tab.

   Note: Enabling PROFINET immediately turns off DHCP and turns the current IP address to a static IP.

6. The Siemens engineering software (TIA or Step7) can discover the reader now.
7. Using the engineering software, configure how the reader is discovered by the PLC. PROFINET offers two modes:
   a. The reader has a permanent static IP address. The static IP address must be configured in the PLC project and match the setting on the reader.
   b. The IP address of the reader is configured centrally in the PLC project. In this case, the reader boots up with the permanent IP address of 0.0.0.0. Once the PLC identifies the reader on the network by the station name of the reader, PLC assigns the configured IP address temporarily to the device. In this case, 'temporarily' means that the reader starts again with the zero IP on the next boot.
8. Click **Save Settings** in the upper toolbar before disconnecting from the reader.

   Note: The new settings take effect only after the reader is rebooted.

Preparing to Use PROFINET

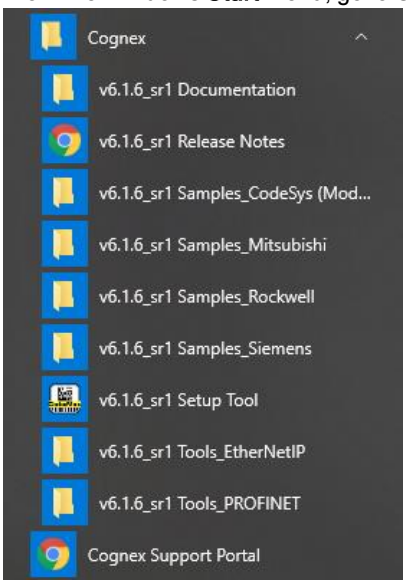
Preparing to use PROFINET involves the following main steps:

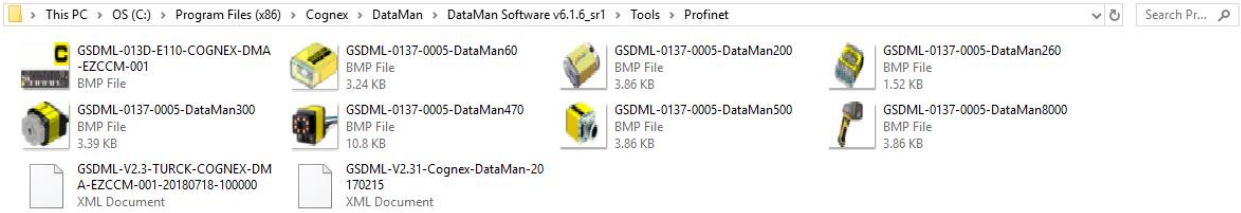
- Make sure that you have either the SIMATIC Step 7 programming software or TIA Portal installed.
- Set up the Siemens Software tool so that it recognizes your DataMan device.
- Install the Generic Station Description (GSD) file.

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.

Perform the following steps to set up PROFINET using SIMATIC or TIA Portal:

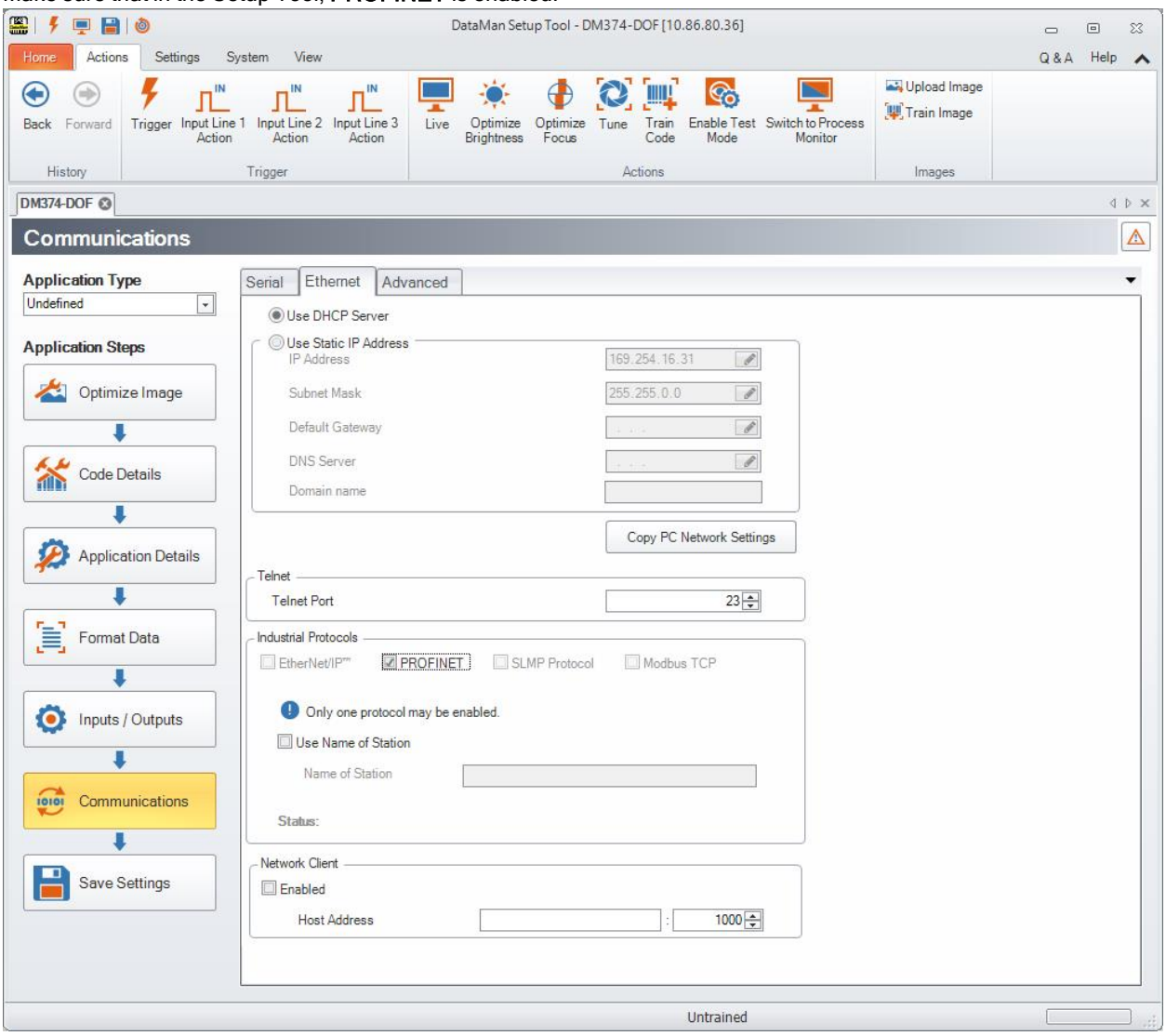
1. Verify that SIMATIC is on your machine.
2. From the Windows **Start** menu, go to Cognex and select the folder that contains the PROFINET tools.



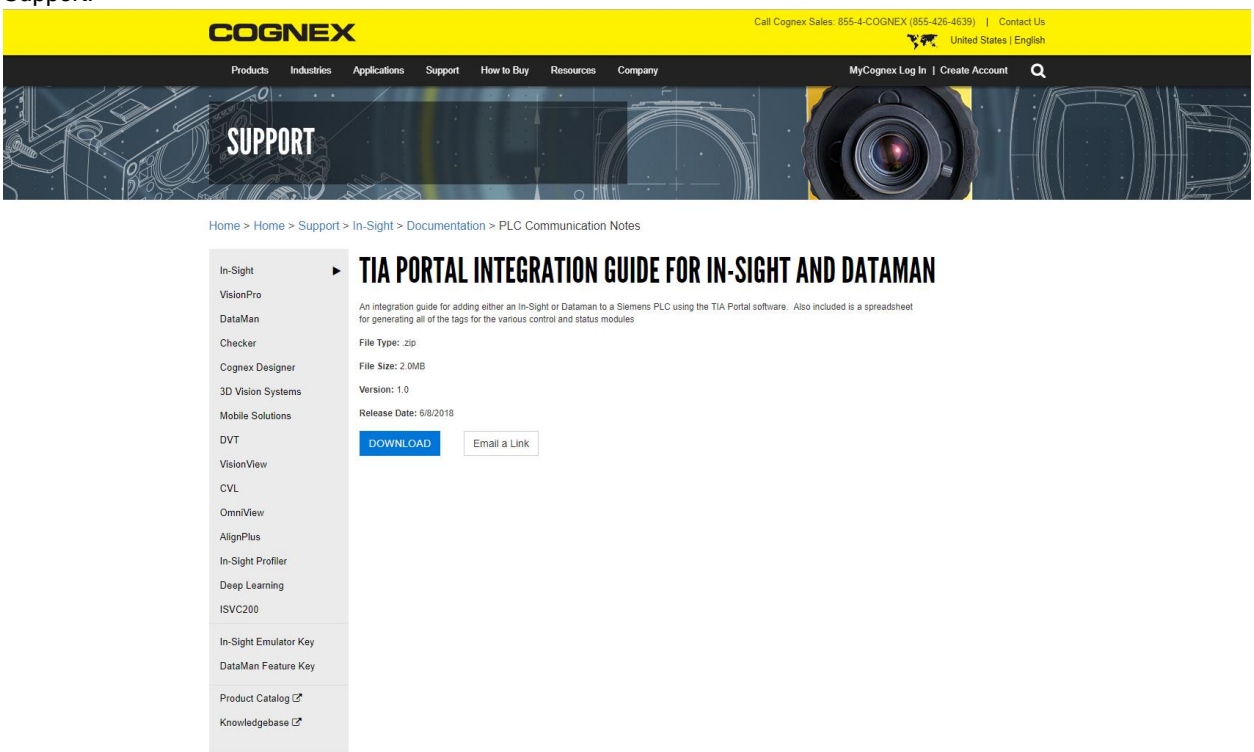


3. Check if the DataMan's firmware is up-to-date by clicking, in the Setup Tool, **View** and then **System Info**.

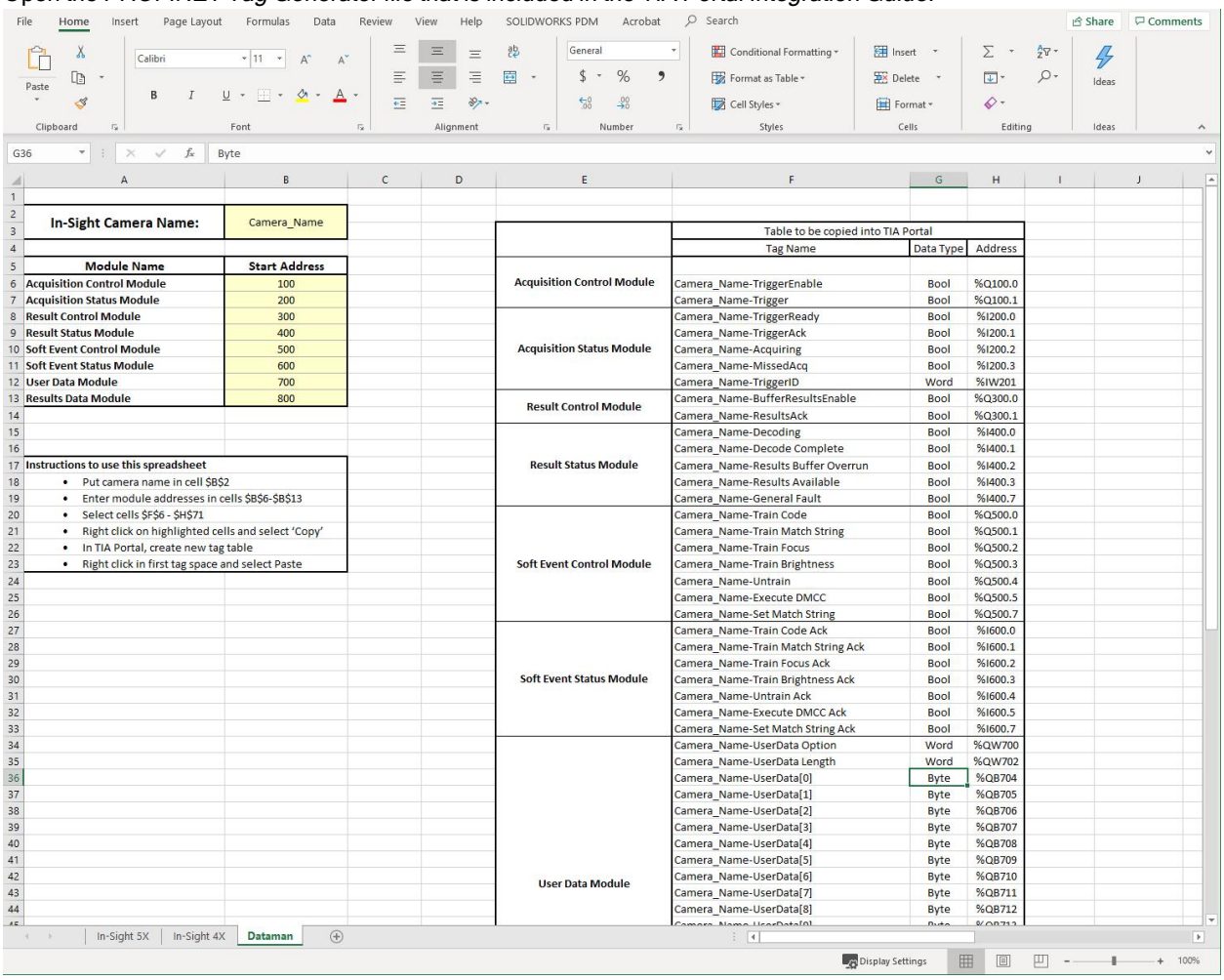
4. Make sure that in the Setup Tool, **PROFINET** is enabled.



5. Download the *TIA Portal Integration Guide* from https://support.cognex.com. Search for *TIA Portal* under In-Sight Support.



6. Open the *PROFINET Tag Generator* file that is included in the *TIA Portal Integration Guide*.



In-Sight Camera Name:	Camera_Name
Module Name	**Start Address**
Acquisition Control Module	100
Acquisition Status Module	200
Result Control Module	300
Result Status Module	400
Soft Event Control Module	500
Soft Event Status Module	600
User Data Module	700
Results Data Module	800

Instructions to use this spreadsheet

- Put camera name in cell B2
- Enter module addresses in cells B6-B13
- Select cells F6 - H71
- Right click on highlighted cells and select 'Copy'
- In TIA Portal, create new tag table
- Right click in first tag space and select Paste

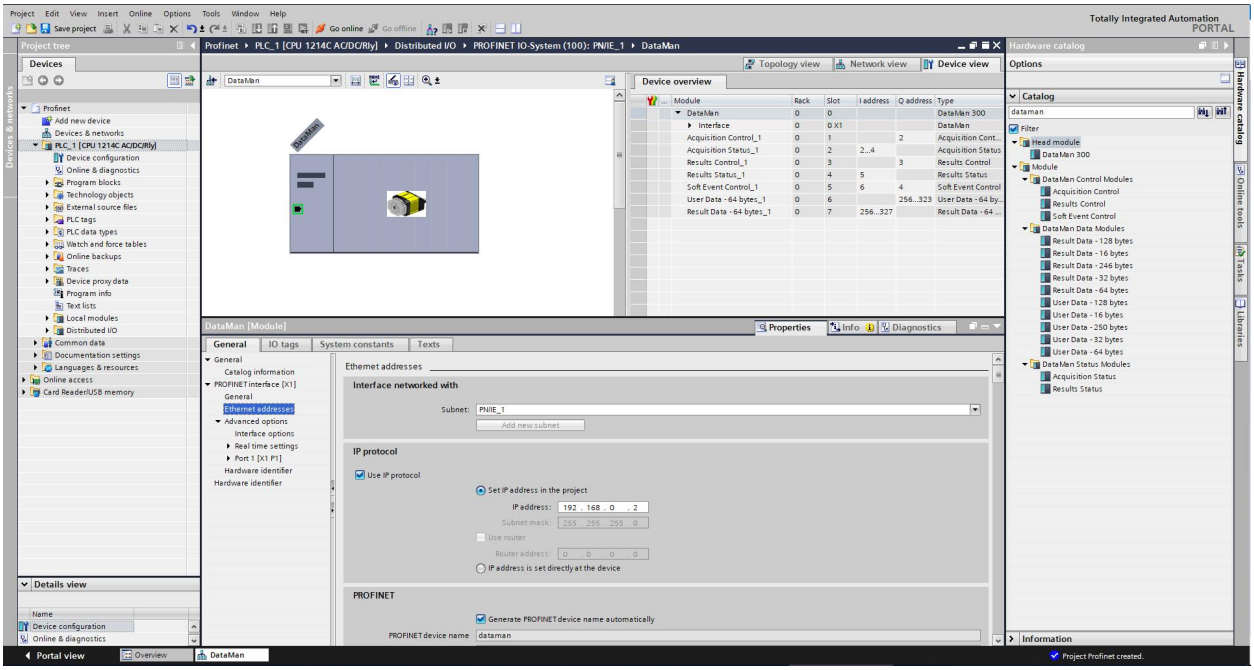
	Table to be copied into TIA Portal		
	Tag Name	Data Type	Address
Acquisition Control Module	Camera_Name-TriggerEnable	Bool	%Q100.0
	Camera_Name-Trigger	Bool	%Q100.1
Acquisition Status Module	Camera_Name-TriggerReady	Bool	%I200.0
	Camera_Name-TriggerAck	Bool	%I200.1
	Camera_Name-Acquiring	Bool	%I200.2
	Camera_Name-MissedAcq	Bool	%I200.3
	Camera_Name-TriggerID	Word	%IW201
Result Control Module	Camera_Name-BufferResultsEnable	Bool	%Q300.0
	Camera_Name-ResultsAck	Bool	%Q300.1
Result Status Module	Camera_Name-Decoding	Bool	%I400.0
	Camera_Name-Decode Complete	Bool	%I400.1
	Camera_Name-Results Buffer Overrun	Bool	%I400.2
	Camera_Name-Results Available	Bool	%I400.3
	Camera_Name-General Fault	Bool	%I400.7
Soft Event Control Module	Camera_Name-Train Code	Bool	%Q500.0
	Camera_Name-Train Match String	Bool	%Q500.1
	Camera_Name-Train Focus	Bool	%Q500.2
	Camera_Name-Train Brightness	Bool	%Q500.3
	Camera_Name-Untrain	Bool	%Q500.4
	Camera_Name-Execute DMCC	Bool	%Q500.5
	Camera_Name-Set Match String	Bool	%Q500.7
Soft Event Status Module	Camera_Name-Train Code Ack	Bool	%I600.0
	Camera_Name-Train Match String Ack	Bool	%I600.1
	Camera_Name-Train Focus Ack	Bool	%I600.2
	Camera_Name-Train Brightness Ack	Bool	%I600.3
	Camera_Name-Untrain Ack	Bool	%I600.4
	Camera_Name-Execute DMCC Ack	Bool	%I600.5
	Camera_Name-Set Match String Ack	Bool	%I600.7
User Data Module	Camera_Name-UserData Option	Word	%QW700
	Camera_Name-UserData Length	Word	%QW702
	Camera_Name-UserData[0]	Byte	%QB704
	Camera_Name-UserData[1]	Byte	%QB705
	Camera_Name-UserData[2]	Byte	%QB706
	Camera_Name-UserData[3]	Byte	%QB707
	Camera_Name-UserData[4]	Byte	%QB708
	Camera_Name-UserData[5]	Byte	%QB709
	Camera_Name-UserData[6]	Byte	%QB710
	Camera_Name-UserData[7]	Byte	%QB711
	Camera_Name-UserData[8]	Byte	%QB712

7. From the Windows **Start** menu, launch the SIMATIC Manager or TIA Portal.

PROFINET Object Model

The PROFINET implementation on DataMan consists of seven I/O modules:

1. Acquisition Control Module
2. Acquisition Status Module
3. Results Control Module
4. Results Status Module
5. SoftEvent Control Module
6. User Data Module
7. Result Data Module



Acquisition Control Module

Controls image acquisition. This module consists of data sent from the PLC to the DataMan device.

Slot number: 1

Total Module size: 1 byte

Bit	Name	Description
0	Trigger Enable	Setting this bit enables triggering via PROFINET. Clearing this bit disables triggering.
1	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.
2-7	Reserved	Reserved for future use

Acquisition Status Module

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

Slot number: 2

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” is set and the device is ready to accept a new trigger.
1	Trigger Ack	Indicates that the DataMan has received a new Trigger. This bit remains True as long as the “Trigger” bit remains True (that is, it is interlocked with the Trigger bit).
2	Acquiring	Indicates that the reader is in the process of acquiring an image.

3	Missed Ack	Indicates that the DataMan 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	Trigger 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 is returned in ResultID of the result data.

Note: The Missed Ack bit in the Acquisition Status Register will only be set if an acquisition triggered from the Acquisition Control Module could not get executed.

Results Control Module

Controls the processing of result data. This module consists of data sent from the PLC to the DataMan device.

Slot number: 3

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 DataMan will respond to the acknowledge 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. The reader can buffer up to 50 and the base station can buffer up to 500 sets of read results.
1	Results Ack	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. If result buffering is enabled, the next set of result data will be pulled from the queue and “Result Available” will again be set True.
2-7	Reserved	Reserved for future use.

Results Status Module

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

Slot number: 4

Total Module size: 1 byte

Bit	Name	Description
0	Decoding	Indicates that the DataMan is decoding an acquired image.[1]
1	Decode Complete	Bit is toggled on the completion of a decode operation when the new results are made available (0 -> 1 or 1 -> 0).
2	Result Buffer Overrun	Indicates that the DataMan has discarded a set of read results because the results queue is full. Cleared when the next set of results are successfully queued.

[1] The value of the **Decoding** bit is identical to the value of the **Acquiring** bit. Both bits are true if an acquisition, or series of acquisitions of the same trigger, is in progress.

3	Results Available	Indicates that a new set of read results are available, that is, the contents of the Result Data module are valid. Cleared when the results are acknowledged.
4-6	Reserved	Reserved for future use
7	General Fault	Indicates that a fault has occurred. Currently only used by Soft Event operations.

SoftEvent Control Module

Used to initiate a SoftEvent and receive acknowledgment of completion. Note that this is a bi-directional I/O module. Module data sent from the PLC initiates the SoftEvent. Module data sent by the DataMan device acknowledges completion.

Slot number: 5

Total Module size: 1 byte (input) and 1 byte (output)

Data written from the PLC to DataMan:

Bit	Name	Description
0	Train Code	Bit transition from 0 -> 1 causes the train code operation to be invoked.
1	Train Match String	Bit transition from 0 -> 1 causes the train match string operation to be invoked.
2	Train Focus	Bit transition from 0 -> 1 causes the train focus operation to be invoked.
3	Train Brightness	Bit transition from 0 -> 1 causes the train brightness operation to be invoked.
4	Untrain	Bit transition from 0 -> 1 causes the untrain operation to be invoked.
5	Reserved	Reserved for future use
6	Execute DMCC	Bit transition from 0 -> 1 causes the DMCC operation to be invoked. Note that a valid DMCC command string must first be placed in “User Data” before invoking this event.
7	Set Match String	Bit transition from 0 -> 1 causes the set match string operation to be invoked. Note that match string data must first be placed in “User Data” before invoking this event.

Data written from the DataMan to PLC:

Bit	Name	Description
0	Train Code Ack	Indicates that the “Train Code” operation is complete
1	Train Match String Ack	Indicates that the “Train Match String” operation is complete
2	Train Focus Ack	Indicates that the “Train Focus” operation is complete
3	Train Brightness Ack	Indicates that the “Train Brightness” operation is complete
4	Untrain Ack	Indicates that the “Untrain” operation is complete
5	Reserved	Reserved for future use
6	Execute DMCC Ack	Indicates that the “Execute DMCC” operation is complete
7	Set Match String Ack	Indicates that the “Set Match String” operation is complete

User Data Module

Data sent from a PLC to a DataMan to support acquisition, decode and other special operations. Currently this module is only used to support the “Execute DMCC” and “Set Match String” SoftEvents.

Note: There are 5 versions of the User Data module. Only one instance can be configured for use in a given application. The “User Data Option” and “User Data Length” fields are the same for each module. The “User Data” field varies in size based on the selected module. Choose the module which is large enough to exchange the amount of data your application requires.

Slot number: 6

Total Module size:

4 + 16 (16 bytes of User Data)

4 + 32 (32 bytes of User Data)

4 + 64 (64 bytes of User Data)

4 + 128 (128 bytes of User Data)

4 + 250 (250 bytes of User Data)

<table>
<tr><th>Byte</th><th>Name</th><th>Description</th></tr>
<tr><td>0-1</td><td>User Data Option</td><td>Optional user data information (16-bit Integer). Used by the “Set Match String” SoftEvent to assign the match string to the specified symbology. Depending on your PLC software, you can move an integer value or a binary to this field:
<table>
<tr><th>Integer</th><th>Binary</th><th>Description</th></tr>
<tr><td>0</td><td>0000 0000 0000 0000</td><td>Assign string to all smybologies.</td></tr>
<tr><td>1</td><td>0000 0000 0000 0001</td><td>Assign string to 2D codes.</td></tr>
<tr><td>2</td><td>0000 0000 0000 0010</td><td>Assign string to QR codes.</td></tr>
<tr><td>3</td><td>0000 0000 0000 0011</td><td>Assign string to 1D/stacked/postal codes.</td></tr>
</table></td></tr>
<tr><td>2-3</td><td>User Data Length</td><td>Number of bytes of valid data actually contained in the “User Data” field (16-bit Integer).</td></tr>
<tr><td>4...</td><td>User Data</td><td>Data sent from the PLC to the DataMan to support acquisition, decoding, and other special operations (array of bytes).</td></tr>
</table>

Result Data Module

Read result data sent from a DataMan to a PLC.

Look into the Result Data Module for non-PROFINET related data if there is an acquisition problem on the reader. The "Result Code" part contains information about trigger overruns or buffer overflows that have occurred on the reader.

Note: There are 5 versions of the Result Data module. Only a single instance can be configured for use in a given application. The “Result ID”, “Result Code”, “Result Extended” and “Result Length” fields are the same for each module. The “Result Data” field varies in size based on the selected module. Choose the module which is large enough to exchange the amount of result data your application requires.

Slot number: 7

Total Module size:

8 + 16 (16 bytes of Result Data)

8 + 32 (32 bytes of Result Data)

8 + 64 (64 bytes of Result Data)

8 + 128 (128 bytes of Result Data)

8 + 246 (246 bytes of Result Data)

Byte	Name	Description
0-1	Result ID	The value of the “Trigger ID” when the trigger that generated these results was issued. Used to match up triggers with corresponding result data (16-bit Integer).
2-3	Result Code	Indicates the success or failure of the read that produced these results (16-bit Integer). Bit 0,1=read, 0=no read Bit 1,1=validated, 0=not validated (or validation not in use) Bit 2,1=verified, 0=not verified (or verification not in use) Bit 3,1=acquisition trigger overrun Bit 4,1=acquisition buffer overflow Bit 5: reserved Bit 6: 1=result is a DMCC response, 0=result is a read result. For more information, see *DMCC Result Feedback from PLC* on page 52. Bits 7-15 reserved
4-5	Result Extended	Currently unused (16-bit Integer).
6-7	Result Length	Actual number of bytes of read data contained in the “Result Data” field (16-bit Integer).
8...	Result Data	Decoded read result data (array of bytes)

PROFINET Operation

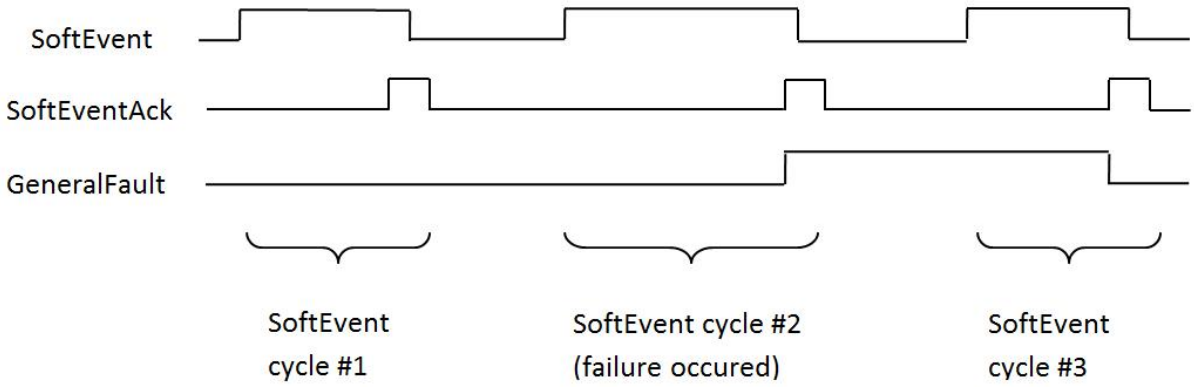
SoftEvents

SoftEvents act as “virtual” inputs. When the value of a SoftEvent changes from 0 -> 1 the action associated with the event will be executed. When the action completes the corresponding SoftEventAck bit will change from 0 -> 1 to signal completion. The acknowledge bit will change back to 0 when the corresponding SoftEvent bit is set back to 0.

The “ExecuteDMCC” and “SetMatchString” SoftEvent actions require user supplied data. This data must be written to the UserData and UserDataLength area of the UserData Module prior to invoking the SoftEvent. Only one SoftEvent can be invoked at a time because both of these SoftEvents depend on the UserData.

General Fault Indicator

When a communication related fault occurs the “GeneralFault” bit will change from 0 -> 1. Currently the only fault conditions supported are SoftEvent operations. If a SoftEvent operation fails, the fault bit will be set. The fault bit will remain set until the next SoftEvent operation or until triggering is disabled and again re-enabled.



Acquisition Sequence

DataMan can be triggered to acquire images by several methods. It can be done explicitly by manipulating the Trigger bit of the Acquisition Control Module, it can be triggered by external hardwired input, and it can be triggered via DMCC. This section describes manipulating the Acquisition Control Module bits.

On startup the “Trigger Enable” bit 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.

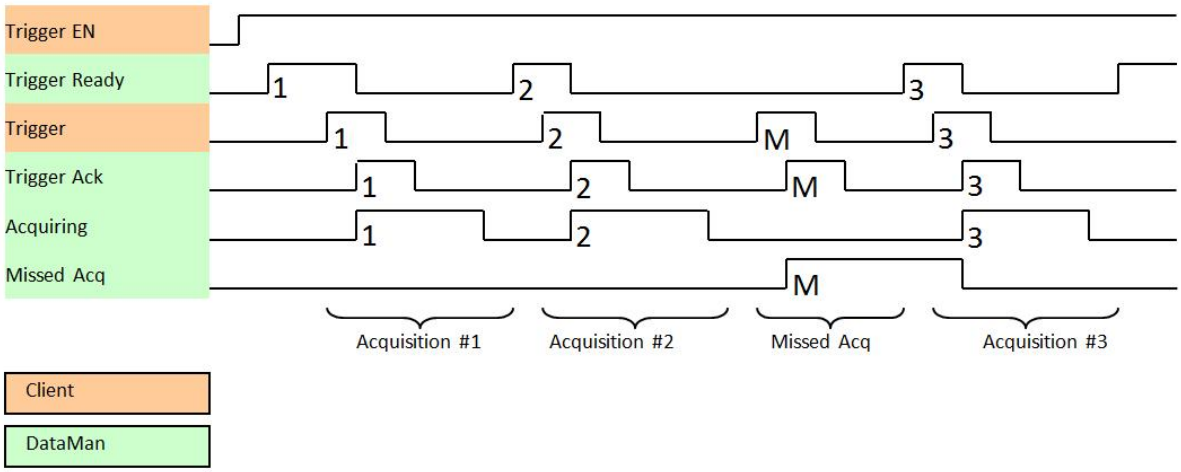
While the Trigger Ready bit is True, each time the reader detects the “Trigger” bit change from 0 to 1, it initiates an image acquisition. Make sure that the client (PLC) holds the bit in the new state until that same state value is seen back in the Trigger Ack bit. This is a necessary handshake to guarantee that the reader detects the change.

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. The Trigger Ready bit is again set to True once the device is ready to begin a new image acquisition.

The reader allows acquisitions to overlap with the decoding of previous acquisitions. The Trigger Ready bit is set high after the acquisition is complete, while decoding may still be in progress. The Decoding bit is deprecated and only mirrors the behavior of the Acquiring bit. If trigger queuing is active or other trigger sources can interfere, the Trigger Ready bit may not be reliable. Conflicting trigger overruns are reported in the Result Code bit of the Result Data module.

To force a reset of the trigger mechanism, set the Trigger Enable bit to False, until the Trigger Ready bit is 0. Then, Trigger Enable can be set to True to re-enable acquisition.

As a special case, an acquisition can be cancelled by clearing the Trigger signal before the read operation is complete. This allows cancelling reads in Continuous and Manual mode if no code is in the field of view. To ensure that a read is not unintentionally cancelled, it is advised that the PLC hold the Trigger signal True until both TriggerAck and ResultsAvailable are True or DecodeComplete toggles state.



Decode / Result Sequence

After an image is acquired, it is decoded. When the decoding is complete and a result is ready to be delivered, the reader toggles the *DecodeComplete* bit of the *DecodeStatusRegister*.

Decode results are reported asynchronously to the Result Status module. If BufferResultsEnable is set to False, then the decode results are immediately placed into DecodeResults, and the Results Available bit is set to True.

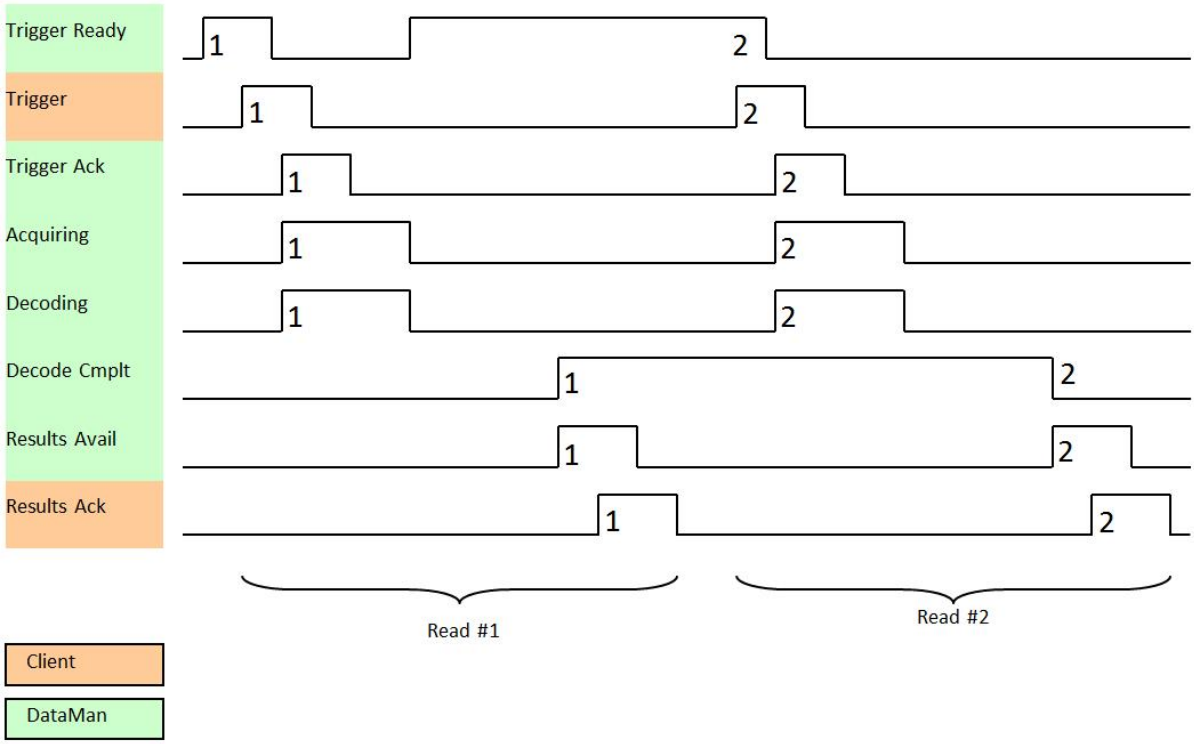
> **Note**: The only way to ensure that you are not losing results is to use BufferResults. Make sure that your application can read each DecodeResult in time if you do not use BufferResultsEnable.

If the Results Buffer Enable bit is set to True, the new results are queued. The earlier decode results remain in the Results Module until the client acknowledges them by setting the “Results Ack” bit to True. After the Results Available bit is cleared,

make sure that the client sets the Results Ack bit back to False to allow the next queued results to be placed into the Results Module. This is a necessary handshake to ensure the results are received by the DataMan client (PLC), even if short gaps occur between results.

Behavior of DecodeStatusRegister

Bit	Bit Name	Results if Buffering Disabled	Results if Buffering Enabled
0	Decoding	Set when acquiring and decoding an image. The value of the Decoding bit is always the same as the Acquiring bit.	Set when acquiring and decoding an image. The value of the Decoding bit is always the same as the Acquiring bit.
1	Decode Complete	Toggled on completion of an image decode.	Toggled on completion of an image decode.
2	Results Buffer Overflow	Remains set to zero.	Set when decode results could not be queued because the client failed to acknowledge a previous result. Cleared when the decode result is successfully queued.
3	Results Available	Set when new results are placed in the Results Module. Stays set until the results are acknowledged by setting Results Ack to 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

The Results Buffering feature allows a queue for decode results. When enabled, the feature queues a finite number of result data sets until the PLC (client) has time to read them. This smooths out data flow if the PLC slows down for short periods or if there are surges in read activity.

If the feature is enabled, the reader overlaps acquisition and decode operations. This can lead to faster over all trigger rates. When reads occur faster than results can be sent out (the queue is full), the most recent results are discarded until space becomes available in the results queue.

If the feature is disabled, the more recent results overwrite the earlier results. The most recent results are kept. Earlier results are discarded if the PLC does not read them fast enough.

If the queue overflows while the feature is disabled:

- The difference between the TriggerID and ResultID values becomes greater than 1. This difference represents the number of reads that occurred but could not be queued because the queue was full.
- The number of lost reads equals TriggerID - ResultID - 1.
- After the next read, the ResultID value returns to the typical operating value of TriggerID - 1.

Result Fragmentation

The Result Fragmentation feature splits decode result strings into fragments if the strings are too large for the Results Data field. The fragments are then pushed to the buffer like normal results. Then, the PLC reads the fragments one by one.

Result Fragmentation depends on Result Buffering. This means that results are only fragmented if Result Buffering is enabled.

Note: For PROFINET, the Result Fragmentation feature is automatically enabled when you enable Results Buffering.

Enabling Results Buffering and Result Fragmentation for PROFINET

To enable results buffering, set the Results Buffer Enable bit in the Results Control Module to 1. When the *ResultsAvailable* bit in the Results Status Module is set to 1, follow the instructions below depending on the length of the Result Length.

Result Length Larger Than Result Data Module Size

If the Result Length is larger than the Result Data Module size set in the PLC, then there are additional fragments of the decode string to be sent to the PLC. Proceed with the following steps:

1. Read the initial data, which is the first fragment of the decode string equal to the Result Data Module size set in the PLC.
2. Set the *ResultsAck* bit in the Results Control Module to 1 to acknowledge reading. As a result, the *ResultsAvailable* bit in the Results Status Module goes to 0.
3. Set the *ResultsAck* bit in the control object to 0 again. As a result:
    - The *ResultAvailable* bit in the Results Status Module goes back to 1.
    - The *Result Length* decreases to show the remaining bytes to be read.
    - The *ResultID* in the Inspection Result Data Module stays the same because the fragments are all part of the same decode string.
4. Repeat the steps above until the Result Length is smaller than the size of the Decode Results Module.

Result Length Smaller Than Result Data Module Size

If the Result Length is smaller than the Result Data Module size set in the PLC, proceed with the following steps:

1. Read the data, which is the first set of buffered results.
2. Set the *ResultsAck* bit in the Control Module to 1 to acknowledge reading. As a result, the *ResultsAvailable* bit in the Status Module goes to 0.

3. Set the *ResultsAck* bit in the Results Control Module to 0 again. As a result:
    - The *ResultAvailable* bit in the Results Status Module goes back to 1.
    - The *ResultID* in the Inspection Result Data Module increments, indicating that a new decode string has been sent to the PLC.
    - The *Result Length* now contains the length of the remaining data of the full code: last length - the Result Data Module size set in the PLC.
4. Read the remaining data.
5. If not finished, repeat the handshaking between *ResultsAvailable* and *ResultsAck* to acknowledge the remaining results, until all results have been read.

Operation of Result Fragmentation

The PLC does not automatically reassemble the result string. Once the PLC reads a result fragment, you have to copy the fragment to a new user-defined variable. After this, you have to set and clear the ResultAck bit, which updates the Decode Results with the next fragment. This process continues until all fragments have been sent to the PLC.

In contrast, if Results Buffering is disabled, large decode result strings are cut off to fit into the Results Data field, which means that the end of the string gets lost.

Identifying Decode Result Chunks

Decode result fragments look like normal results. To identify result fragments, look at the Result ID:

- The Result ID of the result fragments are the same.
- The Result ID of the last result fragment is different from the Result ID of the next result.

Independent of whether results fragmentation is enabled, the Result Length always represents the number of characters to be read from the Results Data plus the remaining characters that did not fit into the Results Data buffer. With results fragmentation enabled, the remaining characters will be found in follow-up fragments, the remaining characters are discarded.

You can identify the first and last fragments of a result as follows:

- The first fragment has a different Result ID from the previous Result ID.
- The Result Length of the last fragment is lower or equal to the Result Data buffer size.
- All fragments of a result share the same Result ID.

Maximum Result Data Length

Without results fragmentation, result data is truncated to fit into the results data buffer size. Depending on the protocol, the result buffer size can be fixed or configurable.

The Result Fragmentation feature allows processing of up to 50 times longer results, depending on the results buffering queue size, with an absolute limit of 65535 characters.

The maximum number of result characters for PROFINET is 246 bytes(using the largest result data module).

DMCC Result Feedback from PLC

The DMCC Result Feedback feature allows the reader to send DMCC responses back to the PLC.

By default, this feature is off. To enable the feature, use the `FFP.DMCC-RESULTS` DMCC command. Changing this configuration does not require you to reboot or reconnect reader.

When you enable the feature, the reader sends DMCC responses to the PLC using the same mechanism it uses to send read results. To distinguish DMCC results from read results, bit 6 is of the ResultCode field is set to 1. The following Result Data Module fields contain DMCC information instead of read results:

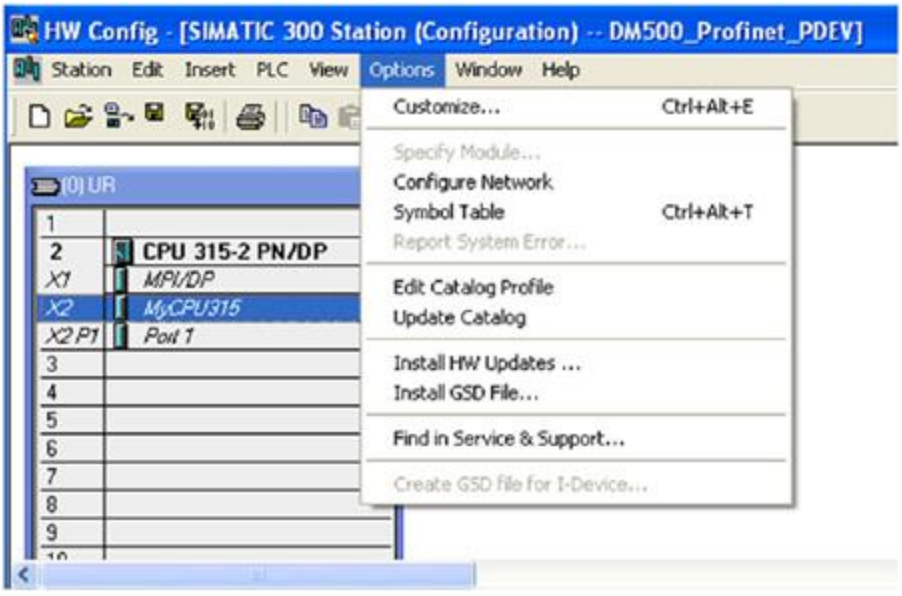
Byte	Name	Description
0-1	ResultID	Contains a DMCC result code: • 0: no error • 100: unidentified error • 101: invalid command • 102: invalid parameter • 103: incorrect checksum • 104: parameter rejected/altered due to reader state • 105: reader unavailable (offline)
2-3	ResultCode	Bit 6 is set to 1 to indicate a DMCC response.
6-7	ResultLength	Actual number of bytes of DMCC result data contained in the *ResultData* field (16-bit Integer).
8...	ResultData	DMCC result data (array of bytes).

You can use DMCC Result Feedback in combination with Results Fragmentation.

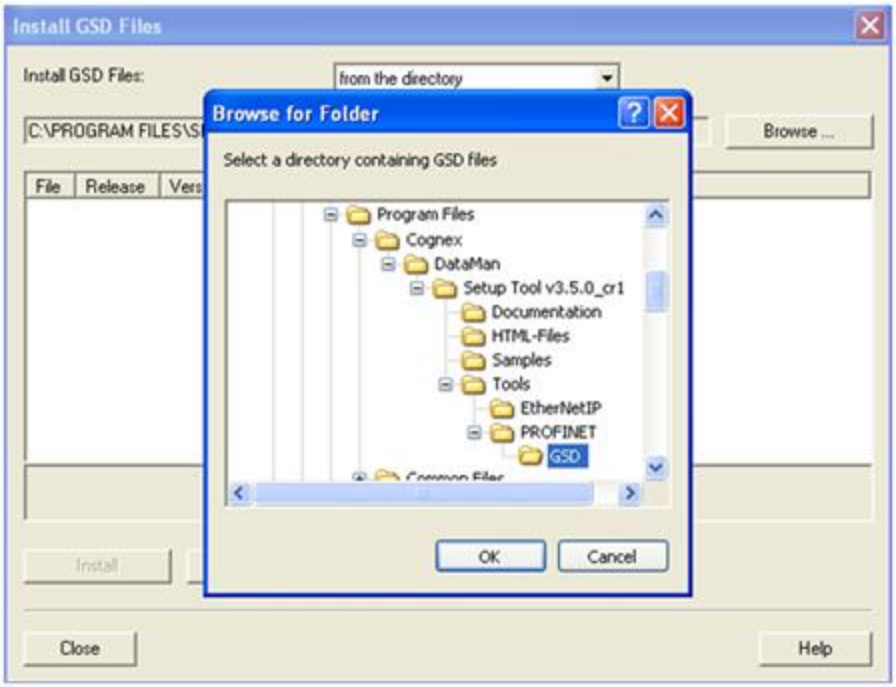
SIMATIC Examples

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

1. If you already have a project, click **Cancel** to skip past the New Project wizard. Otherwise, let the wizard guide you through creating a new project.
2. In the opened project, double-click the Hardware icon to open the **HW Config** dialog. From the main menu, select **Options** -> **Install GSD File...**.

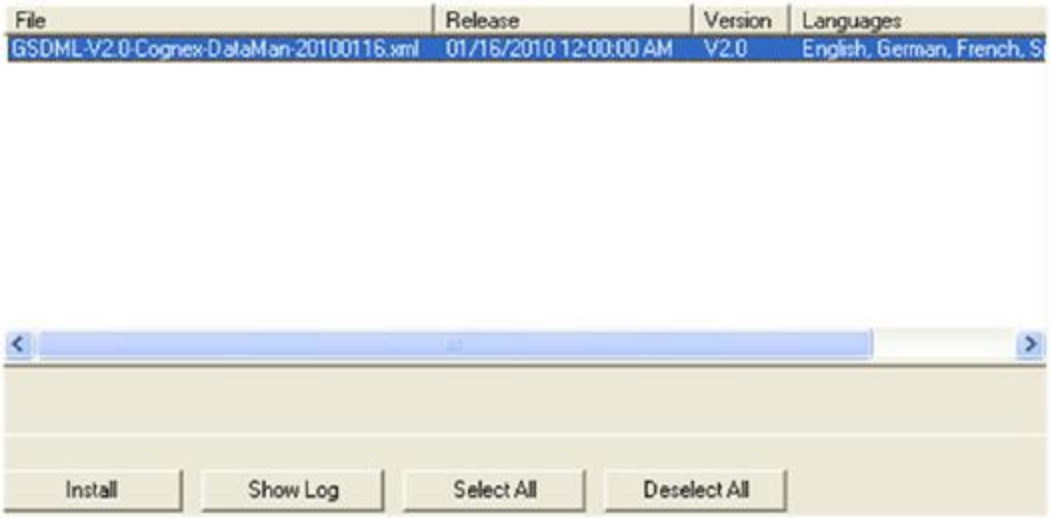


3. Browse to the installation folder of the GSD file (or the folder where you saved the GSD file if you downloaded it from the web).



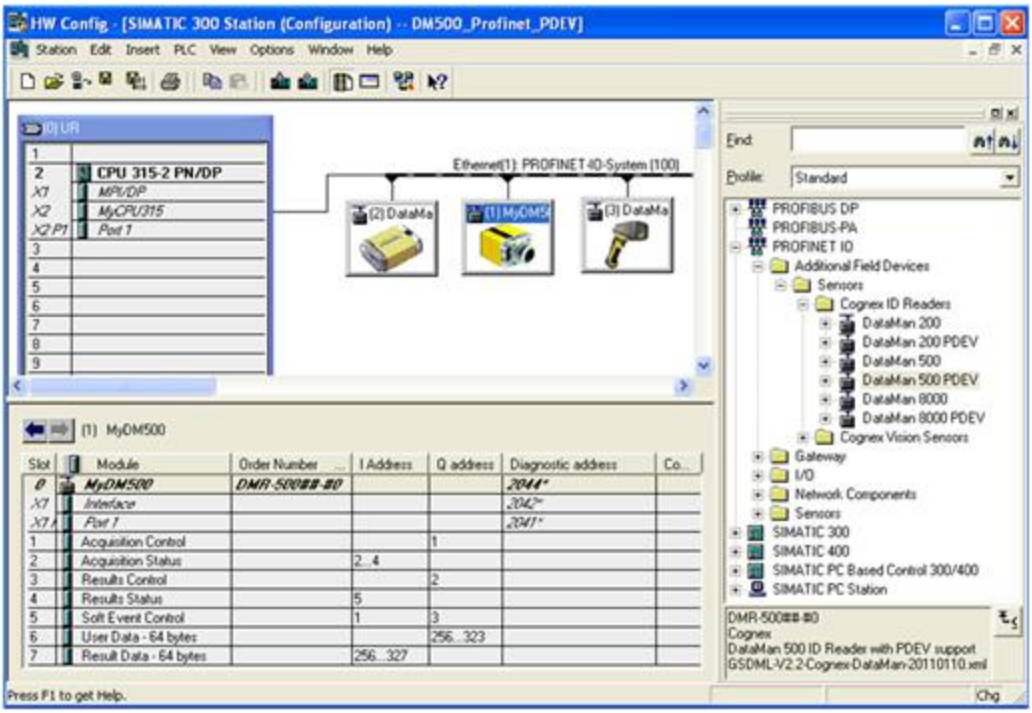
4. Select the GSD file you wish to install and follow the displayed instructions to complete the installation.

> **Note**: If there is more than one GSD file in the list, and you are unsure which to install, choose the one with the most recent date.



5. Add your DataMan device to your project. This makes the DataMan available in the Hardware Catalog. Launch the SIMATIC Hardware Config tool.
6. In the main menu, select View -> Catalog.
7. The catalog is displayed. Expand the **PROFINET IO** tree to the **Cognex ID Readers** node.

8. Press and hold the left mouse button to drag the DataMan reader and drop it on the PROFINET IO network symbol in the left pane.



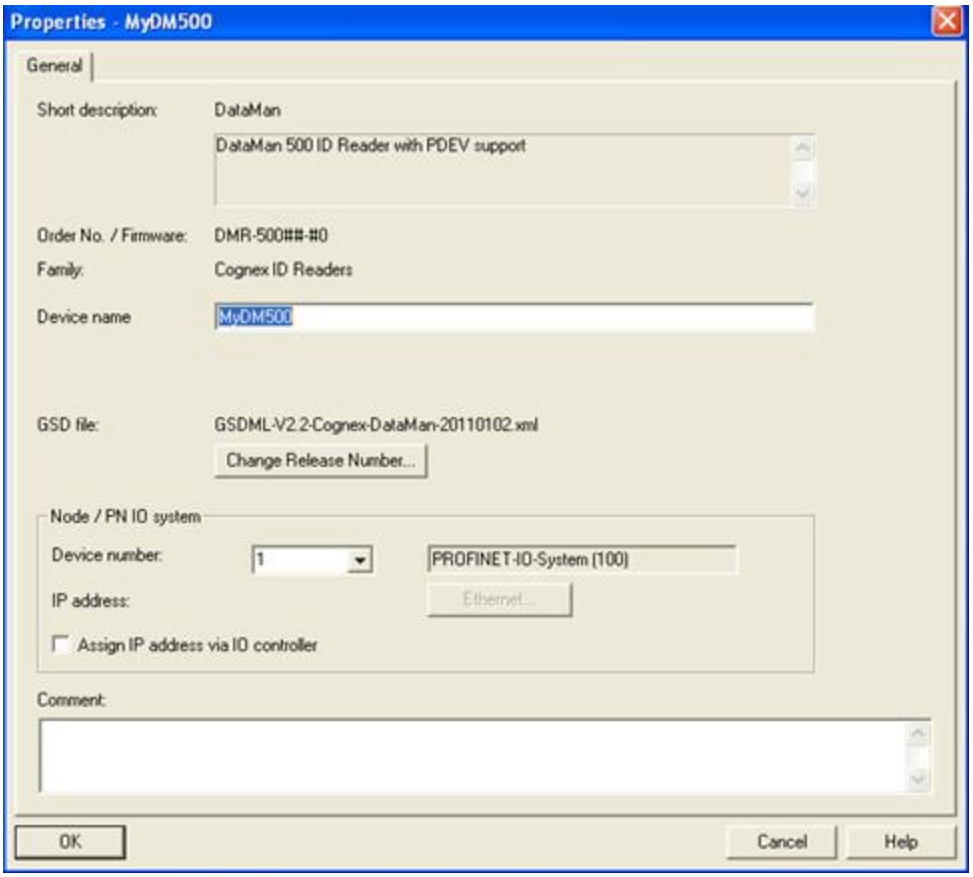
The HW Config tool automatically maps the DataMan I/O modules into the memory space.

> **Note**: By default, the 64 byte User Data and 64 byte Result Data Modules are inserted. There are multiple sizes available for both of these modules. To optimize performance, use the module size that most closely matches the actual data requirements of your application. You can change the module by deleting the one in the table and inserting the appropriate sized module from the catalog.

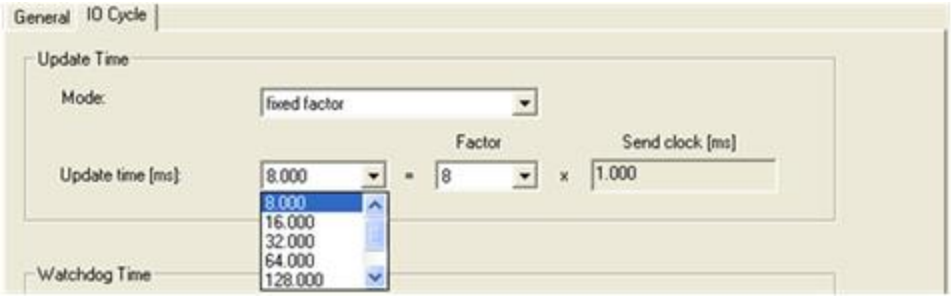
9. Right-click the DataMan icon and select **Object Properties...**.

10. Give the reader a name. This must match the name of your actual DataMan reader. The name must be unique and follow DNS naming conventions. For details, see the SIMATIC Software help.

11. If your DataMan reader is configured to use its own static IP, uncheck **Assign IP address via IO controller**. As an alternative, if you wish the PLC to assign an IP address, select the Ethernet button and configure the appropriate address.

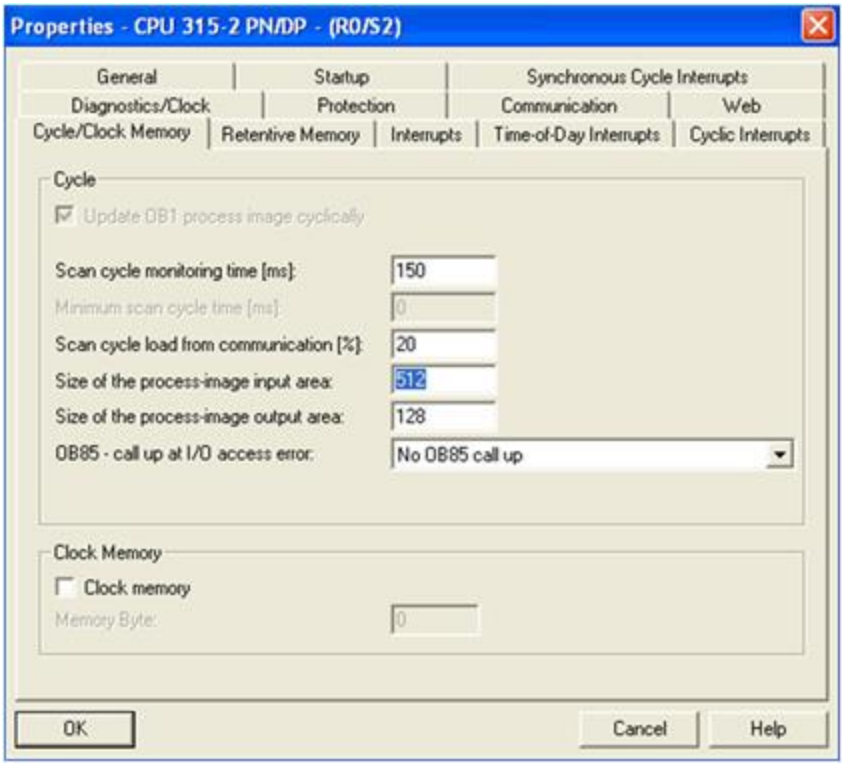


12. In the **IO Cycle** tab, select the appropriate cyclic update rate for your application.



13. By default, the SIMATIC software maps the User Data and Result Data Modules to offset 256. This is outside of the default process image area size of 128. That is, 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.

Symbol Table

It is recommended that you define symbols for the DataMan I/O module elements to make the code much easier to read and reduce mistakes. This sample table shows symbols defined for a typical instance of a DataMan reader. It is possible that DataMan I/O modules are at different addresses in your project. Make sure to adjust your symbol definitions based on the specific offsets of the I/O modules.

Symbol Editor - [S7 Program(3) (Symbols) -- DM200_ResultString\SIMATIC 300 ...

Symbol Table Edit Insert View Options Window Help

All Symbols

	Status	Symbol	Address	Data type	Comment
1		Reader_TriggerEnable	Q 1.0	BOOL	
2		Reader_Trigger	Q 1.1	BOOL	
3		Reader_TriggerReady	I 2.0	BOOL	
4		Reader_TriggerAck	I 2.1	BOOL	
5		Reader_Acquiring	I 2.2	BOOL	
6		Reader_MissedAck	I 2.3	BOOL	
7		Reader_TriggerID	MW 3	WORD	
8		Reader_BufferEnable	Q 2.0	BOOL	
9		Reader_ResultsAck	Q 2.1	BOOL	
10		Reader_Decoding	I 5.0	BOOL	
11		Reader_DecodeComplete	I 5.1	BOOL	
12		Reader_ResultsOverrun	I 5.2	BOOL	
13		Reader_ResultsAvailable	I 5.3	BOOL	
14		Reader_GeneralFault	I 5.7	BOOL	
15		Reader_TrainCode	Q 3.0	BOOL	
16		Reader_TrainMatchString	Q 3.1	BOOL	
17		Reader_TrainFocus	Q 3.2	BOOL	
18		Reader_TrainBrightness	Q 3.3	BOOL	
19		Reader_UnTrain	Q 3.4	BOOL	
20		Reader_ExecuteDmcc	Q 3.6	BOOL	
21		Reader_SetMatchString	Q 3.7	BOOL	
22		Reader_TrainCodeAck	I 1.0	BOOL	
23		Reader_TrainMatchStrAck	I 1.1	BOOL	
24		Reader_TrainFocusAck	I 1.2	BOOL	
25		Reader_TrainBrightAck	I 1.3	BOOL	
26		Reader_UnTrainAck	I 1.4	BOOL	
27		Reader_ExecuteDmccAck	I 1.6	BOOL	
28		Reader_SetMatchStringAck	I 1.7	BOOL	
29		Reader_UserDataOption	QW 256	WORD	
30		Reader_UserDataLength	QW 258	WORD	
31		Reader_ResultID	MW 256	WORD	
32		Reader_ResultCode	MW 258	WORD	
33		Reader_ResultExtended	MW 260	WORD	
34		Reader_Resultlength	MW 262	WORD	

Press F1 to get Help. NUM

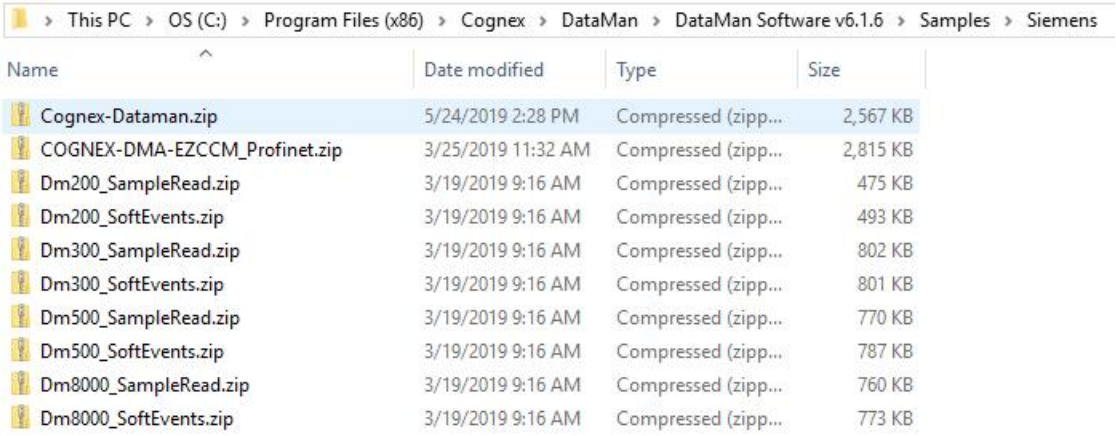
Trigger and Get Results

Run the sample program “DM200_SampleRead” for the complete example program.

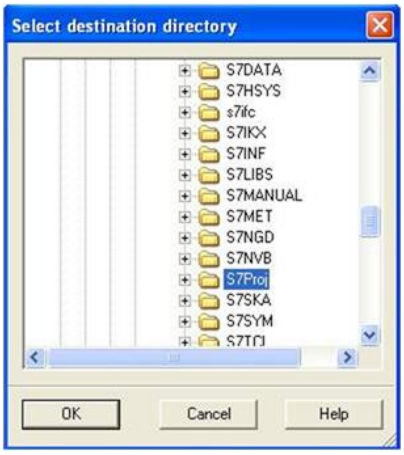
Note: This sample can be used with any PROFINET enabled DataMan reader.

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.



5. Save the project on your PC.



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

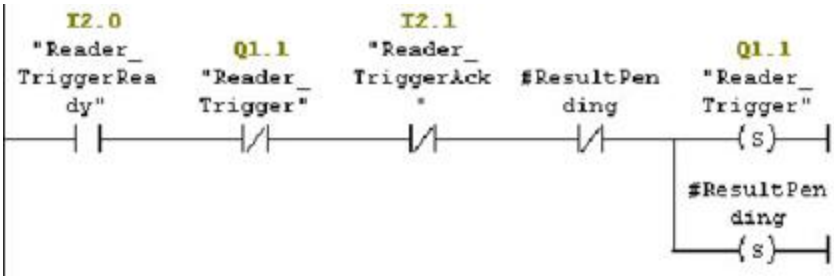
Reduced to the basics, the process of reading and retrieving results consists of the following:

1. Define an area in your application to save read results. There are many options regarding how and where result data can be stored. In this example, a Data Block (DB) is defined containing the fields of the Result Data module that are relevant for our application.

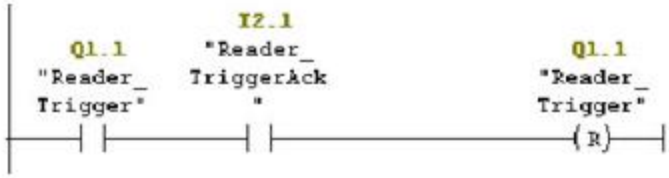
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	ID	INT	0	ID of this read result
+2.0	Flags	INT	0	Flags indicating success or failure
+4.0	Length	INT	0	Number of data bytes in the array Value[]
+6.0	Value	ARRAY[1..63]		Code value read
*1.0		CHAR		
=70.0		END_STRUCT		

2. Enable the reader.

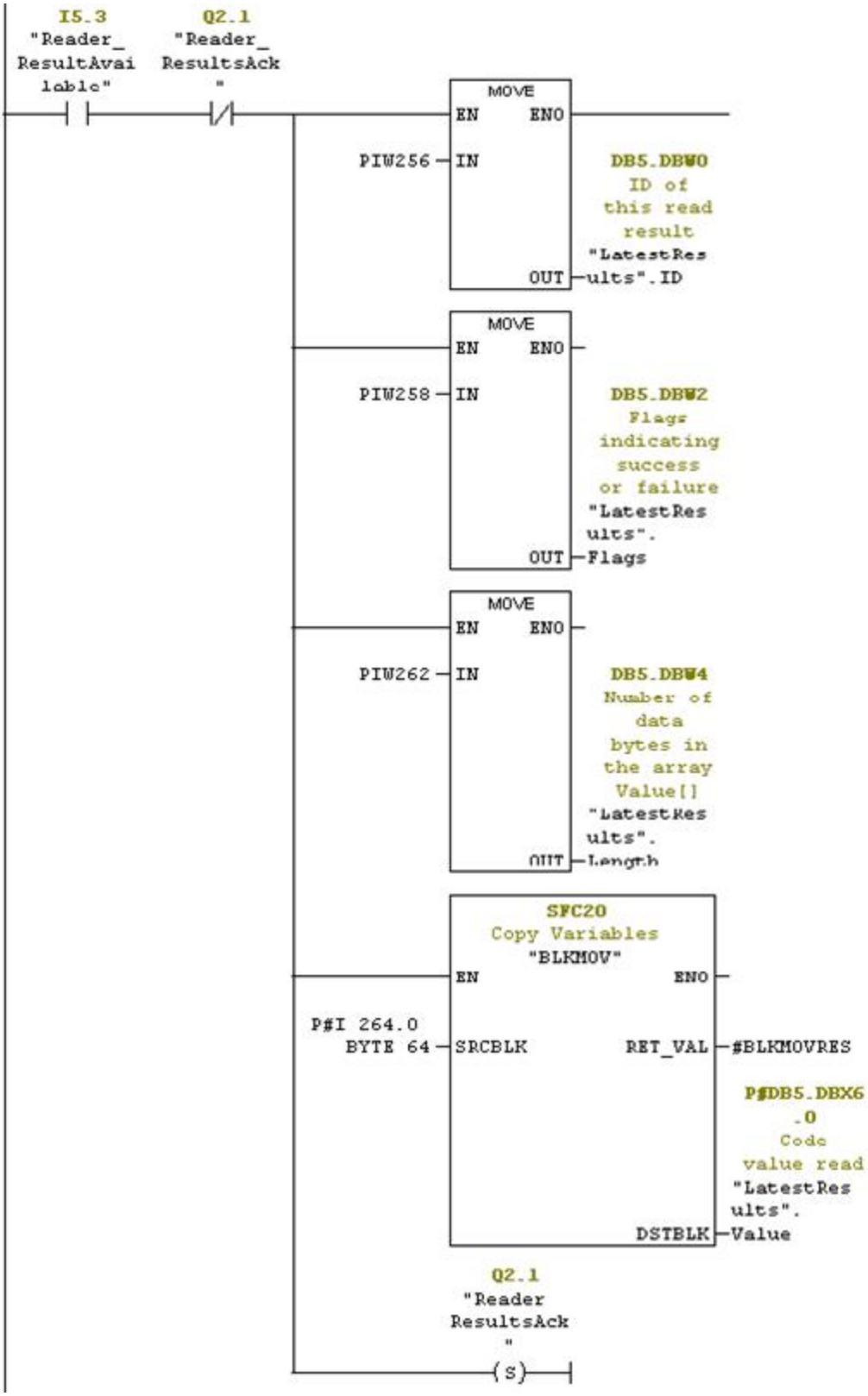
3. Set the trigger signal and set coil to indicate a read is pending.



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

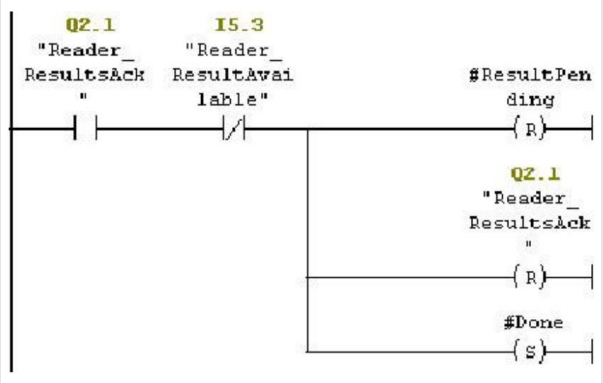


5. As soon as the results are available, save a copy of the result data and set the results acknowledge signal.



6. When the reader sees the result acknowledge signal, clear result acknowledge, clear the read pending coil, and signal that the read process is complete.

Note: The reader clears “Results Available” as soon as it sees the PLC’s “Results Ack” signal.



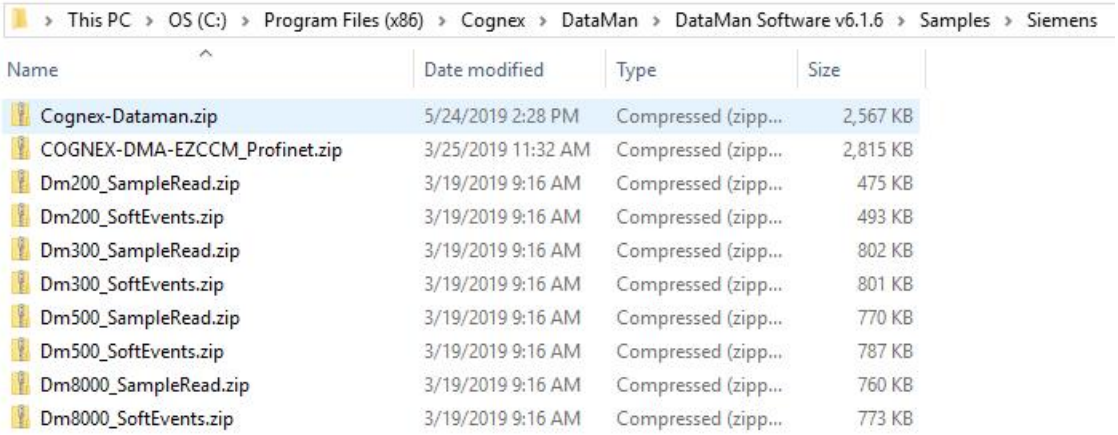
Using SoftEvents

Run the sample program “DM200_SoftEvents” for the complete example program.

Note: This sample can be used with any PROFINET enabled DataMan reader.

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.



5. Select Dm200_SoftEvents.zip from the Siemens folder.
6. Select a destination folder to save the project on your PC.

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

SoftEvents are a means of invoking an activity by manipulating a single control bit. The activity for each bit is predefined (for more details, see section SoftEvents). With the exception of “Execute DMCC” and “Set Match String” all SoftEvents may be invoked in the same way. “Execute DMCC” and “Set Match String” require the added step of loading the User Data module with application data before invoking the event.

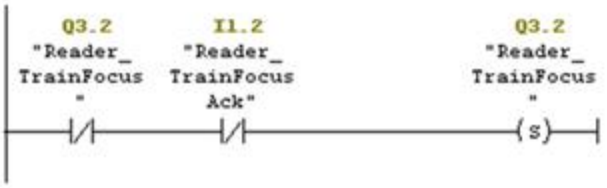
Reduced to the basics, the process of invoking a SoftEvent consists of the following:

FC3 : Train Focus

Initiate the "Train Focus" operation and monitor it to completion.

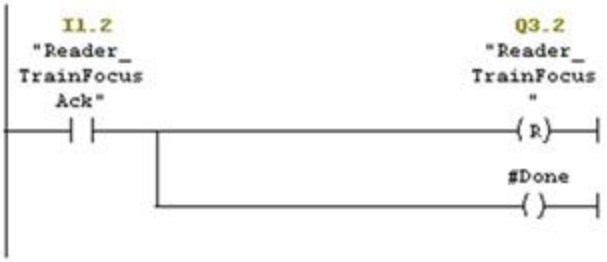
Network 1 : Title:

Issue "Train Focus" signal.



Network 2 : Title:

Release "Train Focus" signal as soon as it is acknowledged by the reader.



Network 3 : Title:

Set the return FC state.

Note, this will only return TRUE after the acknowledge has been received from the reader. Otherwise it will return FALSE.

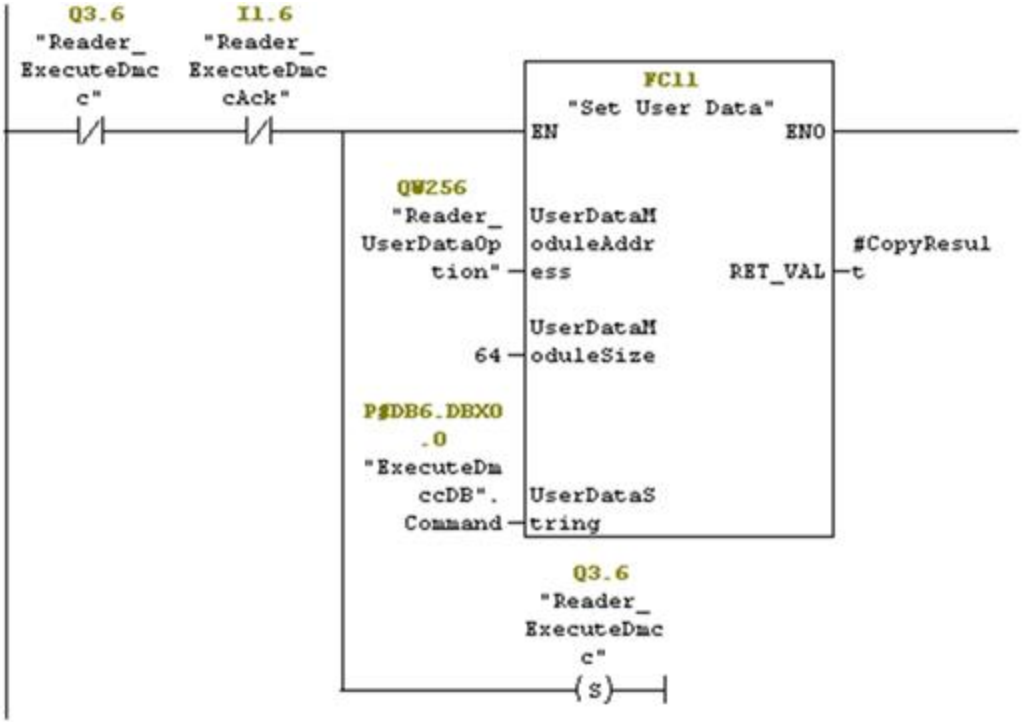
Executing DMCC Commands

Refer to sample program “DM200_SoftEvents” for the complete example program. For information on how to install it, see section Using SoftEvents.

Note: This sample can be used with any PROFINET enabled DataMan reader.

“Execute DMCC” is a SoftEvent which requires the added step of loading the User Data module with the desired DMCC command string before invoking the event. By default, the SoftEvent mechanism does not provide a means of returning DMCC response data other than a failure indication. To collect DMCC response data, you need to enable the DMCC Result Feedback feature, which is not shown in the following example.

The process of executing a DMCC command is the same for all other SoftEvents as in the example above except the step of invoking the SoftEvent also includes copying the command string to the User Data Module. In this example the command string is in a Data Block. This example can be expanded to utilize a Data Block with an array of command strings that the copy function can reference by an index value. It allows the user to pre-define all DMCC commands that are required by the application and invoke them by index.



The function “Set User Data” (FC11) copies the provided string to the User Data module. Refer to the example program for the actual STL code.

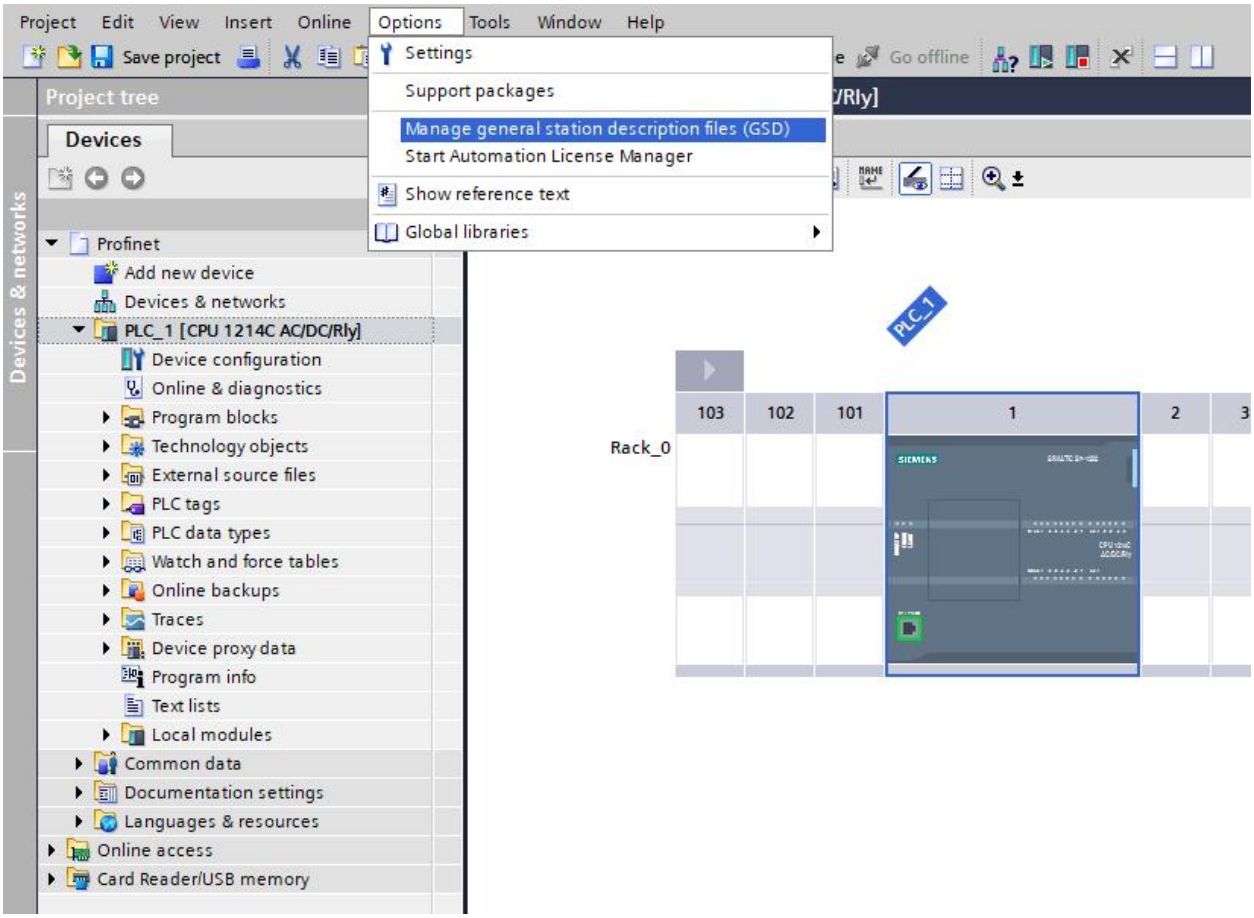
TIA Portal Examples

This section gives some examples of using a DataMan reader with a Siemens S7-1200 PLC, assuming that you are familiar with the S7-1200 and TIA Portal.

1. Launch TIA Portal, and either create a new project, or open an existing one.

> **Note**: If you have existing projects with older GSD files and would like to start a project that references only latest GSDs, refer to *Removing Outdated GSD Files from TIA Portal* on page 75. This ensures that only a single GSD option is available for each product.

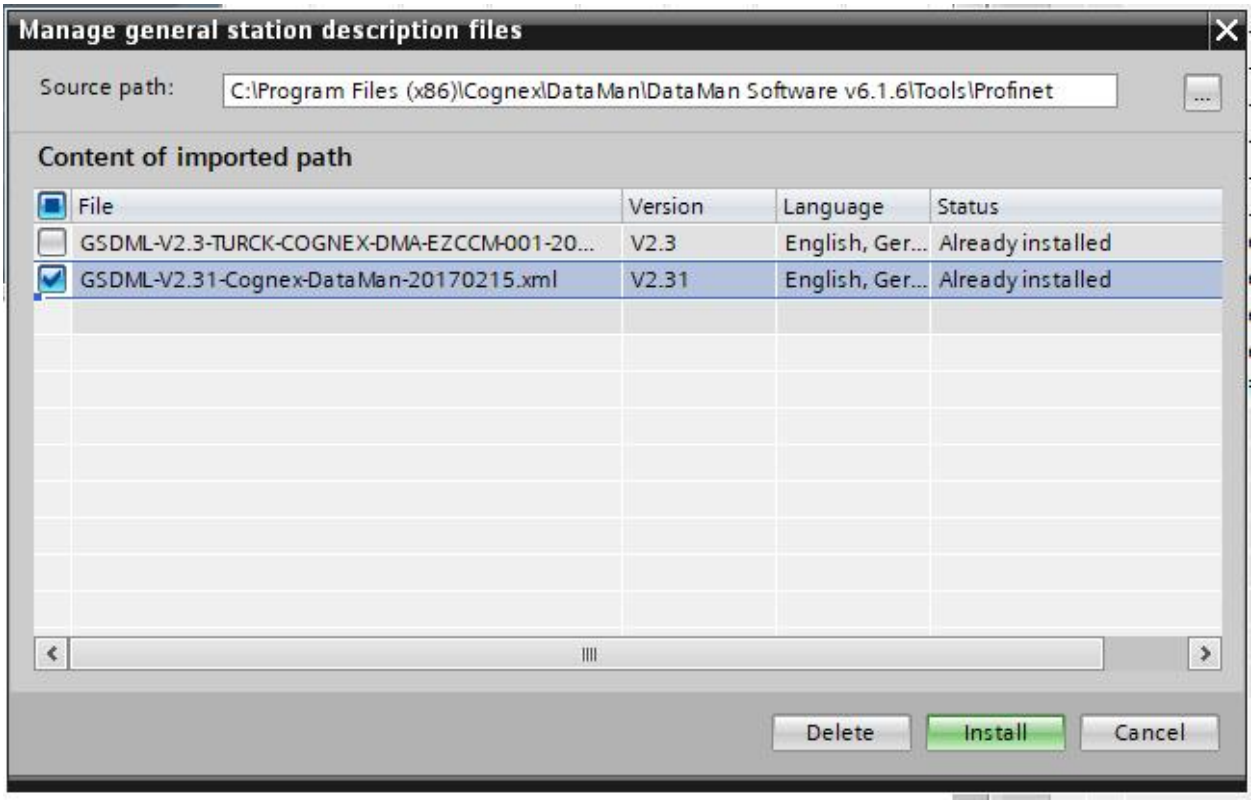
2. In **Project View**, click **Options** and click **Manage general station description files (GSD).**



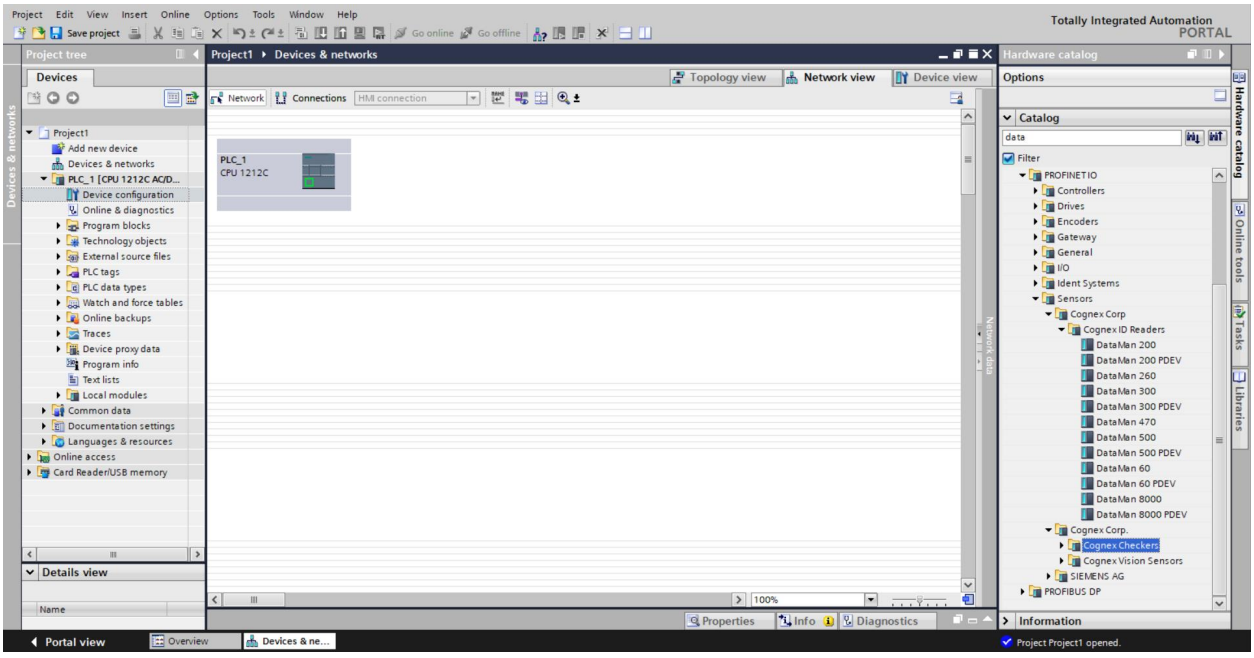
3. Browse to the installation folder of the GSD file (or the location where you saved the GSD file if you downloaded it from the web).

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

> **Note**: If there is more than one GSD file in the list and you are unsure which to install, choose the one with the most recent date.



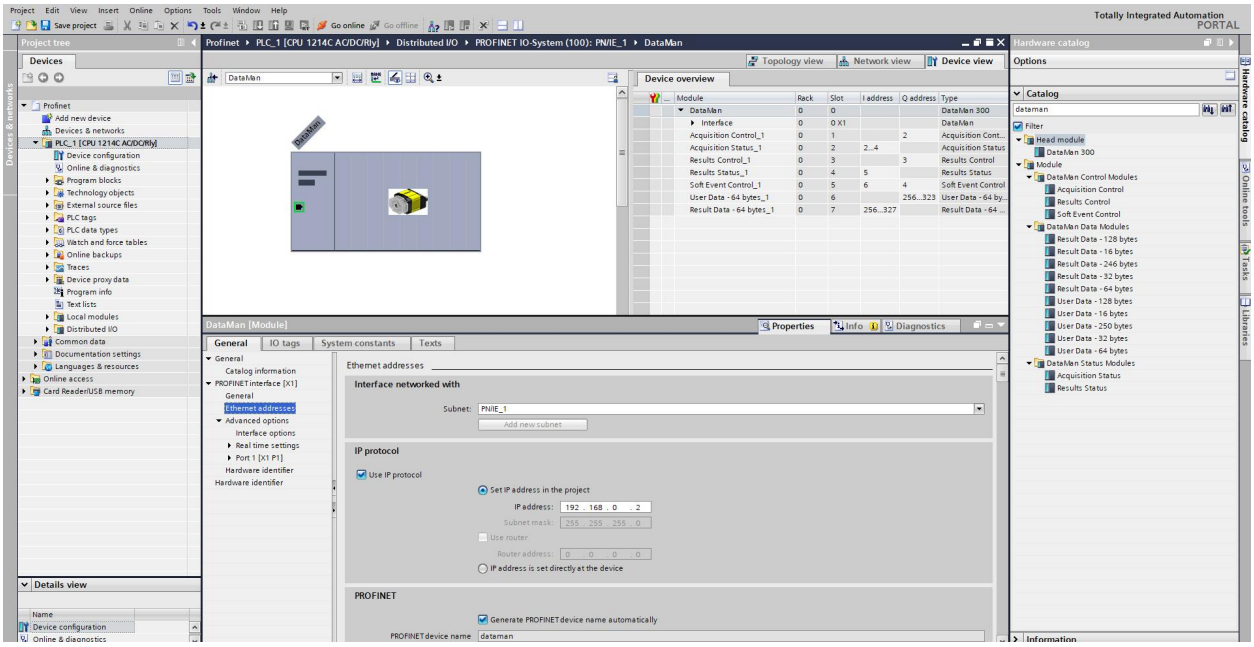
5. Go to **Device Configuration** and add your device in the **Network View** tab.
6. The **Hardware catalog** is displayed on the right. Expand the “PROFINET IO” tree to the “Cognex ID Readers” node.
7. Press and hold the left mouse button to drag the DataMan reader and drop it on the PROFINET IO network symbol in the left pane.



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

Note: By default, the 64 byte User Data and 64 byte Result Data Modules are inserted. There are multiple sizes available for both of these modules. To optimize performance, use the module size that most closely matches the actual data requirements of your application. You can change the module by deleting the one in the table and inserting the appropriate sized module from the catalog.

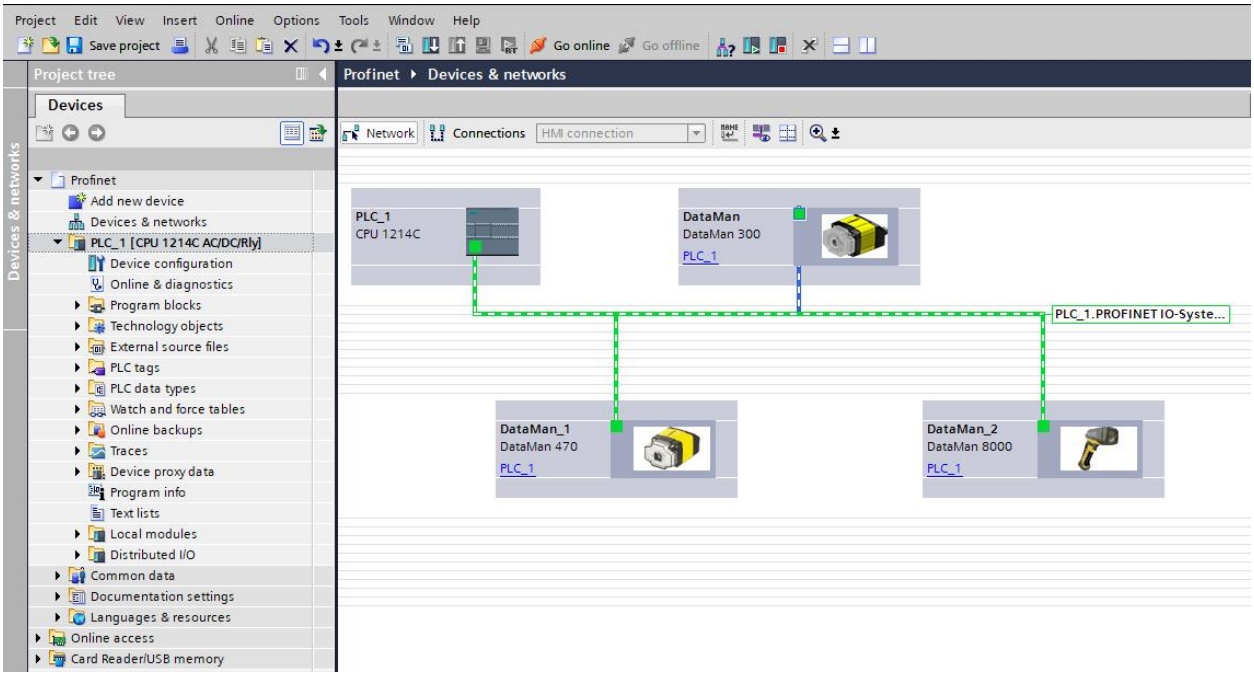
8. Right-click the DataMan icon and select **Properties...**.



9. Give the reader a name. This must match the name of your actual DataMan reader. The name must be unique and follow DNS naming conventions. For details, see the TIA Portal help documentation.

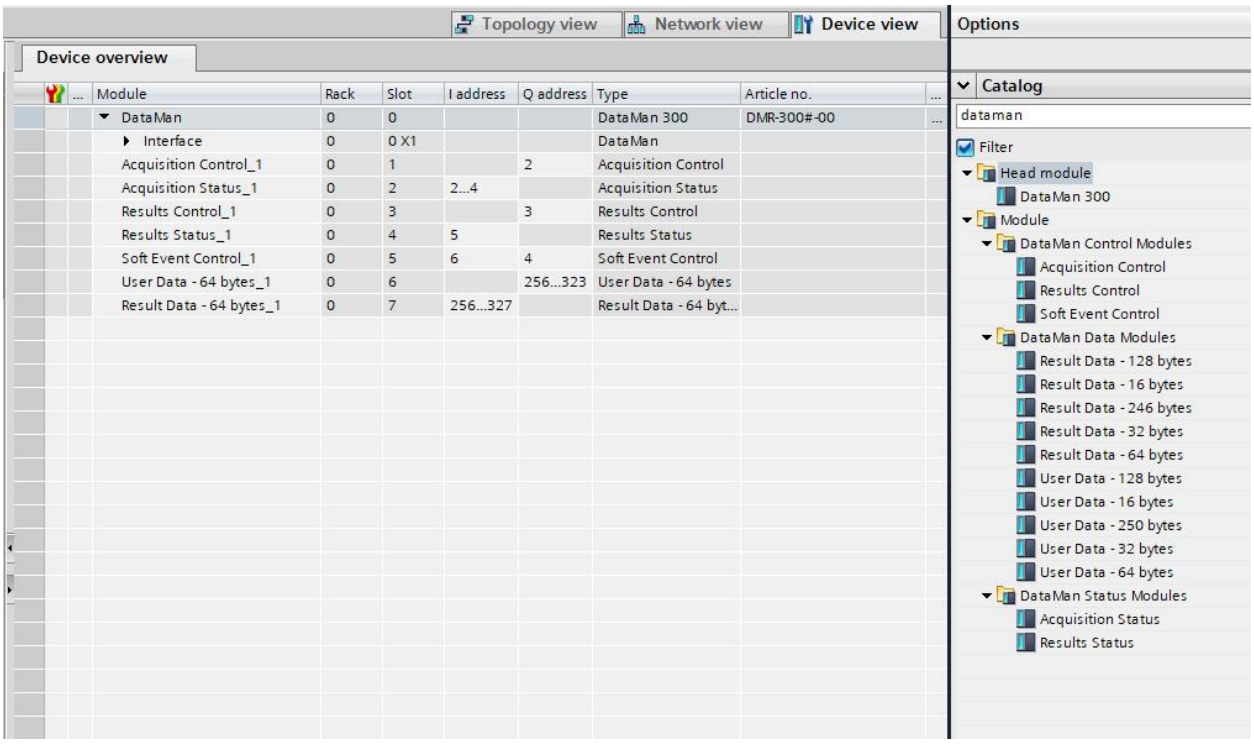
10. If your DataMan reader is configured to use its own static IP, select the **IP address is set directly at the device** radio button. As an alternative, if you wish the PLC to assign an IP address, select the Ethernet button and configure the appropriate address.

11. Check the checkbox next to the device image and connect the selected device with the PLC by drawing a line with your mouse.



12. By default, the SIMATIC software maps the User Data and Result Data Modules to offset 256. This is outside of the default process image area size of 128. That is, 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.



Module	Rack	Slot	I address	Q address	Type	Article no.
DataMan	0	0			DataMan 300	DMR-300#-00
Interface	0	0 X1			DataMan	
Acquisition Control_1	0	1		2	Acquisition Control	
Acquisition Status_1	0	2	2...4		Acquisition Status	
Results Control_1	0	3		3	Results Control	
Results Status_1	0	4	5		Results Status	
Soft Event Control_1	0	5	6	4	Soft Event Control	
User Data - 64 bytes_1	0	6		256...323	User Data - 64 bytes	
Result Data - 64 bytes_1	0	7	256...327		Result Data - 64 byt...	

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.

Symbol Table

It is recommended that you define symbols for the DataMan I/O module elements to make the code much easier to read and reduce mistakes. This sample table shows symbols defined for a typical instance of a DataMan reader. It is possible that DataMan I/O modules are at different addresses in your project. Make sure to adjust your symbol definitions based on the specific offsets of the I/O modules. For more information, see the *TIA Portal Integration Guide for In-Sight and DataMan* at https://support.cognex.com/en/downloads/detail/in-sight/3687/1033.

Start ▸ PLC_1 [CPU 1212C AC/DC/Rly] ▸ PLC tags ▸ Tag table_1 [30]

Tag table_1

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	Dataman-TriggerEnable	Bool	%Q2.0		☑	☑	☑	
2	Dataman-Trigger	Bool	%Q2.1		☑	☑	☑	
3	Dataman-TriggerReady	Bool	%I2.0		☑	☑	☑	
4	Dataman-TriggerAck	Bool	%I2.1		☑	☑	☑	
5	Dataman-Acquiring	Bool	%I2.2		☑	☑	☑	
6	Dataman-MissedAcq	Bool	%I2.3		☑	☑	☑	
7	Dataman-TriggerID	Word	%IW3		☑	☑	☑	
8	Dataman-BufferResultsEnable	Bool	%Q3.0		☑	☑	☑	
9	Dataman-ResultsAck	Bool	%Q3.1		☑	☑	☑	
10	Dataman-Decoding	Bool	%I5.0		☑	☑	☑	
11	Dataman-Decode Complete	Bool	%I5.1		☑	☑	☑	
12	Dataman-Results Buffer Overrun	Bool	%I5.2		☑	☑	☑	
13	Dataman-Results Available	Bool	%I5.3		☑	☑	☑	
14	Dataman-General Fault	Bool	%I5.7		☑	☑	☑	
15	Dataman-Train Code	Bool	%Q6.0		☑	☑	☑	
16	Dataman-Train Match String	Bool	%Q6.1		☑	☑	☑	
17	Dataman-Train Focus	Bool	%Q6.2		☑	☑	☑	
18	Dataman-Train Brightness	Bool	%Q6.3		☑	☑	☑	
19	Dataman-Untrain	Bool	%Q6.4		☑	☑	☑	
20	Dataman-Execute DMCC	Bool	%Q6.6		☑	☑	☑	
21	Dataman-Set Match String	Bool	%Q6.7		☑	☑	☑	
22	Dataman-Train Code Ack	Bool	%I4.0		☑	☑	☑	
23	Dataman-Train Match String Ack	Bool	%I4.1		☑	☑	☑	
24	Dataman-Train Focus Ack	Bool	%I4.2		☑	☑	☑	
25	Dataman-Train Brightness Ack	Bool	%I4.3		☑	☑	☑	
26	Dataman-Untrain Ack	Bool	%I4.4		☑	☑	☑	
27	Dataman-Execute DMCC Ack	Bool	%I4.6		☑	☑	☑	
28	Dataman-Set Match String Ack	Bool	%I4.7		☑	☑	☑	
29	Dataman-UserData Option	Word	%QW256		☑	☑	☑	
30	Dataman-UserData Length	Word	%QW258		☑	☑	☑	

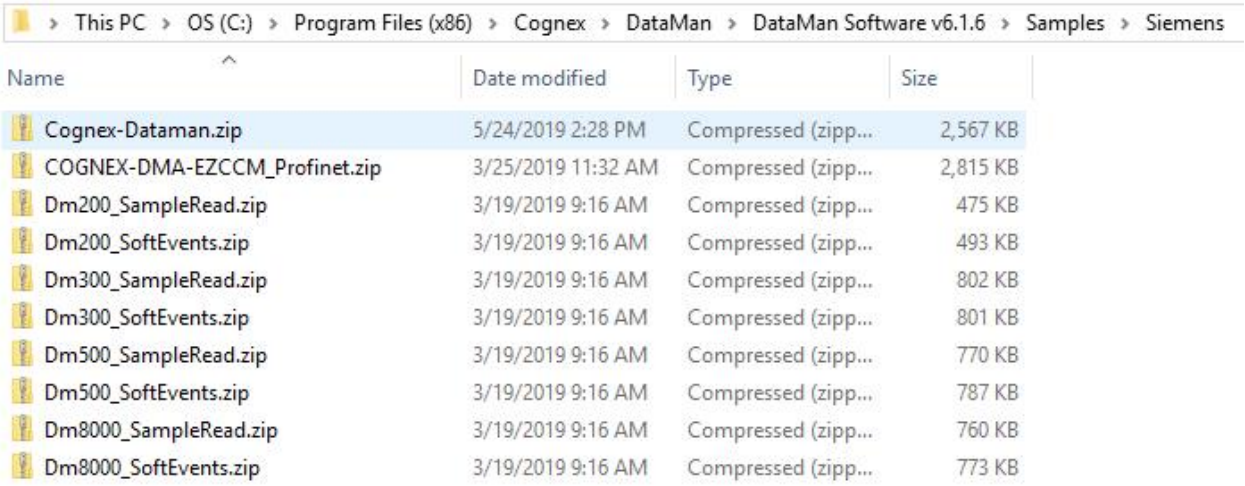
Trigger and Get Results

Open the sample program “Cognex-Dataman” for the complete example program.

Note: This sample can be used with any PROFINET enabled DataMan reader.

Perform the following steps to install the program:

1. Start TIA Portal.
2. Browse to find the sample file on your PC.



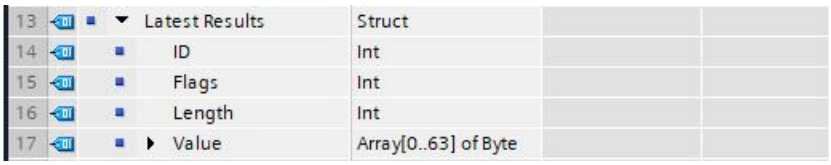
This PC › OS (C:) › Program Files (x86) › Cognex › DataMan › DataMan Software v6.1.6 › Samples › Siemens

Name	Date modified	Type	Size
Cognex-Dataman.zip	5/24/2019 2:28 PM	Compressed (zipp...	2,567 KB
COGNEX-DMA-EZCCM_Profinet.zip	3/25/2019 11:32 AM	Compressed (zipp...	2,815 KB
Dm200_SampleRead.zip	3/19/2019 9:16 AM	Compressed (zipp...	475 KB
Dm200_SoftEvents.zip	3/19/2019 9:16 AM	Compressed (zipp...	493 KB
Dm300_SampleRead.zip	3/19/2019 9:16 AM	Compressed (zipp...	802 KB
Dm300_SoftEvents.zip	3/19/2019 9:16 AM	Compressed (zipp...	801 KB
Dm500_SampleRead.zip	3/19/2019 9:16 AM	Compressed (zipp...	770 KB
Dm500_SoftEvents.zip	3/19/2019 9:16 AM	Compressed (zipp...	787 KB
Dm8000_SampleRead.zip	3/19/2019 9:16 AM	Compressed (zipp...	760 KB
Dm8000_SoftEvents.zip	3/19/2019 9:16 AM	Compressed (zipp...	773 KB

3. Save the project on your PC. TIA Portal extracts the sample archive and makes it available.

Reduced to the basics, the process of reading and retrieving results consists of the following:

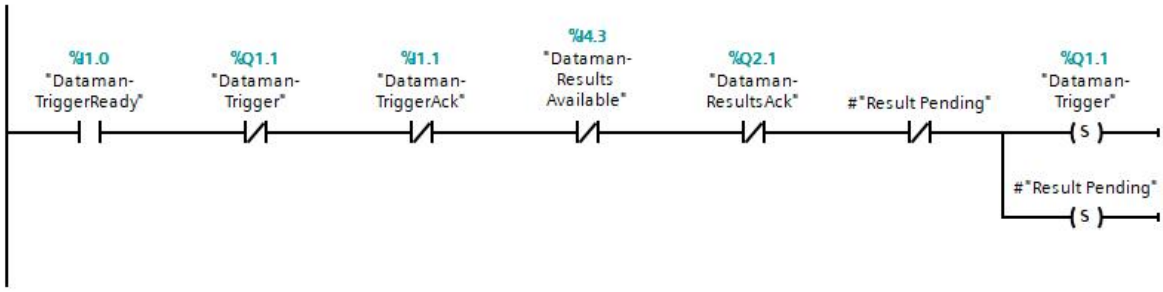
1. Define an area in your application to save read results. There are many options regarding how and where result data can be stored. In this example, a temporary tag is defined containing the fields of the Result Data module that are relevant for our application.



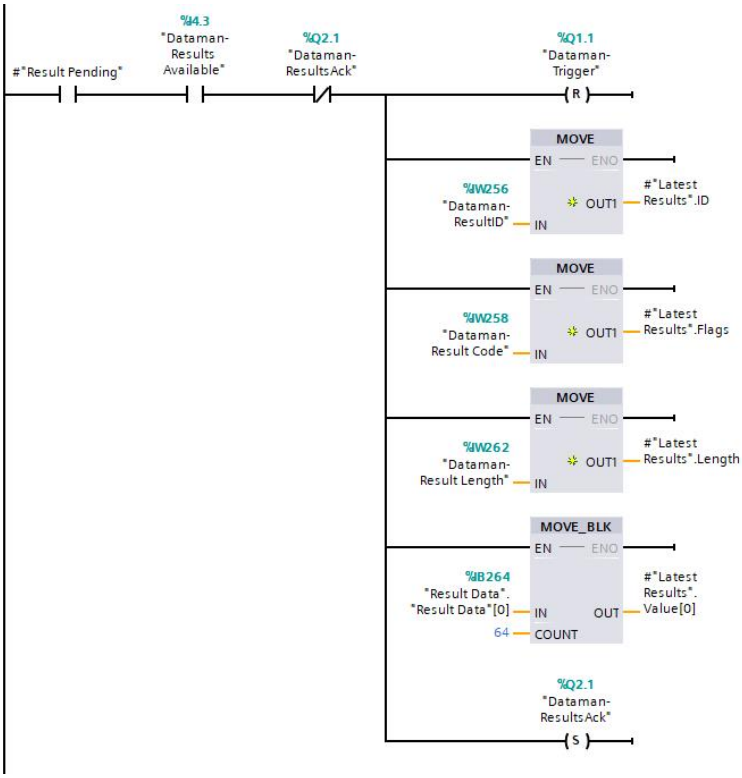
13	Latest Results	Struct	
14	ID	Int	
15	Flags	Int	
16	Length	Int	
17	Value	Array[0..63] of Byte	

2. Enable the reader.

3. Set the trigger signal and set coil to indicate a read is pending.

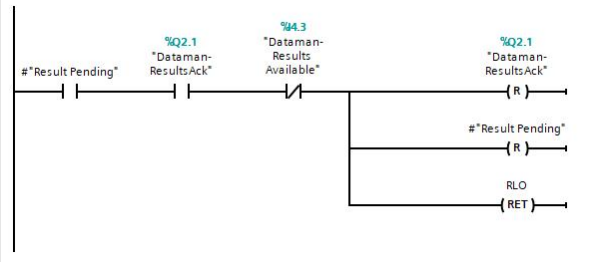


4. As soon as the results are available, clear the trigger signal and save a copy of the result data and set the results acknowledge signal.



5. When the reader sees the result acknowledge signal, clear result acknowledge, clear the read pending coil, and signal that the read process is complete.

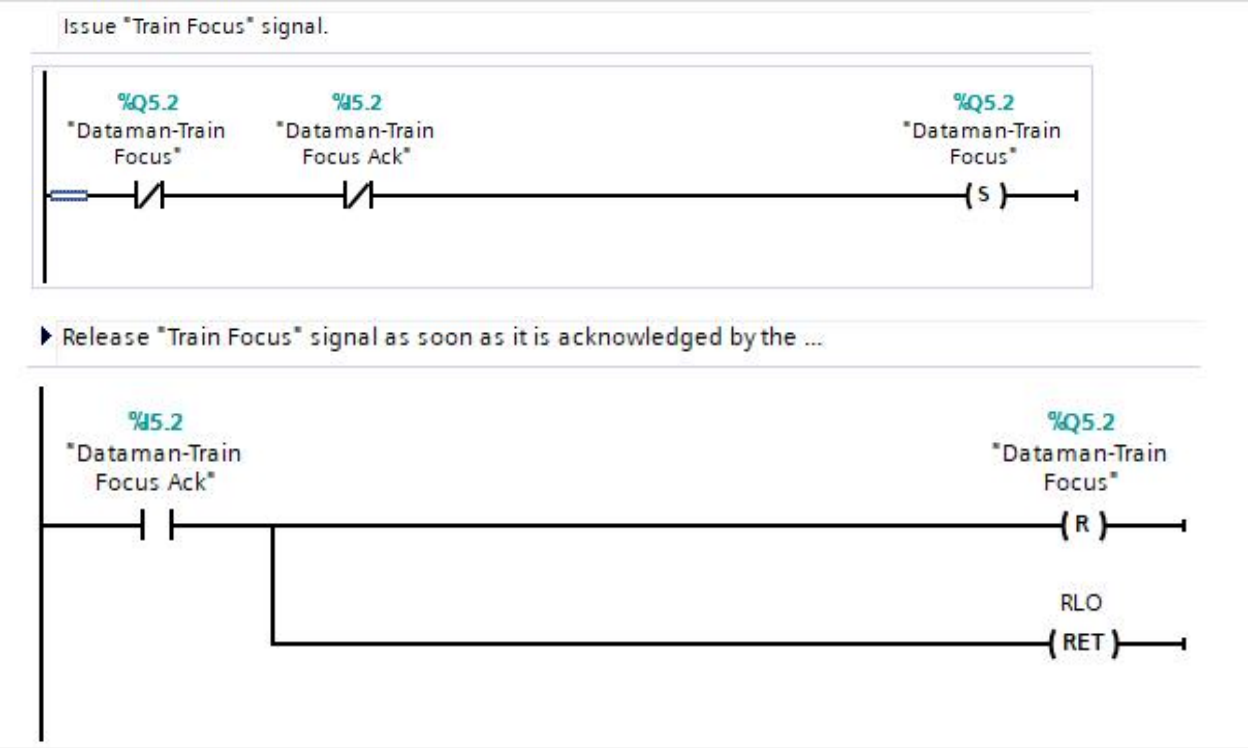
Note: The reader clears “Results Available” as soon as it sees the PLC’s “Results Ack” signal.



Using SoftEvents

SoftEvents are a means of invoking an activity by manipulating a single control bit. The activity for each bit is predefined. For more details, see section SoftEvents. With the exception of “Execute DMCC” and “Set Match String” all SoftEvents may be invoked in the same way. “Execute DMCC” and “Set Match String” require the added step of loading the User Data module with application data before invoking the event.

Reduced to the basics, the process of invoking a SoftEvent consists of the following:



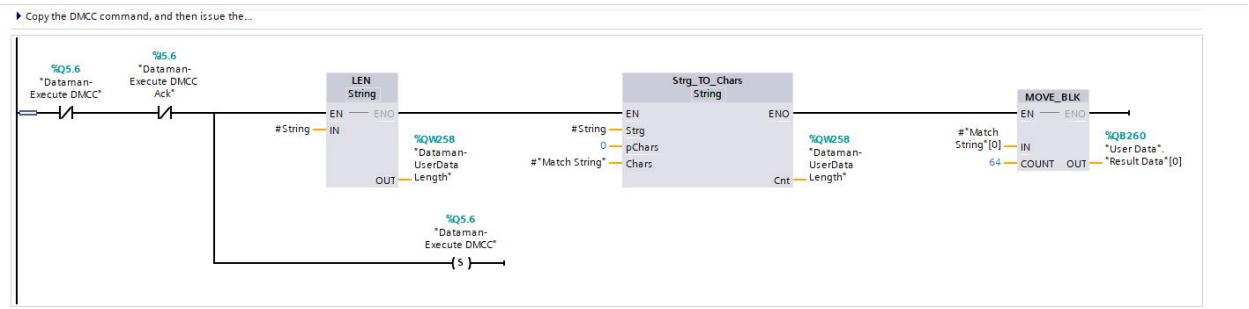
Executing DMCC Commands

Refer to sample program “Cognex-Dataman” for the complete example program. For information on how to install it, see section Using SoftEvents.

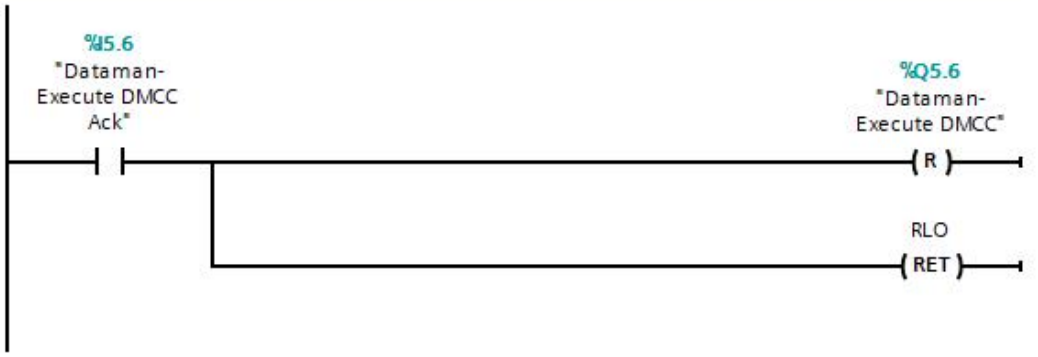
> **Note**: This sample can be used with any PROFINET enabled DataMan reader.

“Execute DMCC” is a SoftEvent which requires the added step of loading the User Data module with the desired DMCC command string before invoking the event. By default, the SoftEvent mechanism does not provide a means of returning DMCC response data other than a failure indication. To collect DMCC response data, you need to enable the DMCC Result Feedback feature, which is not shown in the following example.

The process of executing a DMCC command is the same for all other SoftEvents (see example above) except the step of invoking the SoftEvent also includes copying the command string to the User Data Module. In this example the command string is exists in a Data Block. This example can be expanded to utilize a Data Block with an array of command strings that the copy function can reference by an index value. This allows the user to pre-define all DMCC commands that are required by the application and invoke them by index.



Release "Execute DMCC" signal as soon as it is acknowledged by the reader.



Removing Outdated GSD Files from TIA Portal

Note: Only execute this procedure if you have existing projects with older GSD files and would like to start a project that references only latest GSDs. Do not uninstall GSD files currently used in the project, otherwise your PLC cannot recognize the reader. Such readers are shown as `GSD device_<number>`.

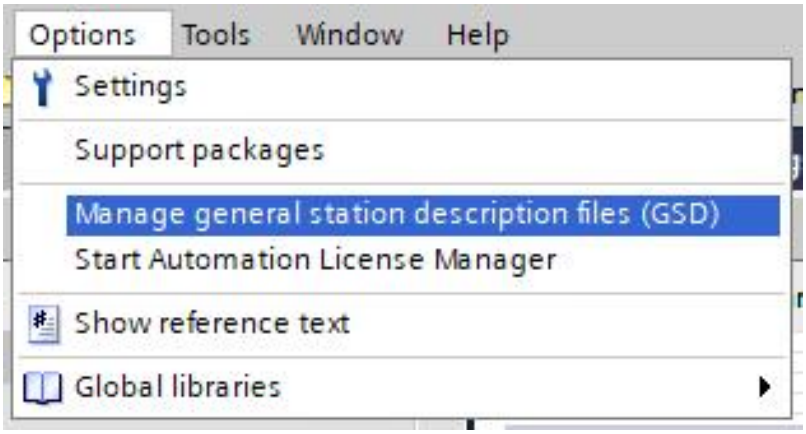
Legacy and current DataMan readers use a different installation location for GSD files in the Hardware catalog of TIA Portal. The old `Cognex Corp` location is superseded by the newer `Cognex Corp.` location. This section describes the procedure of removing the outdated location to ensure that only the latest GSD for each DataMan reader is installed.

Uninstall Outdated GSD Files

You need to uninstall GSD files that reference the legacy `Cognex Corp` installation location.

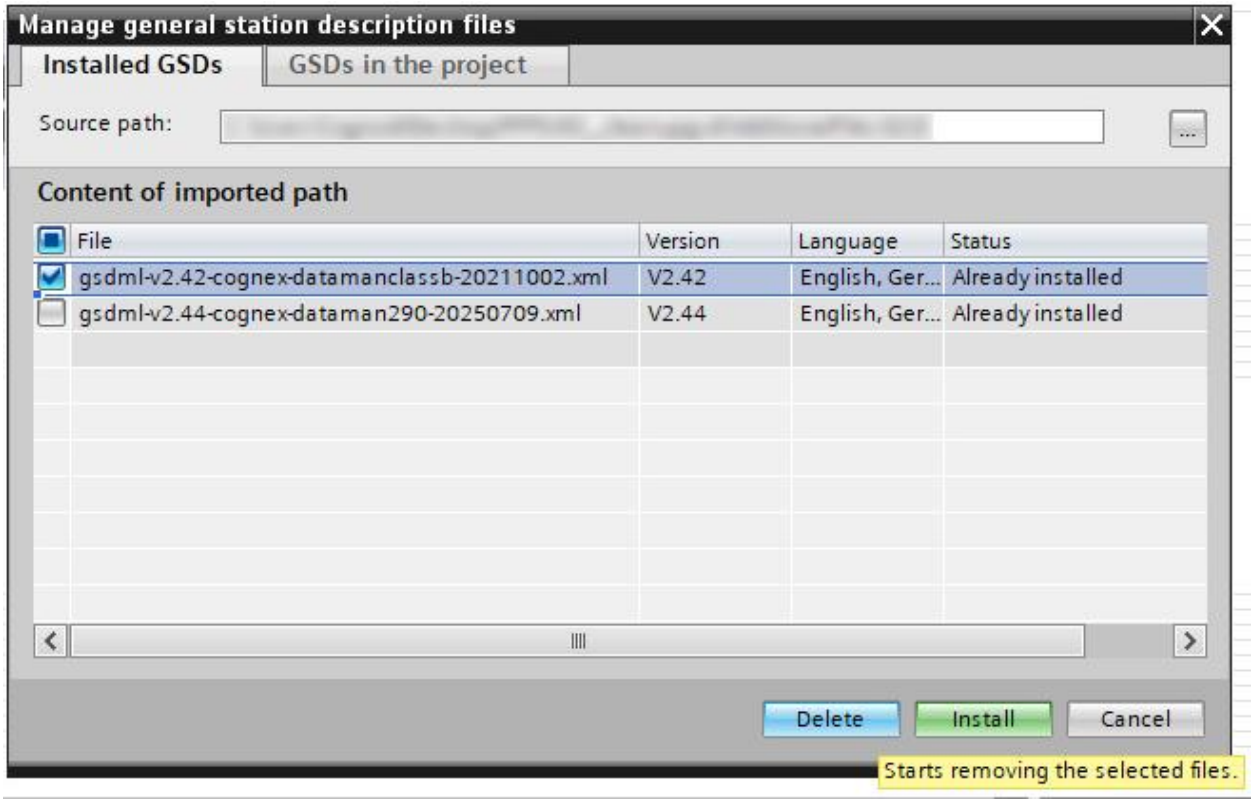
To uninstall outdated GSD files:

1. Open TIA Portal and go to **Options > Manage general station description files (GSD)**.

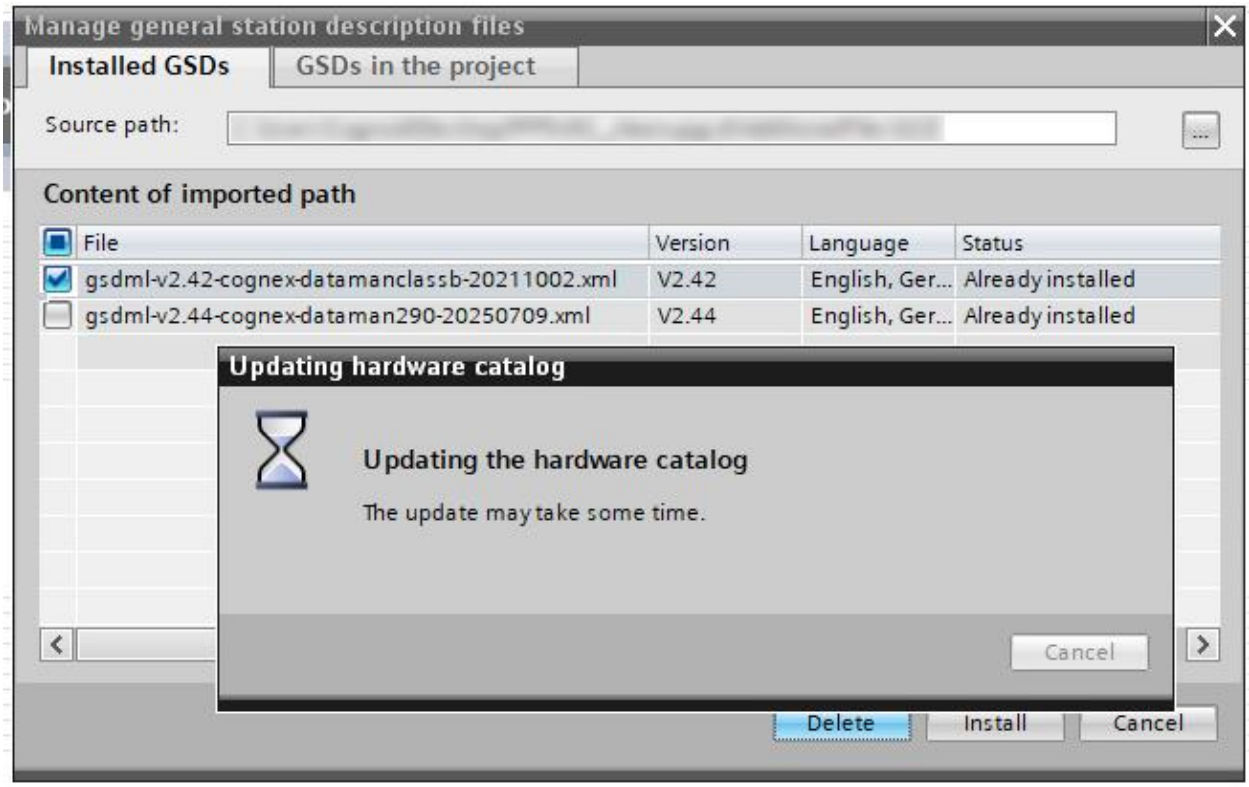


2. Remove files that reference `Cognex Corp` without the period. Select the checkbox next to the files and click **Delete**.

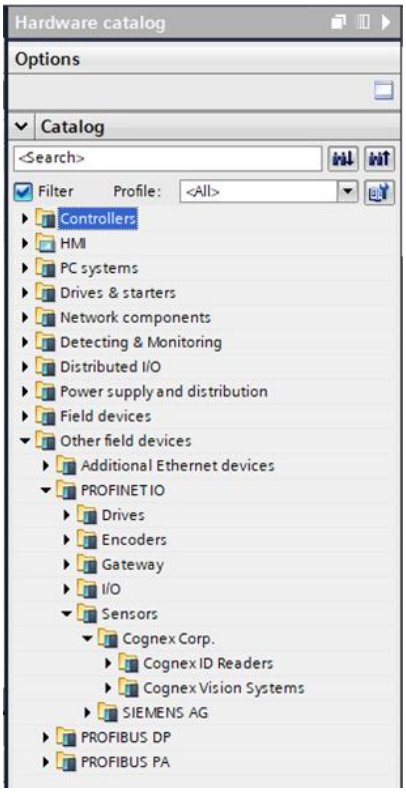
   The following example removes the combined GSD file with the date of 20211002.



3. Wait for the **Hardware catalog** to update.



4. Ensure that the old `Cognex Corp` location was removed.

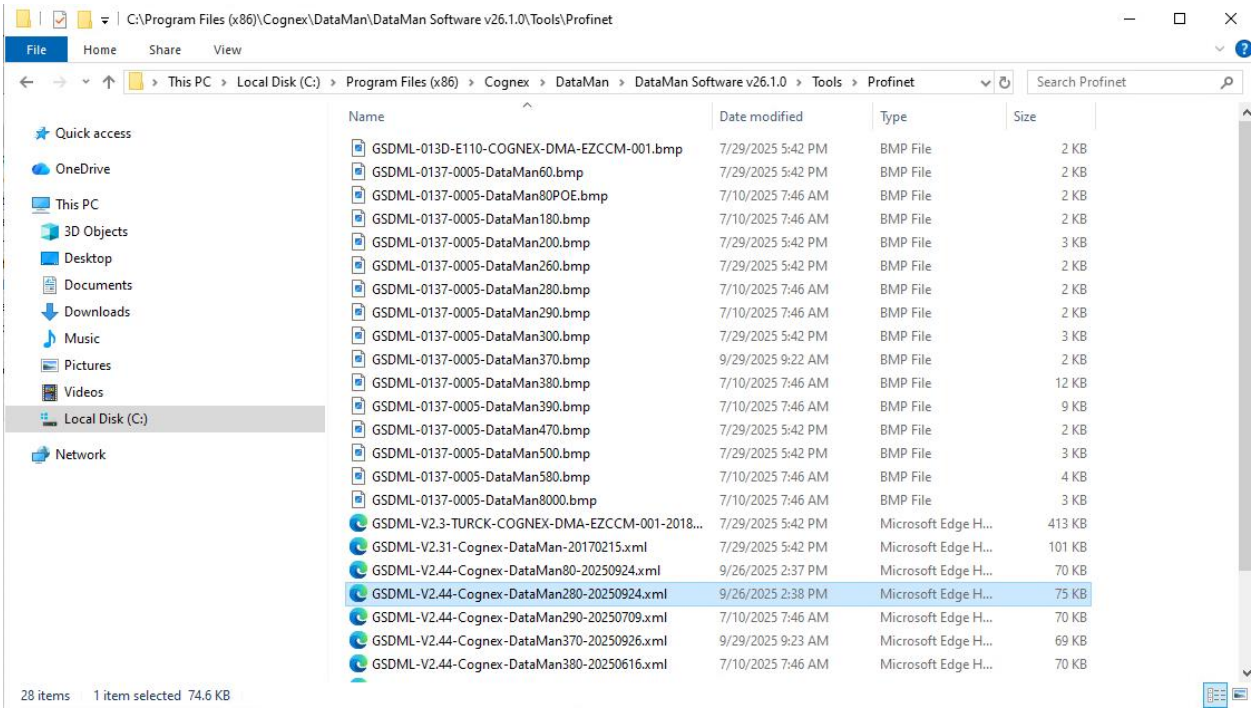


Install GSD Files with the New Location

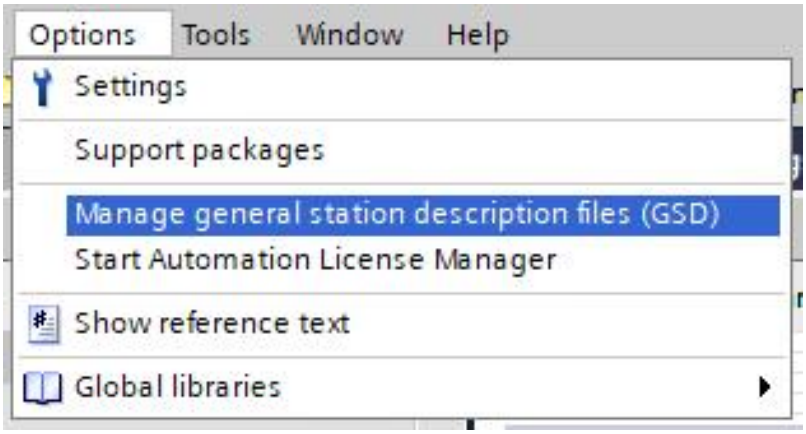
Use the individual GSD file for the reader you want to install. The following example shows GSD installation for a DataMan 280.

To install the GSD file for a DataMan 280:

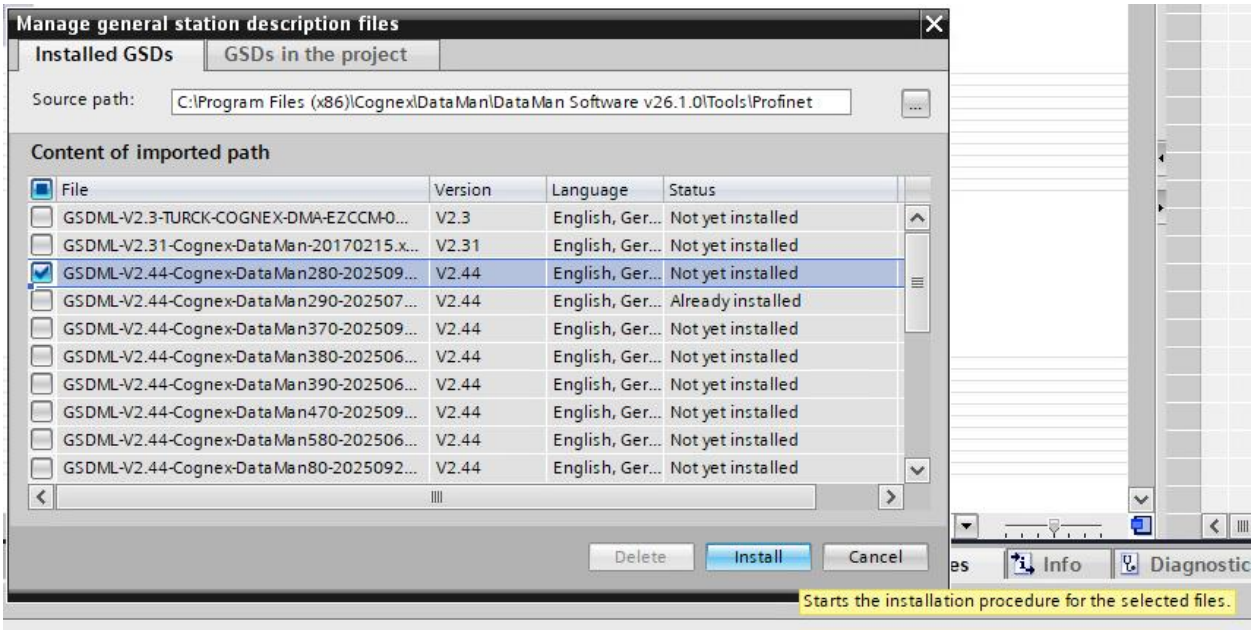
1. Find the GSD file where DataMan Setup Tool is installed. The default folder is `C:\Program Files (x86)\Cognex\DataMan\DataMan Software v<version>\Tools\Profinet`.



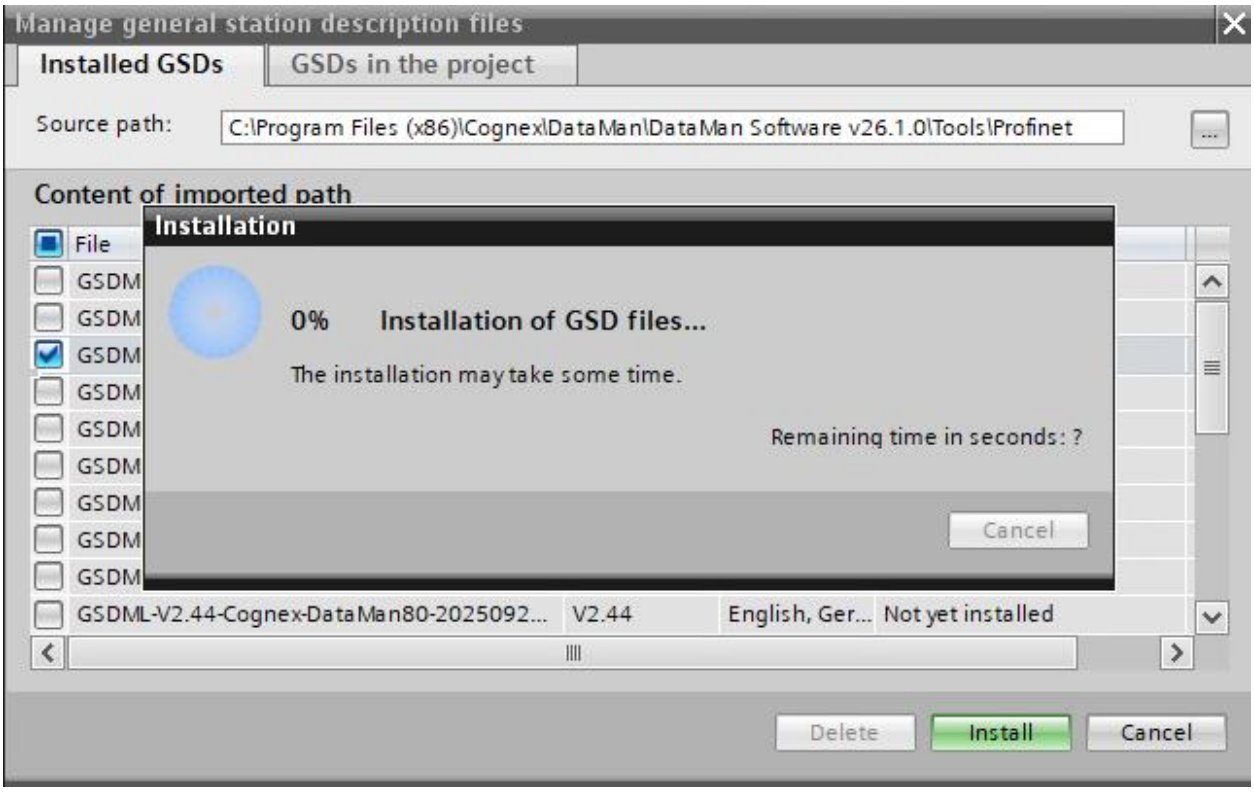
2. Open TIA Portal and go to **Options > Manage general station description files (GSD)**.



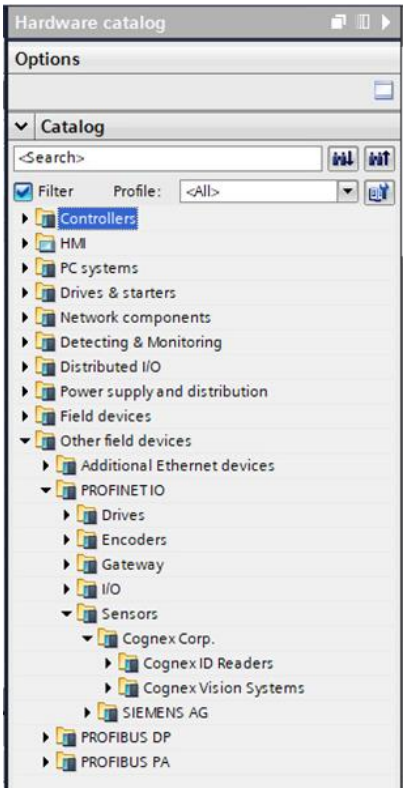
3. Change the **Source path** to the folder located in the first step. Select your file, and click **Install**.



4. Wait until the installation is complete and the **Hardware catalog** is updated.



5. Verify that the new reader is installed at the `Cognex Corp.` location.



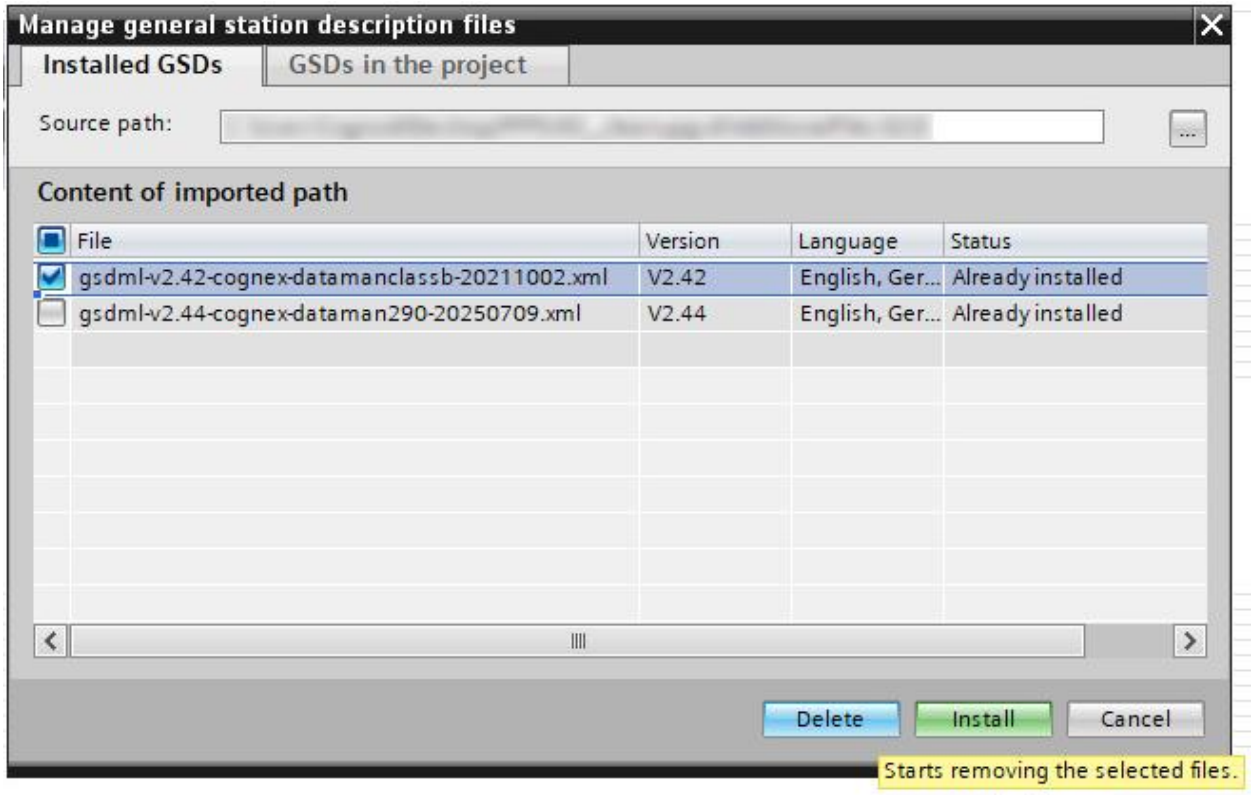
Optional: Remove GSD Files Remaining in the Hardware Catalog

Note: Only execute this procedure if you do not want to open your projects that use legacy GSD files. If you remove them directly from your `ProgramData` folder, TIA Portal might not be able to open projects that reference these files.

Sometimes GSD files can still show up in the **Hardware Catalog** because TIA Portal stored them in an additional place on the file system. The following example shows how to remove the GSD file for a DataMan 580. The example uses TIA V17, but it is valid for other TIA versions.

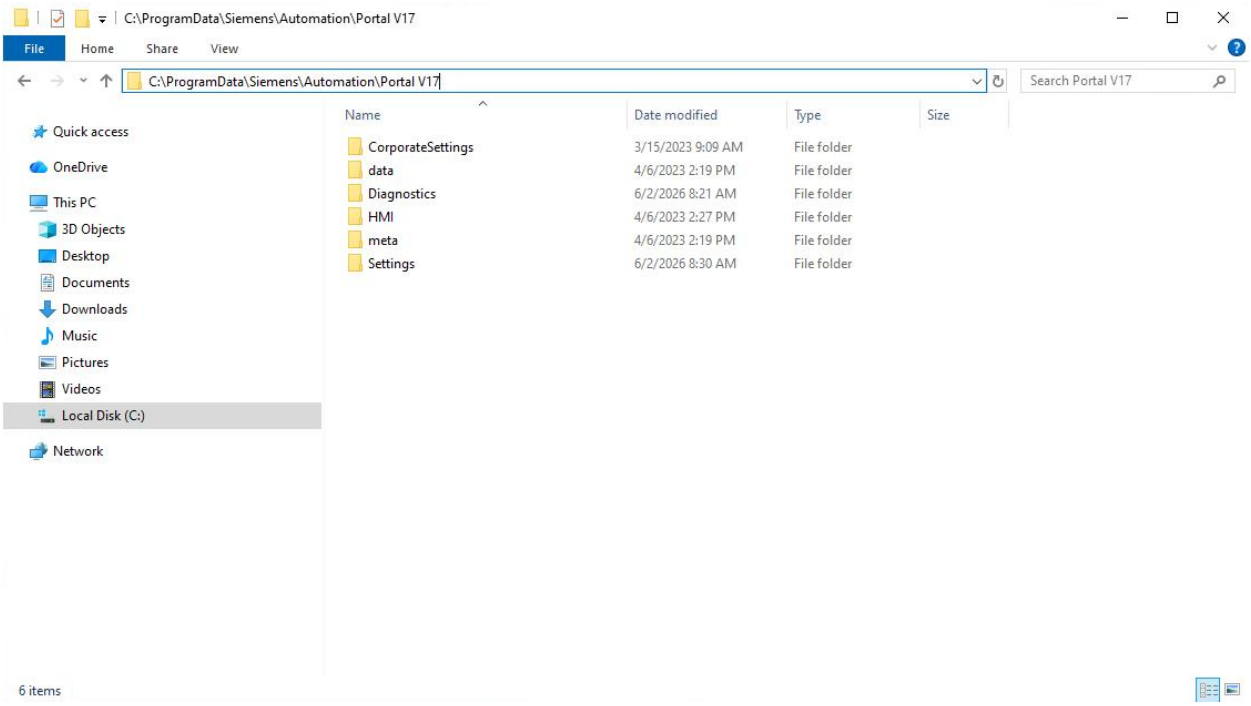
To remove the GSD file for a DataMan 580 from your file system:

1. Verify that the GSD file you want to remove is not installed in your project.
   a. Check that the file does not appear in the **AdditionalFiles\GSD location** of the project.
   b. If the file is installed, uninstall it:



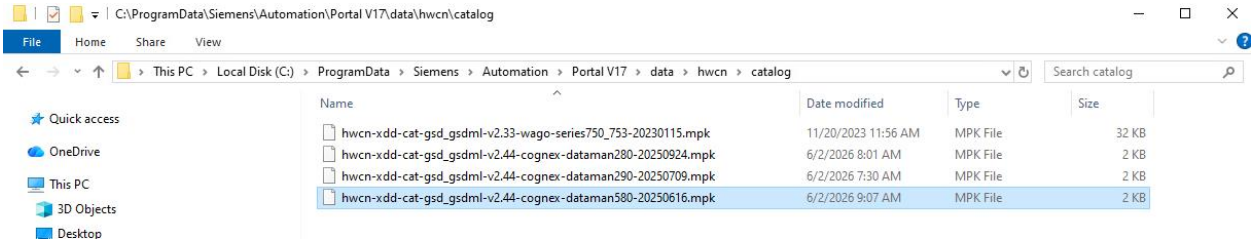
2. Close all active instances of TIA Portal.

3. Navigate to the `ProgramData` folder for your installed TIA instance. In the following example, the folder is `C:\ProgramData\Siemens\Automation\Portal V17`.

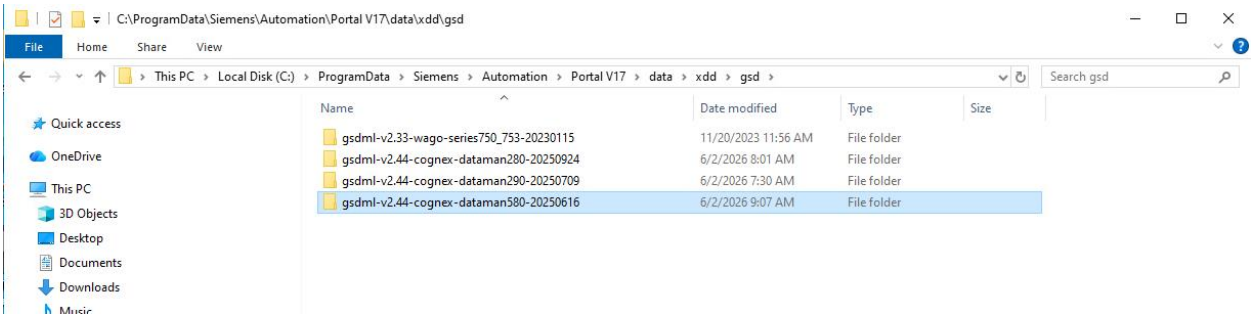


4. Remove the related files from the following locations:

   - `data/hwcn/catalog`



   - `data/xdd/gsd`



5. Launch TIA Portal and verify that the persistent GSD file is removed.

PROFINET Class B

Upgrading a DataMan reader to firmware version 6.1.7 or higher installs a new PROFINET stack, adding support for PROFINET Class B. While this adds new functions and capabilities, it is not a drop-in replacement for a DataMan reader running the Class A stack, which all previous versions of DataMan firmware supported. You need to update the hardware configuration in the TIA software.

If you add new DataMan readers to an existing system that has other DataMan products, both GSD files need to be installed. Refer to the table for which GSD file to use with a given reader.

Firmware Version	GSD File	Description
6.1.7 or later	V2.34 or later	Supports Class B, uses the Port stack. The Class-B configurations are suffixed with CC-B, indicating that the configuration is Class B. Example: DataMan 470 CC-B.
Earlier than 6.1.7	V2.31 or earlier	Uses the Siemens PROFINET stack.

SNMP (Simple Network Management Protocol) v2 is supported only when the V2.34 GSD file is used. PROFINET uses the topology technology to detect and identify devices on SNMP.

Supported MIBs:

- LLDP-MID
- LLDP-EXT-DOT3-MIB
- MIB-II

Writable OIDs:

- 1.3.6.1.2.1.1.4.0 (sysContact, defaults to “Someone <someone@somewhere.net>”)
- 1.3.6.1.2.1.1.5.0 (sysLocation, defaults to “somewhere”)
- 1.3.6.1.2.1.1.6.0 (sysName, defaults to “something”)

These OIDs are reset to the default settings after a factory reset. Other OIDs are also supported in the above MIBs but have read access only.

PLC Changes

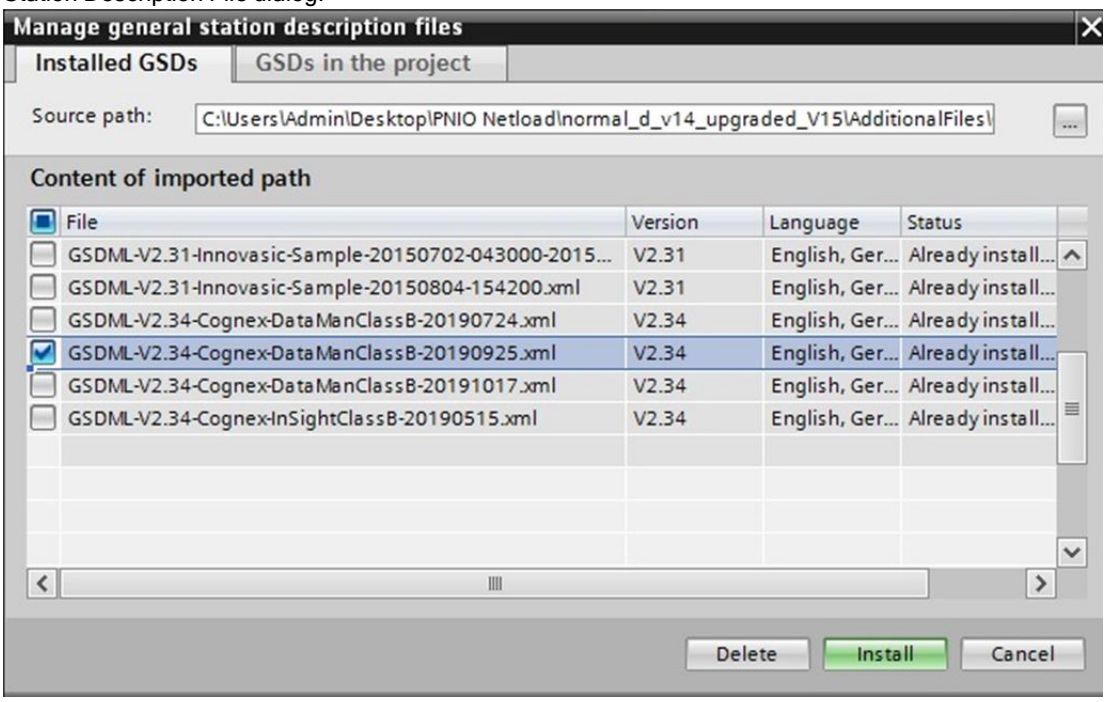
When upgrading the firmware on a DataMan reader to firmware version 6.1.8 or higher, you need to remove the reader from the PLC hardware configuration, replace the General Station Description (GSD) file, then re-add the “CC-B” version of the reader to the hardware configuration.

From the point of view of the PLC, a Class A device is replaced with a Class B device, which fundamentally changes how the PLC communicates with the reader. The PLC treats the reader as a completely different device.

Follow these steps to replace the reader in the hardware configuration of the PLC while maintaining the IO module addressing scheme to ensure that the logic in the project is not affected.

Note: It is highly recommended to follow this process for each reader being upgraded, one at a time.

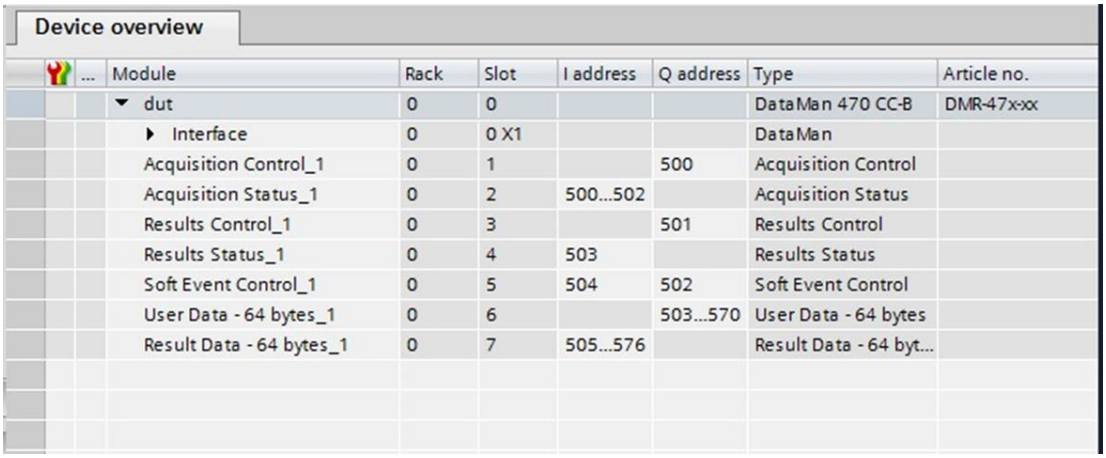
1. Go into the TIA Portal software and install the new GSD file from the Options drop down menu -> Manage General Station Description File dialog.



> **Note**: If you are adding onto an existing hardware configuration and are not removing any existing DataMan readers, skip steps 2, 3, and 5.

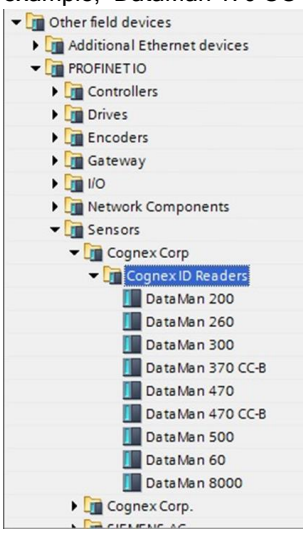
2. Select the source path in the Manage General Station Description Files dialog and browse to the location of the DataMan GSD file: *C:\Program Files (x86)\Cognex\DataMan\DataMan Software <version>\Tools\Profinet*

> **Tip**: The TIA software automatically assigns IO module addresses when a new system is added to the hardware configuration. To prevent different addresses from being assigned when the class B system is added, take a screenshot of the original IO Module addressing. This allows the new reader's IO modules to be mapped to the correct address space in the PLC, and prevent broken logic in the project:



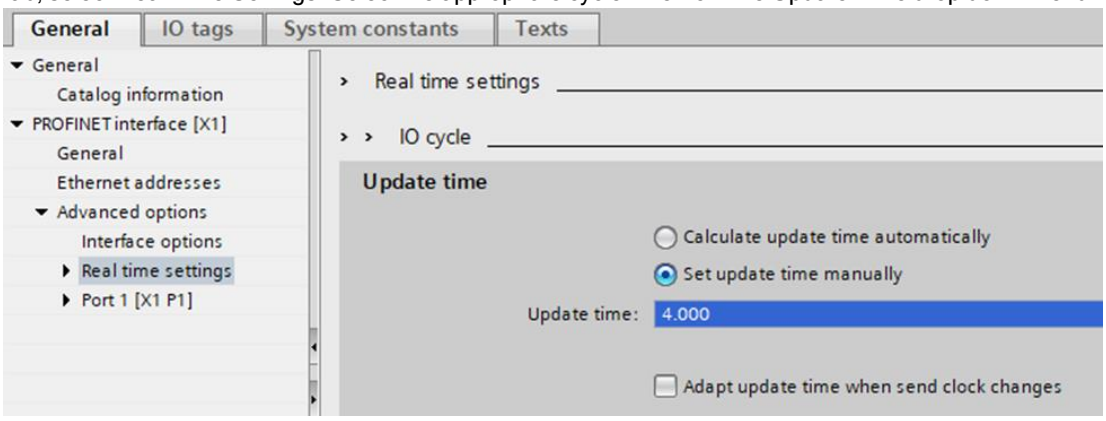
3. Remove the existing DataMan system from the hardware configuration.

4. Expand the catalog in the TIA Portal software, select the “CC-B” version and add it to your configuration. For example, “DataMan 470 CC-B”.



5. Confirm that the IO Module addressing matches what the original vision system was mapped to. If they are different, refer to the screenshot taken in step 2 to restore the IO module addresses to their previous values.

6. To change the cycle time to 8 ms, or to another value, select the reader from the Device view, and on the General tab, select Real Time Settings. Select the appropriate cycle time from the Update Time drop down menu:



> **Note**: The new GSD file has a faster cycle time, defaulting to 4 ms from 8 ms. While this shortens the network update interval, it also doubles the amount of PROFINET traffic from this device.

7. Compile and download the changes to the PLC.

iQ Sensor Solution

iQ Sensor Solution (iQSS) is an engineering tool offered by Mitsubishi Electric. It allows the intuitive configuration and maintenance of sensors. If iQSS is enabled in the Setup tool, DataMan supports setting and monitoring of the following device settings via iQSS:

- IP address and subnet mask
- The selected industrial protocol (SLMP, CC-Link IE Field Basic, or None).
- SLMP scanner parameters and device address mappings

Disabling iQSS blocks any attempts to remotely get or set these device settings.

Note: If you have a wireless DataMan reader, read the section Industrial Protocols for the Wireless DataMan.

DMCC

The following commands can be used to enable or disable the iQ Sensor Solution. The commands can be issued via RS-232 or Telnet connection.

Note: Because you have to make changes to the Telnet client provided by Windows to communicate with DataMan, it is recommended you use third party clients such as PuTTY.

Enable:

```
||>SET IQSS.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET IQSS.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

Choose language:

English:

```
||>SET IQSS.LANGUAGE 0
||>CONFIG.SAVE
||>REBOOT
```

Japanese:

```
||>SET IQSS.LANGUAGE 1
||>CONFIG.SAVE
||>REBOOT
```

Reader Configuration Code

Scanning the following reader configuration codes will enable/disable iQ Sensor Solution for your corded reader.

Enable:

Disable:

Scanning the following reader configuration codes will set the language for the iQSS on your DataMan reader.

English:

Japanese:

Setup Tool

The iQ Sensor Solution can be enabled by checking the box iQ Sensor Solution in the **Communications** step under the **Ethernet** tab. Make sure to save the new selection by choosing **Save Settings** before disconnecting from the reader.

Overview

This topic describes how to configure the iQ Sensor Solution for DataMan devices, and transfer data between a reader and a Mitsubishi Automation Controller using GX Works2. The examples in this topic were written assuming the following components are being used:

- DataMan Reader
- Mitsubishi iQ-R, Q, L or F-series (3E Frame) Automation Controller

Note: FX Series (1E Frame) are not supported.

- Mitsubishi GX Works2 software (SWnDNC-GXW2-E) version 1.492 or later
- Cognex Language Package (English = CognexInsight_en.ipar; Japanese = CognexInsight_ja.ipar) included within Setup Tool

Note: iQ Sensor Solution is only available on Setup Tool using 6.1.5 or later versions.

Note: For more information on Mitsubishi iQ Sensor Solution, refer to Mitsubishi's iQ Sensor Solution Reference Manual.

Mitsubishi iQ Sensor Solution is a factory protocol that enables the GX Works2 software to control readers over the SLMP Scanner protocol. With the iQ Sensor Solution, GX Work2 is able to discover readers on the network, read/write the reader's SLMP Scanner parameter settings and monitor the status of a reader.

Note: For more information on GX Works2, refer to the GX Works2 Operating Manual.

Discovering DataMan Readers on GX Work2

The following procedure assumes that the DataMan reader has not been previously configured for SLMP Scanner and is connected to the same Ethernet subnet as a supported Mitsubishi Automation Controller.

1. Launch GX Works2.
2. From the **Tool** menu, select **Register Profile**. The Register Profile window opens.
3. Create a new project in GX Works2.
4. Navigate to the Cognex DataMan Language Package (English = CognexInsight_en.ipar; Japanese = CognexInsight_ja.ipar), and click **Register**.

   Note: By default, the Cognex Language Package is installed to: C:\Program Files (x86)\Cognex\In-Sight\In-Sight Explorer 4.9.x\Factory Protocol Description\ESPP.

5. Select **Parameter** --> **PLC Parameter** on the **Project** view. The L Parameter Setting window is displayed.
6. Select the Built-in **Ethernet Port Settings** tab.
7. Check the **Set Open Setting** in Ethernet Configuration window checkbox. The following message will be displayed.
8. Click **Yes**; the **Open Settings** button will be changed to the **Ethenet Conf.** button.
9. Click the **Ethernet Conf.** button. The Ethernet Configuration window will be displayed.
10. Select **Ethernet Configuration** --> **Online** --> **Detect Now**; the following message is displayed.
11. Read the message and click **Yes**. The actual system configuration is reflected on the Ethernet Configuration window.
12. Select **Close with Reflecting the Setting**. The settings on the Ethernet Configuration window will be saved, and the system configuration setting is completed.

Note: Up to 16 readers may be displayed (in ascending order of their IP addresses) on the Ethernet Configuration window after executing the automatic detection of connected devices. If the **Detect Now** button is pressed, and a reader in the network is not iQSS-compatible, the reader will not appear in the Ethernet configuration window. If a reader is iQSS-compatible but no profile was previously added to GX Works2, "Module With No Profile Found" will be displayed.

Note: When a reader is detected in the actual system configuration, error information is displayed on the Output window when there is an error in the system configuration after executing the automatic detection of connected devices. Double-click the error on the Output window and correct the error at the error jump destination.

Note: For troubleshooting, refer to Mitsubishi's iQ Sensor Solution Reference Manual.

Monitoring the Status of the DataMan Reader

1. Launch GX Works2 and open a project that has the specified network configured.
2. From the Diagnostics menu, select **Sensor/Device Monitor**. The Module Selection (Sensor/Device Monitor) window will be displayed.
3. Click **OK**; the Status of readers connected to the Built-in Ethernet port LCPU will be displayed on the Sensor/Device Monitor window.
4. Select the desired reader from the "list of devices" or "device map area". The status of the reader will be displayed on the Monitor Information window.

   Note: Only one reader can be monitored in the Ethernet Configuration window at a time.

iQSS Error Codes

This section describes the iQSS error codes.

Error Code	Description
0	SLMP connection established
1	SLMP connection in progress
2	SLMP connection rejected
3	SLMP connection or configuration faulted
4	SLMP poll rate configured too low
5	SLMP connection idle
6	SLMP connection timed out
7	SLMP Status block error
8	SLMP Control block error
9	SLMP Output block error
10	SLMP Input block error
11	SLMP Command block error
12	SLMP Command Result block error
13	SLMP not started (if none or if different protocol like CC-Link is started)
14	SLMP settings require a reader reboot to become effective

CC-Link IE Field Basic

The CC-Link IE Field Basic 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. It is Mitsubishi Electric's publicly available, standardized communication format for communicating with iQ- Series, FX Series, iQ-R, iQ-F, Q, and L Series PLCs through Ethernet or serial connections. DataMan supports the CC-Link IE Field Basic on Ethernet only. For more information, see the Mitsubishi website.

By default the DataMan reader has the CC-Link Protocol disabled. The protocol can be enabled in the Setup Tool, via DMCC, or by scanning a parameter code.

Note: If you have a wireless reader, see *Industrial Protocols for the Wireless DataMan* on page 148

DMCC

The following commands can be used to enable or disable the CC-Link IE Field Basic. The commands can be issued via RS-232 or Telnet connection.

Note: Because you have to make changes to the Telnet client provided by Windows to communicate with the reader, it is recommended you use third party clients such as PuTTY.

Enable:

```
||>SET CC-LINK.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET CC-LINK.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

Reader Configuration Code

Scanning the following reader configuration codes will enable/disable CC-Link Protocol for your corded reader.

Enable:

Disable:

DataMan Setup Tool

The CC-Link IE Field Basic can be enabled by checking the box CC-Link in the **Communications** step under the **Ethernet tab** - > **Industrial Protocols**. Make sure to save the new selection by choosing **Save Settings** before disconnecting from the reader.

Note: You must reboot the reader for the change to take effect.

CC-Link IE Field Basic Modules

The CC-Link IE Field Basic implementation on DataMan consists of four I/O modules:

1. Control Block
2. Status Block
3. Output Data Block
4. Input Data Block

Signal Layout

Block name	Link Device Name	Data Length	Data Flow
Control Block	RY	64 bits (only the first 32 bits are used)	PLC to Reader
Status Block	RX	64 bits (only the first 32 bits are used)	Reader to PLC
Output Data Block	RWw	32 Words (64 bytes)	PLC to Reader
Input Data Block	RWr	32 Words (64 bytes)	Reader to PLC

Note: The number of link devices can be increased up to a factor of 4, if the reader is configured in GXWorks to occupy more than one station. This can be useful if inspection results or DMCC commands need to be transmitted that are longer than 62 bytes.

Control Block (RY)

The Control block contains bit type data. However, the block may be defined to exist in either bit or word memory in the PLC. This block consists of the control signals sent from the PLC to the reader. It is used by the PLC to initiate actions and acknowledge certain data transfers.Control Block

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				Results Ack	Buffer Results Enable	Trigger	Trigger Enable
Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**
Reserved							
Bit 23	**Bit 22**	**Bit 21**	**Bit 20**	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**
Reserved						Reserved	Reserved
Bit 31	**Bit 30**	**Bit 29**	**Bit 28**	**Bit 27**	**Bit 26**	**Bit 25**	**Bit 24**
Soft Event 7	Soft Event 6	Soft Event 5	Soft Event 4	Soft Event 3	Soft Event 2	Soft Event 1	Soft Event 0

Control Block (RY) Field Descriptions

Bit	Name	Description
0	Trigger Enable	This field is set to enable triggering via the *Trigger* bit. Clear this field to disable the network triggering mechanism.

Bit	Name	Description
1	Trigger	Setting this bit triggers an acquisition. **Note**: The *Trigger Ready* bit must be set high before triggering an acquisition.
2	Buffer Results Enable	When this bit is set, each read result set *(ResultID*, *ResultCode*, *ResultLength* and *ResultData* fields) will be held in the *Output Block* until it is acknowledged. Once acknowledged, the next set of read results will be made available from the buffer. If new read results arrive before the earlier set is acknowledged the new set will be queued in the reader's buffer. The reader can buffer up to 50 and the base station can buffer up to 500 sets of read results. See Operation for a description of the acknowledgement handshake sequence.
3	ResultsAck	Set by the PLC to acknowledge that it has received the latest results (*ResultID*, *ResultCode*, *ResultLength* and *ResultData* fields). When the reader sees this bit transition from 0->1 it clears the *ResultsAvailable* bit. This forms a logical handshake between the PLC and reader. If result buffering is enabled, the acknowledgement will cause the next set of queued results to be moved from the buffer. See Operation for a description of the acknowledgement handshake sequence.
4-15	Reserved	Future use
16	Reserved	Future use
17	Reserved	Future use
18-23	Reserved	Future use
24-31	SoftEvents	Bits act as virtual discrete inputs. When a bit transitions from 0->1 the associated action is executed. After executing the action the reader sets the corresponding *SoftEventAck* to signal that the action is complete. This forms a logical handshake between the PLC and reader. Bit0: Train code Bit1: Train match string Bit2: Train focus Bit3: Train brightness Bit4: Un-Train Bit5: Reserved (future use) Bit6: Execute DMCC command Bit7: Set match string

Status Block (RX)

The status block contains bit type data. However, the block may be defined to exist in either bit or word memory in the PLC. This block consists of the status signals sent from the reader to the PLC. It is used by the reader to signal status and handshake certain data transfers.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				Missed Acq	Acquiring	Trigger Ack	Trigger Ready
Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**
Genereal Fault	Reserved			Results Available	Results Buffer Overrun	Decode Complete Toggle	Decoding
Bit 23	**Bit 22**	**Bit 21**	**Bit 20**	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**
Reserved						Reserved	Reserved
Bit 31	**Bit 30**	**Bit 29**	**Bit 28**	**Bit 27**	**Bit 26**	**Bit 25**	**Bit 24**

Soft Event Ack 7	Soft Event Ack 6	Soft Event Ack 5	Soft Event Ack 4	Soft Event Ack3	Soft Event Ack 2	Soft Event Ack 1	Soft Event Ack 0

Status Block (RX) Field Descriptions

Bit	Name	Description
0	Trigger Ready	Indicates when the reader is ready to accept a new *Trigger*. The reader sets this bit when *TriggerEnable* has been set and the reader is ready to accept a new trigger.
1	Trigger Ack	Indicates when the reader recognizes that *Trigger* has been set. This bit will remain set until the *Trigger* bit has been cleared.
2	Acquiring	Set to indicate that the reader is in the process of acquiring an image. **Note**: For CF26, this bit is “Reserved”.
3	Missed Acq	Indicates that the reader missed a requested acquisition trigger. The bit is cleared when the next acquisition is issued.
4-7	Reserved	Future use
8	Decoding	Deprecated. The value of the Decoding bit is always the same as the Acquiring bit.
9	Decode Complete Toggle	Indicates new result data is available. Bit toggles state (0->1 or 1->0) each time new result data becomes available.
10	Results Buffer Overrun	Set to indicate that the reader has discarded a set of read results because the PLC has not acknowledged the earlier results. Cleared when the next set of result data is successfully queued in the buffer. This bit only has meaning if result buffering is enabled.
11	Results Available	Set to indicate that new result data is available. Bit will remain set until acknowledged with *ResultsAck* even if additional new read results become available.
12-14	Reserved	Future use
15	General Fault	Set to indicate that an Ethernet communications fault has occurred. Currently only used by soft event operations. Bit will remain set until the next successful soft event or until TriggerEnable is set low and then high again.
16	Reserved	Future use.
17	Reserved	Future use
18-23	Reserved	Future use
24-31	SoftEvent Ack	Set to indicate that the reader has completed the soft event action. Bit will remain set until the corresponding SoftEvent bit is cleared. This forms a logical handshake between the PLC and reader. Bit0: Ack train code Bit1: Ack train match string Bit2: Ack train focus Bit3: Ack train brightness Bit4: Ack untrain Bit5: Reserved (future use) Bit6: Ack Execute DMCC command Bit7: Ack set match string

Input Data Block (RWw)

The Input Data block contains word type data. This is data sent from the PLC to the reader. The block consists of user defined data that may be used as input to the acquisition/decode operation.

Word 0	Word 1	Word 2..N
UserDataOption	User Data Length	User Data

Input Data Block (RWw) Field Descriptions

<table>
<tr><th>Word</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>User Data Option</td><td>Optional user data information (16-bit Integer). Used by the “Set Match String” SoftEvent to assign the match string to the specified symbology. Depending on your PLC software, you can move an integer value or a binary to this field:
<table>
<tr><th>Integer</th><th>Binary</th><th>Description</th></tr>
<tr><td>0</td><td>0000 0000 0000 0000</td><td>Assign string to all smybologies.</td></tr>
<tr><td>1</td><td>0000 0000 0000 0001</td><td>Assign string to 2D codes.</td></tr>
<tr><td>2</td><td>0000 0000 0000 0010</td><td>Assign string to QR codes.</td></tr>
<tr><td>3</td><td>0000 0000 0000 0011</td><td>Assign string to 1D/stacked/postal codes.</td></tr>
</table></td></tr>
<tr><td>1</td><td>User Data Length</td><td>Number of bytes of valid data actually contained in the UserData field.</td></tr>
<tr><td>2..N</td><td>User Data</td><td>User defined data that may be used as an input to the acquisition/decode.</td></tr>
</table>

Output Data Block (RWr)

The Output Data block contains word type data. This is data sent from the reader to the PLC. The block consists primarily of read result data.

Word 0	Word 1	Word 2	Word 3	Word 4	Word 5..N
Reserved	Trigger ID	Result ID	Result Code	Result Length	Result Data

Output Data Block (RWr) Field Descriptions

Word	Name	Description
0	Reserved	Future use
1	Trigger ID	Trigger identifier. Identifier of the next trigger to be issued. Used to match issued triggers with result data that is received later. This same value will be returned as the ResultID of the corresponding read.
2	Result ID	Result set identifier. This is the value of TriggerID when the corresponding trigger was issued. Used to match up triggers with corresponding result data.

Word	Name	Description
3	Result Code	Indicates the success or failure of the read that produced this result set. Bit0: 1=Read, 0=No read Bit1: 1=Validated, 0=Not Validated Bit2: 1=Verified, 0=Not Verified Bit3: 1=Acquisition trigger overrun Bit4: 1=Acquisition buffer overrun Bit5: Reserved Bit 6: 1=result is a DMCC response, 0=result is a read result. For more information, see *DMCC Result Feedback from PLC* on page 96. Bit7-15: Reserved (future use)
4	Result Data Length	Number of bytes of valid data actually in the ResultData field.
5..N	Result Data	Result data from this acquisition/decode.

CC-Link IE Field Basic Operation

This section describes how the features of CC-Link IE Field Basic operate.

Results Buffering

The Results Buffering feature allows a queue for decode results. When enabled, the feature queues a finite number of result data sets until the PLC (client) has time to read them. This smooths out data flow if the PLC slows down for short periods or if there are surges in read activity.

If the feature is enabled, the reader overlaps acquisition and decode operations. This can lead to faster over all trigger rates. When reads occur faster than results can be sent out (the queue is full), the most recent results are discarded until space becomes available in the results queue.

If the feature is disabled, the more recent results overwrite the earlier results. The most recent results are kept. Earlier results are discarded if the PLC does not read them fast enough.

If the queue overflows while the feature is disabled:

- The difference between the TriggerID and ResultID values becomes greater than 1. This difference represents the number of reads that occurred but could not be queued because the queue was full.
- The number of lost reads equals TriggerID - ResultID - 1.
- After the next read, the ResultID value returns to the typical operating value of TriggerID - 1.

Result Fragmentation

The Result Fragmentation feature splits decode result strings into fragments if the strings are too large for the Results Data field. The fragments are then pushed to the buffer like normal results. Then, the PLC reads the fragments one by one.

Result Fragmentation depends on Result Buffering. This means that results are only fragmented if Result Buffering is enabled.

Note: For CC-Link IE Field Basic, you have to enable Result Fragmentation manually.

Enabling Result Fragmentation

To enable or disable Result Fragmentation, you have the following options:

- Use the FFP.RESULT-FRAG DMCC.
- Use SoftEvent 6 to enable the Results Buffering and Result Fragmentation features from the PLC.

Operation of Result Fragmentation

The PLC does not automatically reassemble the result string. Once the PLC reads a result fragment, you have to copy the fragment to a new user-defined variable. After this, you have to set and clear the ResultAck bit, which updates the Decode Results with the next fragment. This process continues until all fragments have been sent to the PLC.

In contrast, if Results Buffering is disabled, large decode result strings are cut off to fit into the Results Data field, which means that the end of the string gets lost.

Identifying Decode Result Chunks

Decode result fragments look like normal results. To identify result fragments, look at the Result ID:

- The Result ID of the result fragments are the same.
- The Result ID of the last result fragment is different from the Result ID of the next result.

Independent of whether results fragmentation is enabled, the Result Length always represents the number of characters to be read from the Results Data plus the remaining characters that did not fit into the Results Data buffer. With results fragmentation enabled, the remaining characters will be found in follow-up fragments, the remaining characters are discarded.

You can identify the first and last fragments of a result as follows:

- The first fragment has a different Result ID from the previous Result ID.
- The Result Length of the last fragment is lower or equal to the Result Data buffer size.
- All fragments of a result share the same Result ID.

Maximum Result Data Length

Without results fragmentation, result data is truncated to fit into the results data buffer size. Depending on the protocol, the result buffer size can be fixed or configurable.

The Result Fragmentation feature allows processing of up to 50 times longer results, depending on the results buffering queue size, with an absolute limit of 65535 characters.

The maximum number of result characters for CC-Link IE Field Basic is 1014 bytes (fixed).

DMCC Result Feedback from PLC

The DMCC Result Feedback feature allows the reader to send DMCC responses back to the PLC.

By default, this feature is off. To enable the feature, use the `FFP.DMCC-RESULTS` DMCC command. Changing this configuration does not require you to reboot or reconnect reader.

When you enable the feature, the reader sends DMCC responses to the PLC using the same mechanism it uses to send read results. To distinguish DMCC results from read results, bit 6 is of the ResultCode field is set to 1. The following Output Data Block fields contain DMCC information instead of read results:

Word	Name	Description

2	ResultID	Contains a DMCC result code: • 0: no error • 100: unidentified error • 101: invalid command • 102: invalid parameter • 103: incorrect checksum • 104: parameter rejected/altered due to reader state • 105: reader unavailable (offline)
3	ResultCode	Bit 6 is set to 1 to indicate a DMCC response.
4	ResultLength	Actual number of bytes of DMCC result data contained in the *ResultData* field (16-bit Integer).
5..N	ResultData	DMCC result data (array of bytes).

You can use DMCC Result Feedback in combination with Results Fragmentation.

SLMP Protocol

The SLMP Protocol uses standard Ethernet hardware and software to exchange I/O data, alarms, and diagnostics. It is Mitsubishi Electric's publicly available, standardized communication format for communicating with Q, iQ and L Series PLCs through Ethernet or serial connections. DataMan supports SLMP Protocol on Ethernet only.

By default, the DataMan has SLMP Protocol disabled. The protocol can be enabled in the Setup Tool, via DMCC, or by scanning a configuration code.

Note: If you have a wireless DataMan reader, see section *Industrial Protocols for the Wireless DataMan* on page 148.

DMCC

The following commands can be used to enable/disable SLMP Protocol. The commands can be issued via RS-232 or Telnet connection.

Note: Use a third party Telnet client such as PuTTY to communicate with your DataMan reader.

Enable:

```
||>SET SLMP-PROTOCOL.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET SLMP-PROTOCOL.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

Reader Configuration Code

Scanning the following reader configuration codes enables/disables SLMP Protocol for your corded reader.

Note: You must reboot the device for the change to take effect.

Enable:

Disable:

Scanning the following reader configuration codes enables/disables SLMP protocol on your DataMan 8000 Base Station.

Note: You must reboot the device for the change to take effect.

Enable: Disable:

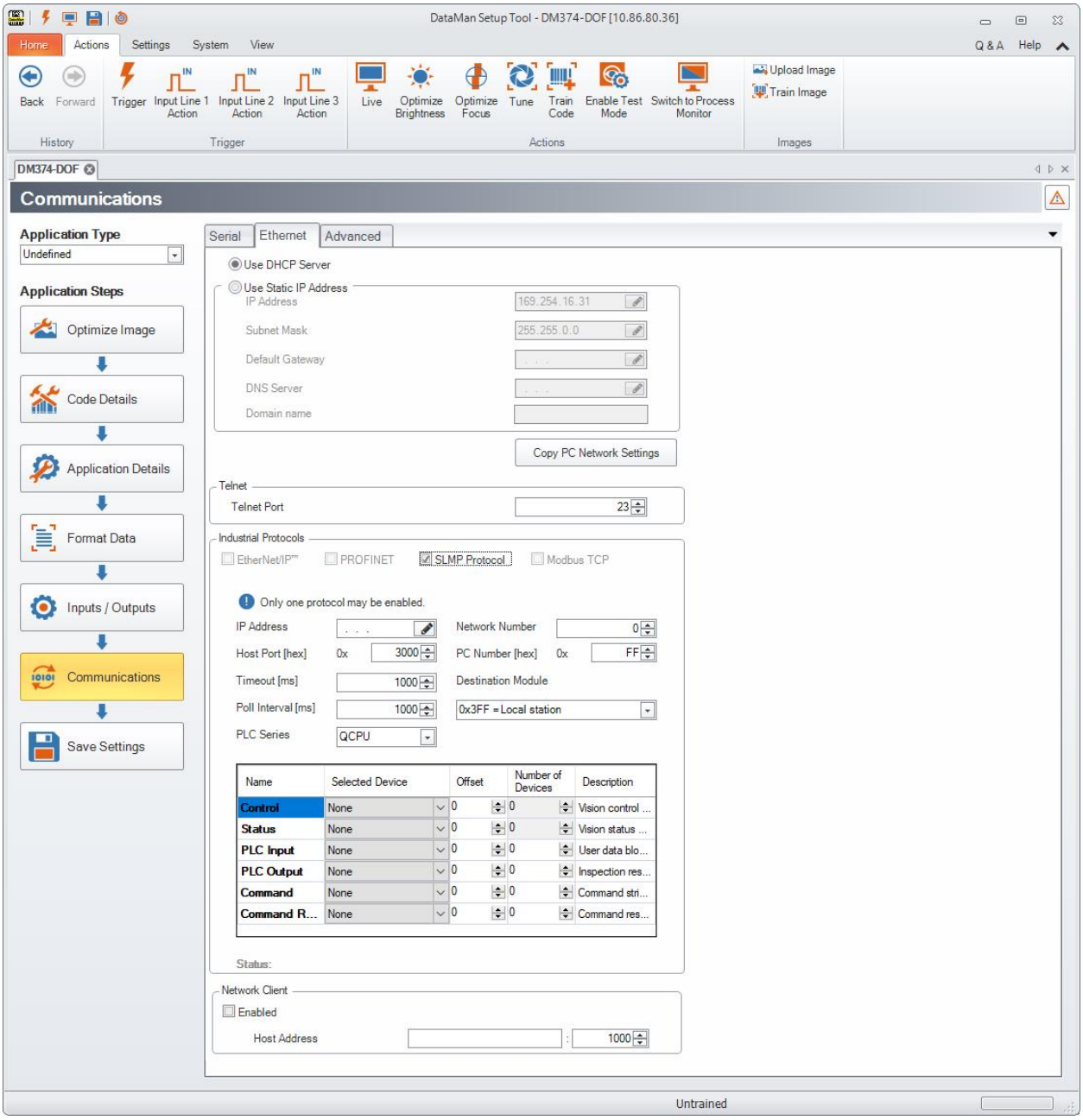
SLMP Protocol Scanner

SLMP Protocol on DataMan is implemented as a client type device also referred to as a scanner. The DataMan reader initiates all communication in the form of read and write requests. The PLC acts as a passive server reacting to the read and write requests. Since the PLC cannot initiate communication, it relies on the reader to periodically ask (scan) the PLC for any actions or information that the PLC requires, such as triggering or retrieving read results.

Preparing to Use SLMP

By default, SLMP Protocol is not enabled on the DataMan reader. The protocol must be enabled and the protocol configuration parameters must be set to correctly interact with a PLC. Protocol configuration is accomplished via the DataMan Setup Tool.

In the Setup Tool's **Communications** application step's **Ethernet** tab, check **SLMP Protocol** to enable this industrial protocol.



Make sure to save the new selection by clicking the **Save Settings** button in the upper toolbar before disconnecting from the reader.

ⓘ **Note**: The new settings take effect only after the reader is rebooted.

SLMP Protocol configuration consists of two aspects: defining the network information and defining the data to be exchanged. All configuration parameters are accessed via the SLMP Protocol tab.

You must modify the **IP Address** to match the address of your PLC. In addition, modify **Network Number**, **PC Number** and **Destination Module** if they differ from your network.

SLMP Network Configuration

The network configuration defines all the information that the DataMan reader needs to establish a connection with a PLC.

Name	Default	Range	Description
IP Address	<empty >	Any valid IP address	IP Address of the PLC to connect to.
Host Port (Hex)	0x3000	Any Hex port number 0x1000-0xFFFF	Port number of the SLMP Protocol channel on the PLC.
Timeout (ms)	1000	5 - 30000	Time to in milliseconds for a response from the PLC to an SLMP Protocol message.
Poll Interval (ms)	1000	10 - 30000	Requested time in milliseconds between successive polls of the Control Block from the PLC.
PLC Series	QCPU	QCPU or LCPU	Defines frame type used. Currently only 3E supported.
Network Number	0	0 - 239	SLMP Protocol network number to communicate with (0 = local network).
PC Number	0xFF	0x01 -0x78 = station on CC-Link IE field network adapter 0x7E = Master station on CC-Link IE field network 0xFF = Direct connect to local station	Station identifier on the specified network of the destination module.
Destination Module	0x3FF	0x3ff = Local station (default) 0x3d0 = Control system CPU 0x3d1 = Standby system CPU 0x3d2 = System A CPU 0x3d3 = System B CPU 0x3e0 = CPU 1 0x3e1 = CPU 2 0x3e2 = CPU 3 0x3e3 = CPU 4	Module identifier of the device to connect to.

Data Block Configuration

The data block configuration defines the data that will be exchanged between the DataMan reader and the PLC. Six data blocks are available. Each block has a predefined function.

Not all data blocks are required. Configure only those data blocks which your application needs. Typically the Control and Status blocks are defined because they control most data flow. However, there are some use cases where even these blocks are not required.

A data block is configured by defining the PLC Device type (that is, memory type), Device offset and Number of Devices contained in the data block. If either the Device type or Number of Devices is undefined, that block will not be used. That is, no data is exchanged for that block.

Block Name	Supported Device Types	Offset	Number of Devices
Control	<none>, D, W, R, ZR, M, X, Y, L, F, B	0 - 65535	0, if type <none> 32, if bit type 2, if word type (read-only)
Status	<none>, D, W, R, ZR, M, X, Y, L, F, B	0 - 65535	0, if type <none> 32, if bit type 2, if word type (read-only)
PLC Input	None, D, W, R, ZR	0 - 65535	0 - 960
PLC Output	None, D, W, R, ZR	0 - 65535	0 - 960
Command	None, D, W, R, ZR	0 - 65535	0 - 960
Command Result	None, D, W, R, ZR	0 - 65535	0 - 960

Interface

This section describes the interface to the DataMan reader as seen by the PLC via SLMP Protocol. The interface model consists of six data blocks grouped in three logical pairs:

- Control and Status
- PLC Input and PLC Output
- Command and Command Response

Not all of the blocks are required. You may select which blocks are appropriate for your particular application. However, Control and Status will generally be included for most applications.

You can define the starting address and device type for each interface block that you choose to use in your application. Undefined blocks will not be exchanged. For any transfer (read or write) the entire block is sent, even if only one field within the block has changed value. The protocol implementation will minimize network use by grouping as many value changes as logically possible into a single transfer.

Control Block

The Control block contains bit type data. However, the block may be defined to exist in either bit or word memory in the PLC. This block consists of the control signals sent from the PLC to the reader. It is used by the PLC to initiate actions and acknowledge certain data transfers.Control Block

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				Results Ack	Buffer Results Enable	Trigger	Trigger Enable
Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**
Reserved							
Bit 23	**Bit 22**	**Bit 21**	**Bit 20**	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**
Reserved						Initiate String Cmd	Set User Data
Bit 31	**Bit 30**	**Bit 29**	**Bit 28**	**Bit 27**	**Bit 26**	**Bit 25**	**Bit 24**
SoftEvent 7	SoftEvent 6	SoftEvent 5	SoftEvent 4	SoftEvent 3	SoftEvent 2	SoftEvent 1	SoftEvent 0

Control Block Field Descriptions

Bit	Name	Description
0	Trigger Enable	This field is set to enable triggering via the *Trigger* bit. Clear this field to disable the network triggering mechanism.
1	Trigger	Setting this bit triggers an acquisition. **Note**: The *Trigger Ready* bit must be set high before triggering an acquisition.
2	Buffer Results Enable	When this bit is set, each read result set *(ResultID*, *ResultCode*, *ResultLength* and *ResultData* fields) will be held in the *Output Block* until it is acknowledged. Once acknowledged, the next set of read results will be made available from the buffer. If new read results arrive before the earlier set is acknowledged, the new set will be queued in the reader's buffer. The reader can buffer up to 50 and the Base Station can buffer up to 500 sets of read results. See Operation for a description of the acknowledgement handshake sequence.
3	ResultsAck	Set by the PLC to acknowledge that it has received the latest results (*ResultID*, *ResultCode*, *ResultLength* and *ResultData* fields). When the reader sees this bit transition from 0 to 1, it clears the *ResultsAvailable* bit. This forms a logical handshake between the PLC and reader. If result buffering is enabled, the acknowledgement will cause the next set of queued results to be moved from the buffer. See section Operation for a description of the acknowledgement handshake sequence.
4-15	Reserved	Reserved for future use.
16	SetUserData	Set by the PLC to signal that new *UserData* is available. After reading the new *UserData*, the reader sets Set *UserDataAck* to signal that the transfer is complete. This forms a logical handshake between the PLC and reader.
17	Initiate StringCmd	Set by the PLC to signal that a new *StringCommand* is available. After processing the command, the reader sets *StringCmdAck* to signal that the command result is available. This forms a logical handshake between the PLC and reader.
18-23	Reserved	Future use

Bit	Name	Description
24-31	SoftEvents	Bits act as virtual discrete inputs. When a bit transitions from 0 to 1, the associated action is executed. After executing the action, the reader sets the corresponding *SoftEventAck* to signal that the action is complete. This forms a logical handshake between the PLC and reader. Bit0: Train code Bit1: Train match string Bit2: Train focus Bit3: Train brightness Bit4: Un-Train Bit5: Reserved (future use) Bit6: Execute DMCC command Bit7: Set match string

Status Block

The status block contains bit type data. The block can also be defined to exist in either bit or word memory in the PLC. This block consists of the status signals sent from the reader to the PLC. The reader uses it to signal status and handshake certain data transfers.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				Missed Acq	Acquiring	Trigger Ack	Trigger Ready
Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**
General Fault	Reserved			Results Available	Results Buffer Overrun	Decode Complete Toggle	Decoding
Bit 23	**Bit 22**	**Bit 21**	**Bit 20**	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**
Reserved						String Cmd Ack	Set User Data Ack
Bit 31	**Bit 30**	**Bit 29**	**Bit 28**	**Bit 27**	**Bit 26**	**Bit 25**	**Bit 24**
SoftEvent Ack 7	SoftEvent Ack 6	SoftEvent Ack 5	SoftEvent Ack 4	SoftEvent Ack3	SoftEvent Ack 2	SoftEvent Ack 1	SoftEvent Ack 0

Status Block Field Descriptions

Bit	Name	Description
0	Trigger Ready	Indicates when the reader is ready to accept a new *Trigger*. The reader sets this bit when *TriggerEnable* has been set and the reader is ready to accept a new trigger.
1	Trigger Ack	Indicates when the reader recognizes that *Trigger* has been set. This bit remains set until the *Trigger* bit is clear.
2	Acquiring	Set to indicate that the reader is in the process of acquiring an image.
3	Missed Acq	Indicates that the reader missed a requested acquisition trigger. The bit is cleared when the next acquisition is issued.
4-7	Reserved	Future use
8	Decoding	Deprecated. The value of the Decoding bit is always the same as the Acquiring bit.

Bit	Name	Description
9	Decode Complete Toggle	Indicates new result data is available. Bit toggles state (0 to 1 or 1 to 0) each time new result data becomes available.
10	Results Buffer Overrun	Set to indicate that the reader has discarded a set of read results because the PLC has not acknowledged the earlier results. Cleared when the next set of result data is successfully queued in the buffer. This bit only has meaning if result buffering is enabled.
11	Results Available	Set to indicate that new result data is available. Bit will remain set until acknowledged with *ResultsAck* even if additional new read results become available.
12-14	Reserved	Future use
15	General Fault	Set to indicate that an Ethernet communications fault has occurred. Currently only used by SoftEvent operations. Bit will remain set until the next successful SoftEvent or until *TriggerEnable* is set low and then high again.
16	Set User Data Ack	Set to indicate that the reader has received new *UserData*. Bit will remain set until the corresponding *SetUserData* bit is cleared. This forms a logical handshake between the PLC and reader.
17	String Cmd Ack	Set to indicate that the reader has completed processing the latest string command and that the command response is available. Bit will remain set until the corresponding *InitiateStringCmd* bit is cleared. This forms a logical handshake between the PLC and reader.
18-23	Reserved	Future use
24-31	SoftEvent Ack	Set to indicate that the reader has completed the SoftEvent action. Bit will remain set until the corresponding SoftEvent bit is cleared. This forms a logical handshake between the PLC and reader. Bit0: Ack train code Bit1: Ack train match string Bit2: Ack train focus Bit3: Ack train brightness Bit4: Ack untrain Bit5: Reserved (future use) Bit6: Ack Execute DMCC command Bit7: Ack set match string

Input Data Block

The Input Data block contains word type data. This is data sent from the PLC to the reader. The block consists of user defined data that may be used as input to the acquisition/decode operation.

Word 0	Word 1	Word 2..N
UserDataOption	User Data Length	User Data

Input Data Block Field Descriptions

<table>
<tr><th>Word</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>User Data Option</td><td>Optional user data information (16-bit Integer). Used by the “Set Match String” SoftEvent to assign the match string to the specified symbology. Depending on your PLC software, you can move an integer value or a binary to this field:
<table>
<tr><th>Integer</th><th>Binary</th><th>Description</th></tr>
<tr><td>0</td><td>0000 0000 0000 0000</td><td>Assign string to all smybologies.</td></tr>
<tr><td>1</td><td>0000 0000 0000 0001</td><td>Assign string to 2D codes.</td></tr>
<tr><td>2</td><td>0000 0000 0000 0010</td><td>Assign string to QR codes.</td></tr>
<tr><td>3</td><td>0000 0000 0000 0011</td><td>Assign string to 1D/stacked/postal codes.</td></tr>
</table></td></tr>
<tr><td>1</td><td>User Data Length</td><td>Number of bytes of valid data actually contained in the UserData field.</td></tr>
<tr><td>2..N</td><td>User Data</td><td>User defined data that may be used as an input to the acquisition/decode.</td></tr>
</table>

Output Data Block

The Output Data block contains word type data. This is data sent from the reader to the PLC. The block consists primarily of read result data.

Word 0	Word 1	Word 2	Word 3	Word 4	Word 5..N
Reserved	Trigger ID	Result ID	Result Code	Result Length	Result Data

Output Data Block Field Descriptions

Word	Name	Description
0	Reserved	Future use
1	Trigger ID	Trigger identifier. Identifier of the next trigger to be issued. Used to match issued triggers with result data that is received later. This same value will be returned as the ResultID of the corresponding read.
2	Result ID	Result set identifier. This is the value of TriggerID when the corresponding trigger was issued. Used to match up triggers with corresponding result data.
3	Result Code	Indicates the success or failure of the read that produced this result set. Bit0: 1=Read, 0=No read Bit1: 1=Validated, 0=Not Validated Bit2: 1=Verified, 0=Not Verified Bit3: 1=Acquisition trigger overrun Bit4: 1=Acquisition buffer overrun Bit5: Reserved Bit 6: 1=result is a DMCC response, 0=result is a read result. For more information, see *DMCC Result Feedback from PLC* on page 112. Bit7-15: Reserved (future use)
4	Result Data Length	Number of bytes of valid data actually in the ResultData field.
5..N	Result Data	Result data from this acquisition/decode.

String Command Block

The String Command block contains word type data. This is data sent from the PLC to the reader. The block is used to transport string based commands (DMCC) to the reader.

Note: Do not send string commands that change the reader configuration at the same time as reads are being triggered. Changing configuration during acquisition/decode can lead to unpredictable results.

Word 0	Word 1..N
Length	String Command

String Command Block Field Descriptions

Word	Name	Description
0	Length	Number of bytes of valid data in the *StringCommand* field.
1..N	String Command	ASCII text string containing the command to execute. No null termination required.

String Command Result Block

The String Command Result block contains word type data. This is data sent from the reader to the PLC. The block is used to transport the response from string based commands (DMCC) to the PLC.

Word 0	Word 1	Word 2..N
Result Code	Length	String Command Result

String Command Result Block Field Descriptions

Word	Name	Description
0	Result Code	Code value indicating the success or failure of the command. Refer to the *Command Reference*, available through the Windows **Start** menu or the DataMan Setup Tool Help menu.
1	Length	Number of bytes of valid data in the *StringCommand* field.
2..N	String Command Result	ASCII text string containing the command to execute. No null termination required.

SLMP Operation

SLMP Protocol is a command/response based protocol. All communications are originated from the DataMan reader. The reader must send read requests to the PLC at a periodic interval to detect changes in the control bits.

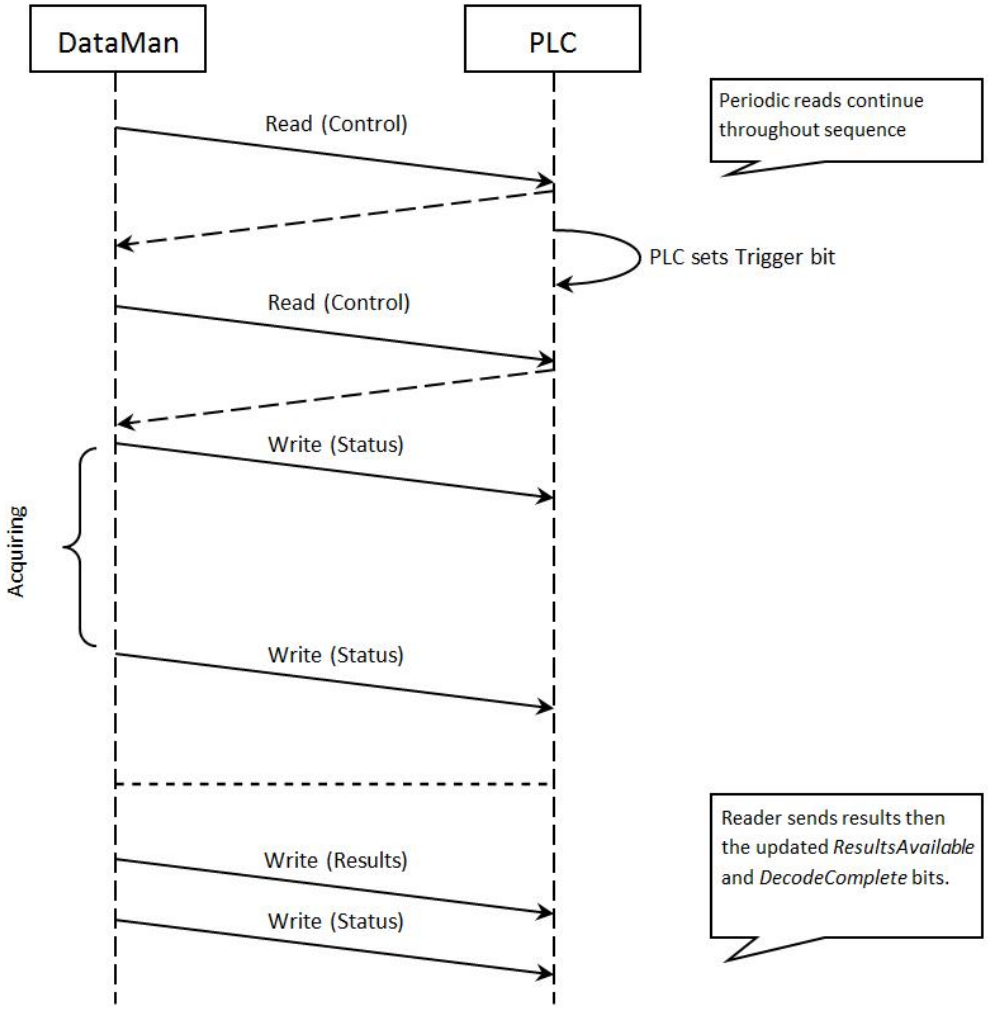
Scanning

To initiate actions or control data transfer, the PLC changes the state of certain bits of the Control block. Since only the reader can initiate communications, the reader scans (that is, reads the Control block from the PLC) at a periodic rate. This rate is user-defined.

After each scan, the reader will process changes in state of the bits in the Control block. Some state changes require additional communications with the PLC, such as writing updated acknowledge bit values or reading a new string command. The reader handles these additional communications automatically. Other state changes initiate activities such as triggering a read or executing a SoftEvent. The reader performs the requested action and later reports the results.

For any transfer (read or write), the entire interface block is sent, even if only one field within the block has changed value. The protocol implementation will minimize network usage by grouping as many value changes as logically possible into a single transfer.

Typical Sequence Diagram



Handshaking

A number of actions are accomplished by means of a logical handshake between the reader and the PLC: triggering, transferring results, executing SoftEvents, string commands, and so on. This ensures that both sides of a transaction know the state of the operation on the opposite side. Network transmission delays always introduce a finite time delay in transfer data and signals. Without this handshaking, one side of a transaction might not detect a signal state change on the other side. Any operation that has both an initiating signal and corresponding acknowledge signal uses this basic handshake procedure.

The procedure involves a four-way handshake.

1. Assert signal
2. Signal acknowledge
3. De-assert signal
4. De-assert acknowledge

The requesting device asserts the signal to request an action (set bit 0 to 1). When the target device detects the signal and the requested operation is complete, it asserts the corresponding acknowledge (set bit 0 to 1). When the requesting device detects the acknowledge, it de-asserts the original signal (1 to 0). Finally, when the target device detects the original signal

de-asserted, it de-asserts its acknowledge (bit 0 to 1). To function correctly, both sides must see the complete assert/de-assert cycle (0 to 1 and 1 to 0). The requesting device should not initiate a subsequent request until the cycle completes.

Acquisition Sequence

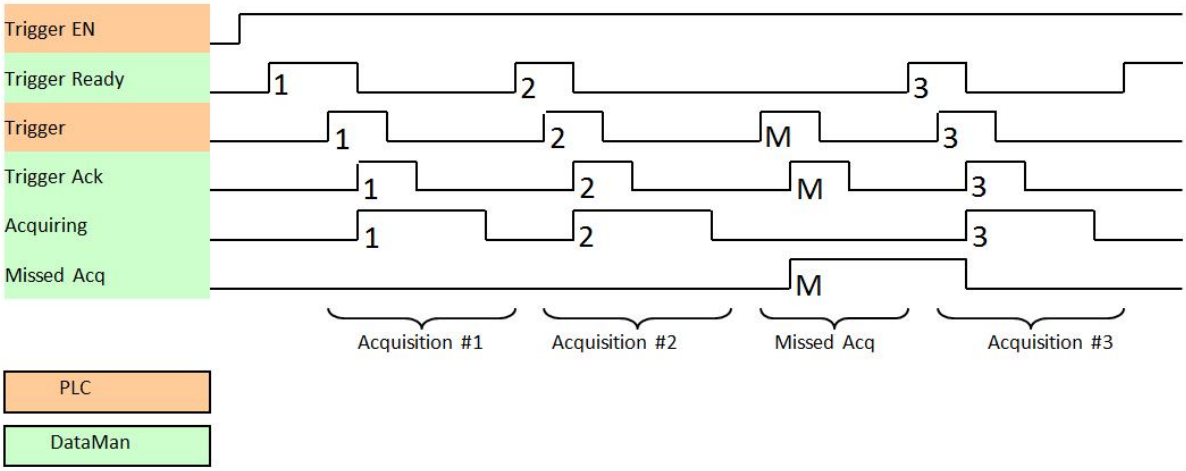
DataMan can be triggered to acquire images by several methods. It can be done via the SLMP Protocol by setting the *Trigger* bit or issuing a trigger String Command, by DMCC (Telnet), or hardwired trigger signal. This section describes the *Trigger* bit method.

On startup the *TriggerEnable* will be False. It must be set to True to enable triggering via the SLMP Protocol *Trigger* bit. When the device is ready to accept triggers, the reader will set the *TriggerReady* bit to True.

While the *TriggerReady* bit is True, each time the reader detects the *Trigger* bit change from 0 to 1, it initiates a read. Make sure that the *Trigger* bit is held in the new state until that same state value is seen in the *TriggerAck* bit. This is a necessary handshake to guarantee that the reader detects the trigger.

During an acquisition, the *TriggerReady* bit will be cleared and the *Acquiring* bit will be set to True. When the acquisition is completed, the *Acquiring* bit will be cleared. When the device is ready to begin another image acquisition, the *TriggerReady* bit will again be set to True.

The reader allows acquisitions to overlap with the decoding of previous acquisitions. The Trigger Ready bit is set high after the acquisition is complete, while decoding may still be in progress. The Decoding bit is deprecated and only mirrors the behavior of the Acquiring bit. If trigger queuing is active or other trigger sources can interfere, the Trigger Ready bit may not be reliable. As result, conflicting trigger overruns may occur, which are reported in Bit 3 of the Result Code word of the Output Data Block.



To force a reset of the trigger mechanism, set the *TriggerEnable* to False until *TriggerReady* is also set to False. Then, *TriggerEnable* can be set to True to re-enable acquisition.

As a special case, an acquisition can be cancelled by clearing the *Trigger* signal before the read operation is complete. This allows for canceling reads in Presentation and Manual mode if no code is in the field of view. To ensure that a read is not unintentionally cancelled, make sure that the PLC holds the Trigger signal True until both *TriggerAck* and *ResultsAvailable* are True (or *DecodeComplete* toggles state).

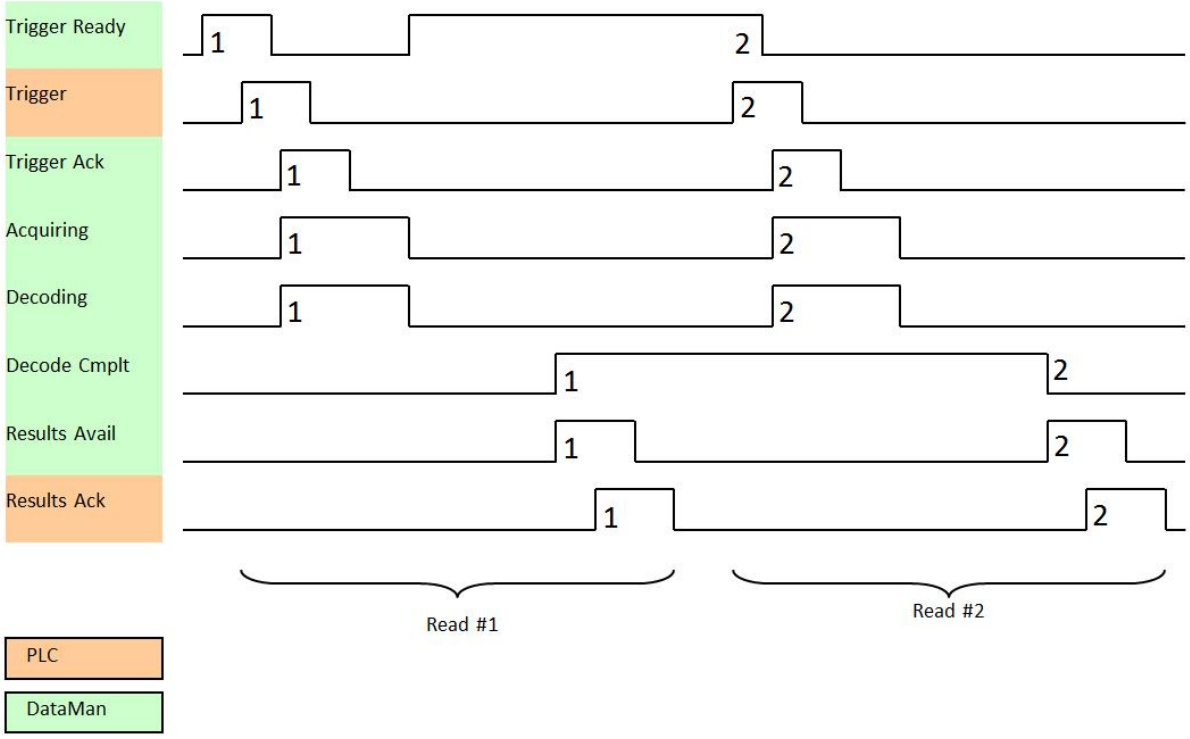
Decode / Result Sequence

After an image is acquired, it is decoded. When the decoding is complete and a result is ready to be delivered, the reader toggles the *DecodeComplete* bit of the *DecodeStatusRegister*.

Decode results are reported asynchronously to the Output Data block. If BufferResultsEnable is set to False, then the decode results are immediately placed into DecodeResults, and the Results Available bit is set to True.

Note: The only way to ensure that you are not losing results is to use BufferResults. Make sure that your application can read each DecodeResult in time if you do not use BufferResultsEnable.

If *ResultsBufferEnable* is set to True, the new results are queued in a buffer and *DecodeComplete* is toggled. The earlier read results remain in the Output Data block until the PLC acknowledges them. After the acknowledgment handshake, if there are more results in the queue, the next set of results will be placed in the Output Data block and *ResultsAvailable* is set to True.



Results Buffering

The Results Buffering feature allows a queue for decode results. When enabled, the feature queues a finite number of result data sets until the PLC (client) has time to read them. This smooths out data flow if the PLC slows down for short periods or if there are surges in read activity.

If the feature is enabled, the reader overlaps acquisition and decode operations. This can lead to faster over all trigger rates. When reads occur faster than results can be sent out (the queue is full), the most recent results are discarded until space becomes available in the results queue.

If the feature is disabled, the more recent results overwrite the earlier results. The most recent results are kept. Earlier results are discarded if the PLC does not read them fast enough.

If the queue overflows while the feature is disabled:

- The difference between the TriggerID and ResultID values becomes greater than 1. This difference represents the number of reads that occurred but could not be queued because the queue was full.
- The number of lost reads equals TriggerID - ResultID - 1.
- After the next read, the ResultID value returns to the typical operating value of TriggerID - 1.

Result Fragmentation

The Result Fragmentation feature splits decode result strings into fragments if the strings are too large for the Results Data field. The fragments are then pushed to the buffer like normal results. Then, the PLC reads the fragments one by one.

Result Fragmentation depends on Result Buffering. This means that results are only fragmented if Result Buffering is enabled.

Note: For SLMP, you have to enable Result Fragmentation manually.

Enabling Result Fragmentation

To enable or disable Result Fragmentation, you have the following options:

- Use the FFP.RESULT-FRAG DMCC.
- Use SoftEvent 6 to enable the Results Buffering and Result Fragmentation features from the PLC.

Operation of Result Fragmentation

The PLC does not automatically reassemble the result string. Once the PLC reads a result fragment, you have to copy the fragment to a new user-defined variable. After this, you have to set and clear the ResultAck bit, which updates the Decode Results with the next fragment. This process continues until all fragments have been sent to the PLC.

In contrast, if Results Buffering is disabled, large decode result strings are cut off to fit into the Results Data field, which means that the end of the string gets lost.

Identifying Decode Result Chunks

Decode result fragments look like normal results. To identify result fragments, look at the Result ID:

- The Result ID of the result fragments are the same.
- The Result ID of the last result fragment is different from the Result ID of the next result.

Independent of whether results fragmentation is enabled, the Result Length always represents the number of characters to be read from the Results Data plus the remaining characters that did not fit into the Results Data buffer. With results fragmentation enabled, the remaining characters will be found in follow-up fragments, the remaining characters are discarded.

You can identify the first and last fragments of a result as follows:

- The first fragment has a different Result ID from the previous Result ID.
- The Result Length of the last fragment is lower or equal to the Result Data buffer size.
- All fragments of a result share the same Result ID.

Maximum Result Data Length

Without results fragmentation, result data is truncated to fit into the results data buffer size. Depending on the protocol, the result buffer size can be fixed or configurable.

The Result Fragmentation feature allows processing of up to 50 times longer results, depending on the results buffering queue size, with an absolute limit of 65535 characters.

The maximum number of result characters for SLMP is 3820 bytes (configurable).

SoftEvents

SoftEvents act as “virtual” inputs. When the value of a *SoftEvent* bit changes from 0 to 1, the action associated with the event is executed. When the action completes, the corresponding *SoftEventAck* bit will change from 0 to 1 to signal completion.

The *SoftEvent* and *SoftEventAck* form a logical handshake. After *SoftEventAck* changes to 1, the original *SoftEvent* should be set back to 0. When that occurs, *SoftEventAck* will automatically be set back to 0.

Note: Do not execute SoftEvents that change the reader configuration at the same time that reads are being triggered. Changing configuration during acquisition/decode can lead to unpredictable results.

Note: The “ExecuteDMCC” and “SetMatchString” SoftEvent actions require user supplied data. This data must be written to the *UserData* and *UserDataLength* area of the Input Data block prior to invoking the SoftEvent. Since both of these SoftEvents depend on the *UserData*, only one may be invoked at a time.

String Commands

The String Command feature allows you to execute string-based DMCCs over the SLMP or the Modbus TCP protocol connection. The DMCC is sent to the reader through the String Command block. The DMCC result is returned through the String Command Result block. Initiating a command and notification of completion is accomplished by signaling bits in the Control and Status blocks.

To execute a DMCC, the command string is placed in the data field of the String Command block. The command string consists of standard ASCII text. The same command format is used for a serial (RS-232) or Telnet connection. The string does not need to be terminated with a null character. Instead, the length of the string (that is, the number of ASCII characters) is placed in the length field of the String Command block.

After executing the DMCC, the result string is returned in the String Command Result block. Similar to the original command, the result string consists of ASCII characters in the same format that is returned through a serial or Telnet connection. Also, there is no terminating null character. Instead the length of the result is returned in the Command String Result length field. The Command String Result block also contains a numeric result code. This allows you to determine the success or failure of the command without having to parse the text string. The values of the result code are defined in the DMCC documentation.

DMCC Result Feedback from PLC

The DMCC Result Feedback feature allows the reader to send DMCC responses back to the PLC.

Note: The SLMP protocol already offers the String Commands feature to initiate DMCC commands with result feedback. The DMCC Result Feedback feature is also available, but the String Commands feature is preferred because it is simpler to use.

By default, this feature is off. To enable the feature, use the `FFP.DMCC-RESULTS` DMCC command. Changing this configuration does not require you to reboot or reconnect reader.

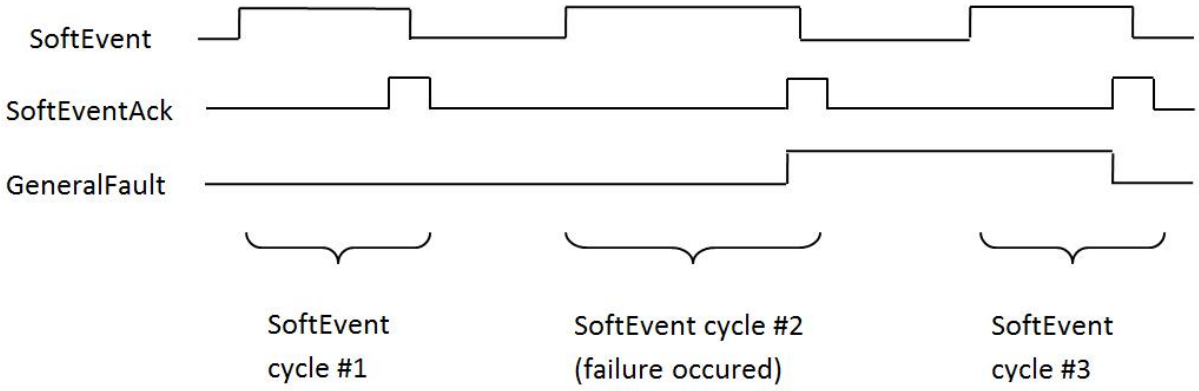
When you enable the feature, the reader sends DMCC responses to the PLC using the same mechanism it uses to send read results. To distinguish DMCC results from read results, bit 6 is of the ResultCode field is set to 1. The following Output Data Block fields contain DMCC information instead of read results:

Word	Name	Description
2	ResultID	Contains a DMCC result code: • 0: no error • 100: unidentified error • 101: invalid command • 102: invalid parameter • 103: incorrect checksum • 104: parameter rejected/altered due to reader state • 105: reader unavailable (offline)
3	ResultCode	Bit 6 is set to 1 to indicate a DMCC response.
4	ResultLength	Actual number of bytes of DMCC result data contained in the *ResultData* field (16-bit Integer).
5..N	ResultData	DMCC result data (array of bytes).

You can use DMCC Result Feedback in combination with Results Fragmentation.

General Fault Indicator

When an SLMP Protocol communication-related fault occurs, the “GeneralFault” bit will change from 0 to 1. Currently the only fault conditions supported are SoftEvent operations. If a SoftEvent operation fails, the fault bit will be set. The fault bit will remain set until the next successful SoftEvent operation, or, until *TriggerEnable* is set to 0 and then back to 1.



SLMP Examples

Included with the DataMan Setup Tool installer is an example PLC program created with Mitsubishi (GX Works2) software. This program demonstrates the DataMan ID readers’ capabilities and proper operation. You can do the same operations by using more advanced features and efficient programming practices with Mitsubishi PLCs. For demonstration purposes, the sample program is used.

Function

The example application demonstrates the following operations:

1. Triggering a read
2. Getting read results
3. Executing string commands (DMCC)
4. Executing SoftEvent operations
    a. Train code
    b. Train match string
    c. Train focus
    d. Train brightness
    e. Un-train
    f. Execute DMCC
    g. Set match string

The “Main” program contains a PLC ladder rung to invoke each of these operations. The operation is invoked by toggling the control bit on the rung from 0 to 1. This invokes the associated subroutine to perform the operation. When the operation is complete, the subroutine sets the control bit back to 0.

Triggering a Read

The example provides two trigger options; “Continuous Trigger” and “Single Trigger”. As the name implies, enabling the “Continuous Trigger” bit will invoke a continuous series of read operations. Once enabled, the “Continuous Trigger” control bit will remain set until you disable it. The “Single Trigger” control bit invokes a single read operation. This control bit will automatically be cleared when the read is completed.

Primarily, the trigger subroutine manages the trigger handshake operation between the PLC and the reader. The control *Trigger* bit is set, the PLC waits for the corresponding *TriggerAck* status bit from the reader, and the control *Trigger* bit is reset. Refer to a description of handshaking in section Operation.

The trigger subroutine contains a delay timer. This is not required for operation. It exists simply to add an adjustable artificial delay between reads for demonstration purposes.

Getting Read Results

For this example the operation of triggering a read and getting read results were intentionally separated. This is to support the situation where the PLC is not the source of the read trigger. For example, the reader may be configured to use a hardware trigger. In such a case, only the get results subroutine would be needed.

Like the triggering subroutine, the get results subroutine manages the results handshake operation between the PLC and the reader. However, it also copies the result data to internal storage. The routine waits for the *ResultsAvailable* status bit to become active, it copies the result data to internal storage, and then executes the *ResultsAck* handshake. Refer to a description of handshaking in section Operation.

The read result consists of a *ResultCode*, *ResultLength*, and *ResultData*. Refer to section Output Data Block Field Descriptions for details of the ResultCode values. The *ResultLength* field indicates how many bytes of actual result data exist in the *ResultData* field. The subroutine converts this byte length to word length before copying the results to internal storage.

The get results subroutine gathers read statistics: number of good reads, number of no-reads, and so on. This is not required for operation. It is simply for demonstration purposes.

Execute String Commands (DMCC)

The string command feature provides a simple way to invoke DMCC commands from the PLC. The command format and command result format is exactly identical to that used for serial or Telnet DMCC operation.

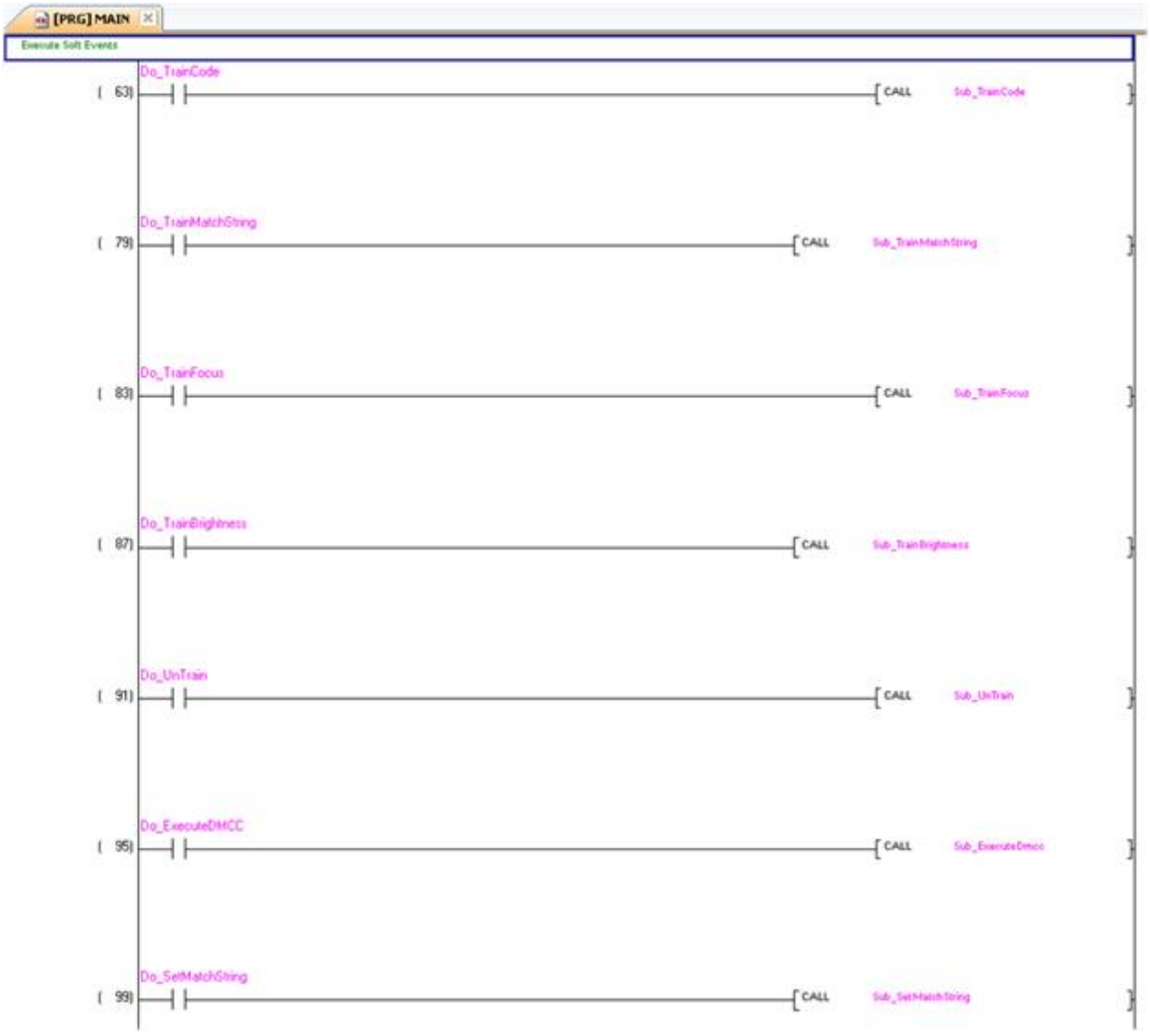
This subroutine copies an example DMCC command (||>GET CAMERA.EXPOSURE) to the String Command block and then manages the string command handshake operation between the PLC and the reader to invoke the command and retrieve the command result. Any valid DMCC command may be invoked with this mechanism. Refer to the DataMan *Command Reference* document available through the Windows **Start** menu or the DataMan Setup Tool **Help** menu.

Execute SoftEvents

SoftEvents are used to invoke a predefined action. Each SoftEvent is essentially a virtual input signal. Each of the SoftEvent subroutines manages the handshake operation between the PLC and the reader to invoke the predefined action. The associated action is invoked when the *SoftEvent* bit toggles from 0 to 1. The subroutine then watches for the associated

SoftEventAck bit from the reader which signals that the action is complete. For a description of handshaking, see section Operation.

Note: The “Execute DMCC” and “Set Match String” SoftEvents make use of the Input Data block. The subroutine for these two events copies the relevant data into the User Data fields of the Input Data block and then invokes the User Data subroutine to transfer the data to the reader. The actual SoftEvent action is invoked only after the user data is transferred. The user data needs to be transferred before invoking either of these events.



Note: The “Train Match String” SoftEvent only prepares the training mechanism. The actual training occurs on the next read operation. Therefore, a trigger must be issued following “Train Match String”.

ModbusTCP

Modbus is an application layer protocol. It provides client/server communication between devices connected to different types of buses or networks. Modbus is a request/response protocol, and its services are specified by using function codes.

Modbus TCP provides the Modbus protocol using TCP/IP. System port 502 is reserved for Modbus communication. It uses standard Ethernet hardware and software to exchange I/O data and diagnostics. DataMan provides Modbus TCP server functionality only.

By default, DataMan has the Modbus TCP protocol disabled. The protocol can be enabled in the Setup Tool, via DMCC, or by scanning a configuration code.

Note: If you have a wireless DataMan reader, see section *Industrial Protocols for the Wireless DataMan* on page 148

DMCC

The following commands can be used to enable/disable Modbus TCP. The commands can be issued via RS-232 or Telnet connection.

Note: Use a third party Telnet client such as PuTTY to communicate with your DataMan reader.

Enable:

```
||>SET MODBUSTCP.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET MODBUSTCP.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

Reader Configuration Code

Scanning the following reader configuration codes enables/disables Modbus TCP for your corded reader.

Note: You must reboot the device for the change to take effect.

Enable:

Disable:

Scanning the following reader configuration codes enables/disables Modbus TCP for your DataMan 8000 Base Station.

Note: You must reboot the device for the change to take effect.

Enable: Disable:

Setup Tool

Modbus TCP can be enabled by checking **Enabled** on the Industrial Protocols pane's Modbus TCP tab. Make sure to save the new selection by choosing "Save Settings" before disconnecting from the reader.

Note: You must reboot your reader for the new settings to take effect.

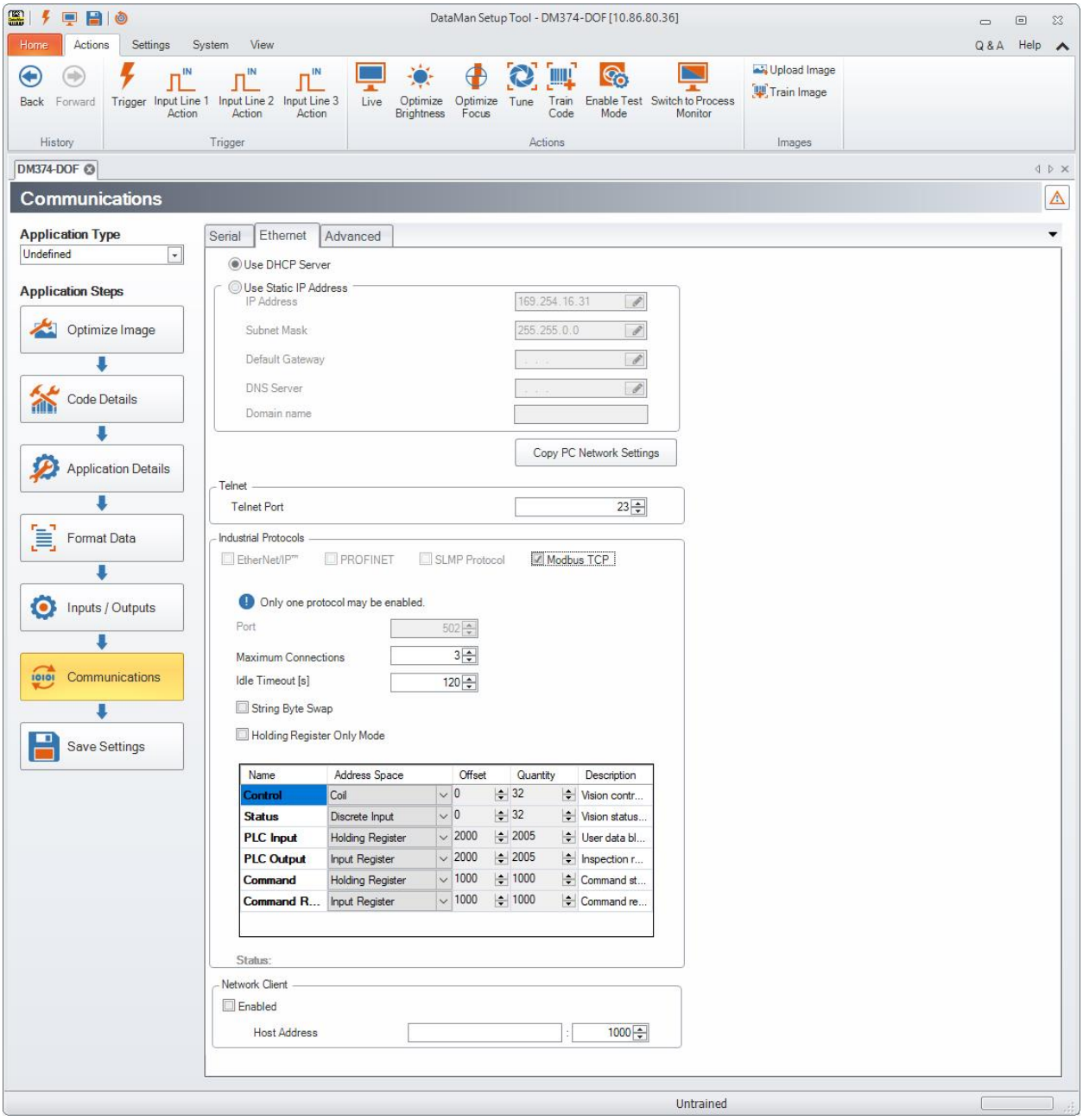
Modbus TCP Handler

Modbus TCP on DataMan is implemented as a server type device. All communication is initiated by the PLC in the form of read and write requests. The PLC acts as a client which actively sends read and write requests.

Preparing to Use Modbus TCP

By default, Modbus TCP is not enabled on the DataMan reader. The protocol must be enabled and the protocol configuration parameters must be set to correctly interact with a PLC. Protocol configuration is accomplished via the DataMan Setup Tool.

In the Setup Tool's **Communications** application step's **Ethernet** tab, check **Modbus TCP** to enable this industrial protocol.



Make sure to save the new selection by clicking the **Save Settings** button in the upper toolbar before disconnecting from the reader.

ⓘ **Note**: The new settings take effect only after the reader is rebooted.

Network Configuration

The network configuration defines all the information that the DataMan reader needs to establish a connection with a PLC. In most cases the default values may be used and no changes are need.

Name	Default	Range	Description
Host Port	502	Fixed	Port number where Modbus TCP can be accessed on this reader.

Max Connections	3	1-6	Maximum number of simultaneous Modbus TCP connections.
Idle Timeout (seconds)	120	1-3600	Timeout period after which the Modbus TCP connection will be closed. If no traffic is received on a Modbus TCP connection for this amount of time, the connection will automatically be closed.
String byte swap	False	True / False	String byte swap enable. If set to True, bytes within each register that forms a string will be swapped.
Holding Register Only Mode	False	True / False	Holding Register Only Mode enable. If set to True, all data blocks will be mapped to the Holding Register space.

Data Block Configuration

The data block configuration defines the data that will be exchanged between the DataMan reader and the PLC. Six data blocks are available. Each block has a predefined function.

DataMan only supports Modbus TCP server operation. For standard server operation, only two configuration options exist. By default the ‘Control’ and ‘Status’ data blocks are located in bit address space (Coil and Discrete Input). If needed, one or both of these data blocks may be redefined to exist in register address space (Holding Register and Input Register). All other data block configurations are fixed.

Standard Block Configuration

Block Name	Address Space	Offset	Schneider Form Addressing	Quantity
Control	Coil or Holding Register	0	000000 - 000031 400000 - 400001	32, if coil 2, if holding register
Status	Discrete Input or Input Register	0	100000 - 100031 300000 - 300001	32, if discrete input 2, if input register
PLC Input	Holding Register	2000	402000 - 404004	1 - 2005
PLC Output	Input Register	2000	302000 - 304004	1 - 2005
Command String	Holding Register	1000	401000 - 401999	1 - 1000
Command String Result	Input Register	1000	301000 - 301999	1 - 1000

Note: The 6 digit 0-based Schneider representation is used here.

DataMan also supports an alternate block configuration. A number of PLCs can only access the ModbusTCP ‘Holding Register’ address space. The alternate 'Holding Register Only' configuration exists to accommodate these PLCs. In this alternate configuration, all reader input and output data is mapped to the ModbusTCP ‘Holding Register’ address space. To enable this alternate mode, perform one of the following options:

- Check the “Holding Register Only” box on the ModbusTCP configuration screen.
- Use the DMCC, which switches on or off the alternate mode (Holding Register Only mode).

Enable:

```
||>SET MODBUSTCP-HOLDING-ONLY ON
```

Disable:

```
||>SET MODBUSTCP-HOLDING-ONLY OFF
```

- scan the Reader Programming Codes
  - for the wireless reader

Enable: Disable:

  - for the corded reader

Enable: Disable:

Holding Register Only" Block Configuration

Block Name	Address Space	Offset	Schneider Form Addressing	Quantity
Control	Holding Register	0	400000 - 400001	2
Status	Holding Register	5000	405000 - 405001	2
PLC Input	Holding Register	2000	402000 - 404004	1 - 2005
PLC Output	Holding Register	7000	407000 - 409004	1 - 2005
Command String	Holding Register	1000	401000 - 401999	1 - 1000
Command String Result	Holding Register	6000	406000 - 406999	1 - 1000

Note: The 6 digit 0-based Schneider representation is used here.

Modbus TCP Interface Model

This section describes the interface to the DataMan reader as seen by the PLC via Modbus TCP. The interface model consists of 6 data blocks grouped in 3 logical pairs:

- Control and Status
- Input Data and Output Data
- String Command and String Response

Control Block

The Control block contains bit type data. This block consists of the control signals sent from the PLC to the reader. It is used by the PLC to initiate actions and acknowledge certain data transfers.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				Results Ack	Buffer Results Enable	Trigger	Trigger Enable

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reserved							
Bit 23	**Bit 22**	**Bit 21**	**Bit 20**	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**
Reserved						Initiate String Cmd	Set User Data
Bit 31	**Bit 30**	**Bit 29**	**Bit 28**	**Bit 27**	**Bit 26**	**Bit 25**	**Bit 24**
SoftEvent 7	SoftEvent 6	SoftEvent 5	SoftEvent 4	SoftEvent 3	SoftEvent 2	SoftEvent 1	SoftEvent 0

Control Block Field Descriptions

Bit	Name	Description
0	Trigger Enable	This field is set to enable triggering via the *Trigger* bit. Clear this field to disable the network triggering mechanism.
1	Trigger	Setting this bit triggers an acquisition. Note that the *Trigger Ready* bit must be set high before triggering an acquisition.
2	Buffer Results Enable	When this bit is set, each read result (*ResultID*, *ResultCode*, *ResultLength* and *ResultData* fields) will be held in the *Output Block* until it is acknowledged. Once acknowledged, the next set of read results will be made available from the buffer. If new read results arrive before the earlier set is acknowledged the new set will be queued in the reader's buffer. The reader can buffer up to 50 and the Base Station can buffer up to 500 sets of read results. See Operation for a description of the acknowledgement handshake sequence.
3	ResultsAck	Set by the PLC to acknowledge that it has received the latest results (*ResultID*, *ResultCode*, *ResultLength* and *ResultData* fields). When the reader sees this bit transition from 0 to 1 it clears the *ResultsAvailable* bit. This forms a logical handshake between the PLC and reader. If result buffering is enabled, the acknowledgement will cause the next set of queued results to be moved from the buffer. See section Operation for a description of the acknowledgement handshake sequence.
4-15	Reserved	Future use
16	SetUserData	Set by the PLC to signal that new *UserData* is available. After reading the new *UserData* the reader sets *SetUserDataAck* to signal that the transfer is complete. This forms a logical handshake between the PLC and reader.
17	Initiate StringCmd	Set by the PLC to signal that a new *StringCommand* is available. After processing the command, the reader sets *StringCmdAck* to signal that the command result is available. This forms a logical handshake between the PLC and reader.
18-23	Reserved	Future use

24-31	SoftEvents	Bits act as virtual discrete inputs. When a bit transitions from 0 -> 1 the associated action is executed. After executing the action the reader sets the corresponding *SoftEventAck* to signal that the action is complete. This forms a logical handshake between the PLC and reader. Bit0: Train code Bit1: Train match string Bit2: Train focus Bit3: Train brightness Bit4: Un-Train Bit5: Reserved (future use) Bit6: Execute DMCC command Bit7: Set match string

Status Block

The status block contains bit type data. This block consists of the status signals sent from the reader to the PLC. It is used by the reader to signal status and handshake certain data transfers.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				Missed Acq	Acquiring	Trigger Ack	Trigger Ready
Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**
General Fault	Reserved			Results Available	Results Buffer Overrun	Decode Complete Toggle	Decoding
Bit 23	**Bit 22**	**Bit 21**	**Bit 20**	**Bit 19**	**Bit 18**	**Bit 17**	**Bit 16**
Reserved						String Cmd Ack	Set User Data Ack
Bit 31	**Bit 30**	**Bit 29**	**Bit 28**	**Bit 27**	**Bit 26**	**Bit 25**	**Bit 24**
SoftEvent Ack 7	SoftEvent Ack 6	SoftEvent Ack 5	SoftEvent Ack 4	SoftEvent Ack3	SoftEvent Ack 2	SoftEvent Ack 1	SoftEvent Ack 0

Status Block Field Descriptions

Bit	Name	Description
0	Trigger Ready	Indicates when the reader is ready to accept a new *Trigger*. The reader sets this bit when *TriggerEnable* has been set and the reader is ready to accept a new trigger.
1	TriggerAck	Indicates when the reader recognizes that *Trigger* has been set. This bit will remain set until the *Trigger* bit has been cleared.
2	Acquiring	Set to indicate that the reader is in the process of acquiring an image.
3	Missed Acq	Indicates that the reader missed a requested acquisition trigger. The bit is cleared when the next acquisition is issued.
4-7	Reserved	Future use
8	Decoding	Deprecated. The value of the Decoding bit is always the same as the Acquiring bit.

9	Decode Complete Toggle	Indicates new result data is available. Bit toggles state (0 -> 1 or 1 -> 0) each time new result data becomes available.
10	Results Buffer Overrun	Set to indicate that the reader has discarded a set of read results because the PLC has not acknowledged the earlier results. Cleared when the next set of result data is successfully queued in the buffer. This bit only has meaning if result buffering is enabled.
11	Results Available	Set to indicate that new result data is available. Bit will remain set until acknowledged with *ResultsAck* even if additional new read results become available.
12-14	Reserved	Future use
15	General Fault	Set to indicate that an Ethernet communications fault has occurred. Currently only used by SoftEvent operations. Bit will remain set until the next successful SoftEvent or until *TriggerEnable* is set low and then high again.
16	Set User Data Ack	Set to indicate that the reader has received new *UserData*. Bit will remain set until the corresponding *SetUserData* bit is cleared. This forms a logical handshake between the PLC and reader.
17	String Cmd Ack	Set to indicate that the reader has completed processing the latest string command and that the command response is available. Bit will remain set until the corresponding *InitiateStringCmd* bit is cleared. This forms a logical handshake between the PLC and reader.
18-23	Reserved	Future use
24-31	SoftEvent Ack	Set to indicate that the reader has completed the SoftEvent action. Bit will remain set until the corresponding SoftEvent bit is cleared. This forms a logical handshake between the PLC and reader. Bit0: Ack train code Bit1: Ack train match string Bit2: Ack train focus Bit3: Ack train brightness Bit4: Ack untrain Bit5: Reserved (future use) Bit6: Ack Execute DMCC command Bit7: Ack set match string

Input Data Block

The Input Data block is sent from the PLC to the reader. The block consists of user defined data that may be used as input to the acquisition/decode operation.

Word 0	Word 1	Word 2..N
User Data Option	User Data Length	User Data

Input Data Block Field Descriptions

Word	Name	Description

<table>
<tr><td>0</td><td>User Data Option</td><td>Optional user data information (16-bit Integer). Used by the “Set Match String” SoftEvent to assign the match string to the specified symbology. Depending on your PLC software, you can move an integer value or a binary to this field:
<table>
<tr><th>Integer</th><th>Binary</th><th>Description</th></tr>
<tr><td>0</td><td>0000 0000 0000 0000</td><td>Assign string to all smybologies.</td></tr>
<tr><td>1</td><td>0000 0000 0000 0001</td><td>Assign string to 2D codes.</td></tr>
<tr><td>2</td><td>0000 0000 0000 0010</td><td>Assign string to QR codes.</td></tr>
<tr><td>3</td><td>0000 0000 0000 0011</td><td>Assign string to 1D/stacked/postal codes.</td></tr>
</table></td></tr>
<tr><td>1</td><td>User Data Length</td><td>Number of bytes of valid data actually contained in the UserData field.</td></tr>
<tr><td>2..N</td><td>User Data</td><td>User defined data that may be used as an input to the acquisition/decode.</td></tr>
</table>

Output Data Block

The Output Data block is sent from the reader to the PLC. The block consists primarily of read result data.

Word 0	Word 1	Word 2	Word 3	Word 4	Word 5...n
Reserved	Trigger ID	Result ID	Result Code	Result Length	Result Data

Output Data Block Field Descriptions

Word	Name	Description
0	Reserved	Future use
1	Trigger ID	Trigger identifier. Identifier of the next trigger to be issued. Used to match issued triggers with result data that is received later. This same value will be returned as the *ResultID* of the corresponding read.
2	Result ID	Result identifier. This is the value of *TriggerID* when the corresponding trigger was issued. Used to match up triggers with corresponding result data.
3	Result Code	Indicates the success or failure of the read that produced this result set. Bit0: 1=Read, 0=No read Bit1: 1=Validated, 0=Not Validated Bit2: 1=Verified, 0=Not Verified Bit3: 1=Acquisition trigger overrun Bit4: 1=Acquisition buffer overrun Bit5: Reserved Bit 6: 1=result is a DMCC response, 0=result is a read result. For more information, see *DMCC Result Feedback from PLC* on page 130. Bit7-15: Reserved (future use)
4	Result Data Length	Number of bytes of valid data actually in the *ResultData* field.
5...n	Result Data	Result data from this acquisition/decode. Formatted as ASCII text with two characters per 16-bit register. No terminating null character.

String Command Block

The String Command block is sent from the PLC to the reader. The block is used to transport string based commands (DMCC) to the reader.

Note: Do not send string commands that change the reader configuration at the same time as reads are being triggered. Changing configuration during acquisition/decode can lead to unpredictable results.

Word 0	Word 1...n
Length	String Command

String Command Block Field Descriptions

Word	Name	Description
0	Length	Number of bytes of valid data in the *StringCommand* field.
1..N	String Command	ASCII text string containing the command to execute. No null termination required.

String Command Result Block

The String Command Result block is sent from the reader to the PLC. The block is used to transport the response from string based commands (DMCC) to the PLC.

Word 0	Word 1	Word 2..N
Result Code	Length	String Command Result

String Command Result Block Field Descriptions

Word	Name	Description
0	Result Code	Code value indicating the success or failure of the command. Refer to the ***Command Reference***, available through the Windows **Start** menu or the DataMan Setup Tool **Help** menu, for specific values.
1	Length	Number of bytes of valid data in the *StringCommand* field.
2..N	String Command Result	ASCII text string containing the command to execute. No null termination required.

Modbus TCP Operation

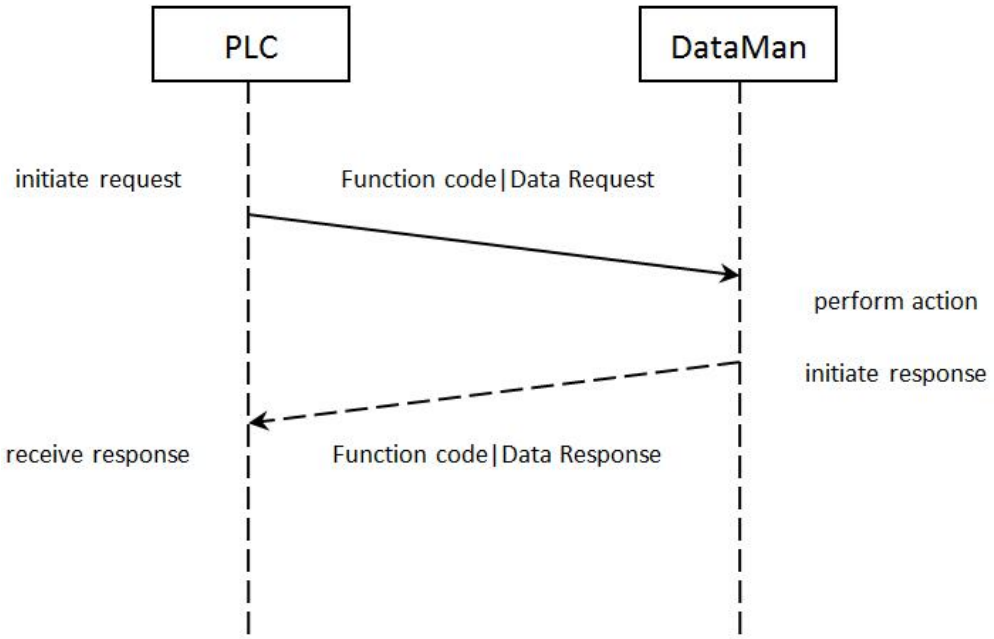
Modbus TCP is a request/response based protocol. All communications are originated from the PLC. The reader acts as server.

Requests

To initiate actions or control data transfer, the PLC changes the state of certain bits of the Control block and sends requests to the reader.

After each request, the reader will process changes in state of the bits in the Control block. Some state changes require additional communications with the PLC, such as writing updated acknowledge bit values or reading a new string command. These additional communications are handled automatically by the reader. Other state changes initiate activities such as triggering a read or executing a SoftEvent. The reader performs the requested action and later reports the results.

Typical Sequence Diagram



Handshaking

A number of actions are accomplished by means of a logical handshake between the reader and the PLC (triggering, transferring results, executing SoftEvents, string commands, and so on). This is done to ensure that both sides of a transaction know the state of the operation on the opposite side. Network transmission delays will always introduce a finite time delay in transfer data and signals. Without this handshaking, it is possible that one side of a transaction might not detect a signal state change on the other side. Any operation that has both an initiating signal and corresponding acknowledge signal will use this basic handshake procedure.

The procedure involves a four-way handshake.

1. Assert signal
2. Signal acknowledge
3. De-assert signal
4. De-assert acknowledge

The requesting device asserts the signal to request an action (set bit 0 -> 1). When the target device detects the signal and the requested operation has completed, it asserts the corresponding acknowledge (set bit 0 -> 1). When the requesting device detects the acknowledge, it de-asserts the original signal (1 -> 0). Finally, when the target device detects the original signal de-asserted, it de-asserts its acknowledge (bit 0 -> 1). To function correctly both sides must see the complete assert/de-assert cycle (0 -> 1 and 1 -> 0). The requesting device should not initiate a subsequent request until the cycle completes.

Acquisition Sequence

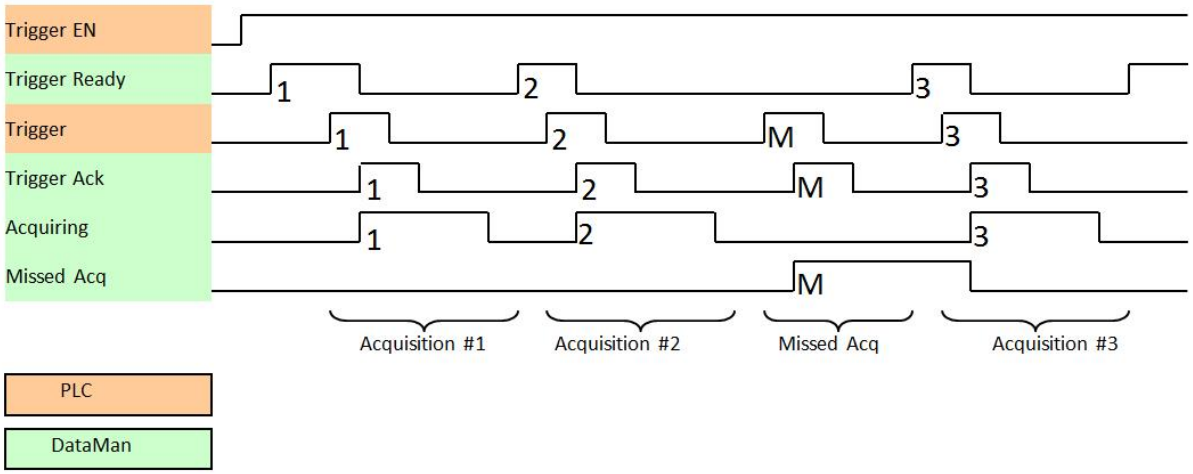
DataMan can be triggered to acquire images by several methods. It can be done by setting the *Trigger* bit or issuing a trigger String Command. It can also be done via DMCC command (Telnet) or hardwired trigger signal. The *Trigger* bit method will be discussed here.

On startup, *TriggerEnable* will be False. It must be set to True to enable triggering via the *Trigger* bit. When the device is ready to accept triggers, the reader will set the *TriggerReady* bit to True.

While the *TriggerReady* bit is True, each time the reader detects the *Trigger* bit change from 0 -> 1, it will initiate a read. The *Trigger* bit should be held in the new state until that same state value is seen in the *TriggerAck* bit (this is a necessary handshake to guarantee that the trigger is seen by the reader).

During an acquisition, the *TriggerReady* bit will be cleared and the *Acquiring* and *Decoding* bits will be set to True. When the acquisition is completed, the *Acquiring* and *Decoding* bits will be cleared. When the device is ready to begin another image acquisition, the *TriggerReady* bit will again be set to True.

The reader allows acquisitions to overlap with the decoding of previous acquisitions. The Trigger Ready bit is set high after the acquisition is complete, while decoding may still be in progress. The Decoding bit is deprecated and only mirrors the behavior of the Acquiring bit. If trigger queuing is active or other trigger sources can interfere, the Trigger Ready bit may not be reliable. As result, conflicting trigger overruns may occur, which are reported in Bit 3 of the Result Code word of the Output Data Block.



To force a reset of the trigger mechanism set the *TriggerEnable* to False until *TriggerReady* is also set to False. Then, *TriggerEnable* can be set to True to re-enable acquisition.

As a special case, an acquisition can be cancelled by clearing the *Trigger* signal before the read operation has completed. This allows for the cancellation of reads in Presentation and Manual mode if no code is in the field of view. To ensure that a read is not unintentionally cancelled, it is advised that the PLC hold the *Trigger* signal True until both *TriggerAck* and *ResultsAvailable* are True (or *DecodeComplete* toggles state).

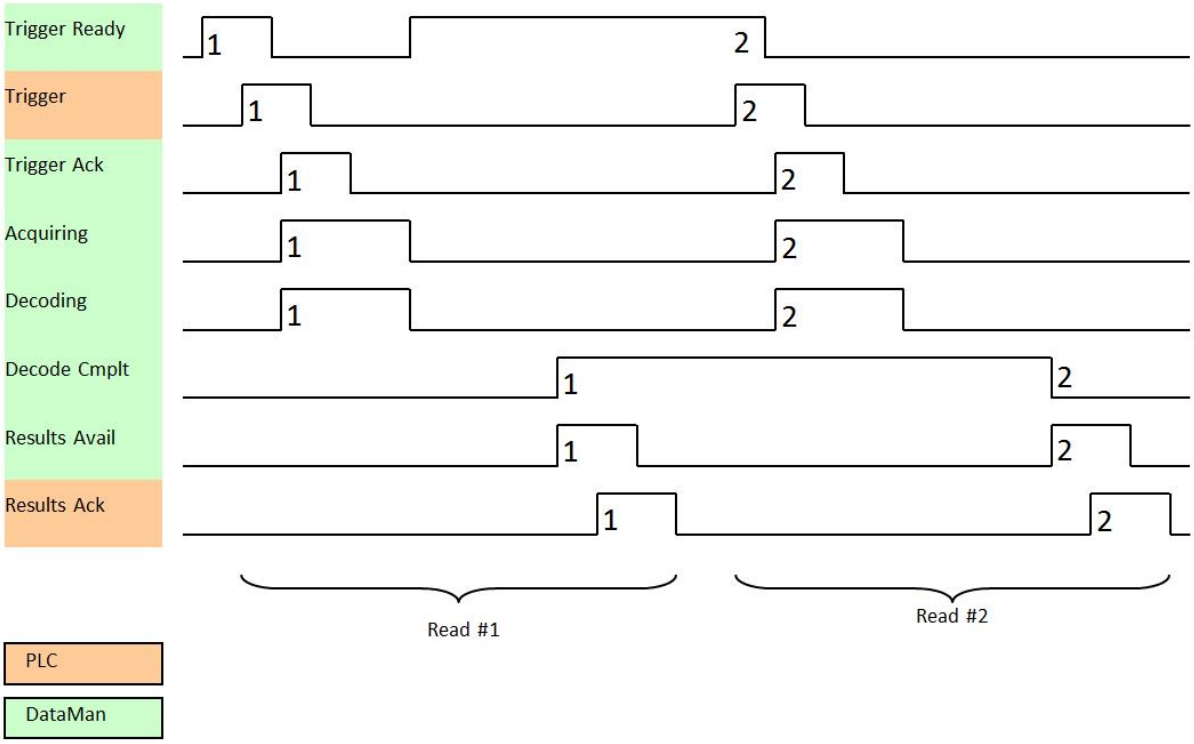
Decode / Result Sequence

After an image is acquired, it is decoded. When the decoding is complete and a result is ready to be delivered, the reader toggles the *DecodeComplete* bit of the *DecodeStatusRegister*.

Decode results are reported asynchronously to the Output Data block. If BufferResultsEnable is set to False, then the decode results are immediately placed into DecodeResults, and the Results Available bit is set to True.

> **Note**: The only way to ensure that you are not losing results is to use BufferResults. Make sure that your application can read each DecodeResult in time if you do not use BufferResultsEnable.

If *ResultsBufferEnable* is set to True, the new results are queued in a buffer and *DecodeComplete* is toggled. The earlier read results remain in the Output Data block until they are acknowledged by the PLC. After the acknowledgment handshake, if there are more results in the queue, the next set of results will be placed in the Output Data block and *ResultsAvailable* is set to True.



Results Buffering

The Results Buffering feature allows a queue for decode results. When enabled, the feature queues a finite number of result data sets until the PLC (client) has time to read them. This smooths out data flow if the PLC slows down for short periods or if there are surges in read activity.

If the feature is enabled, the reader overlaps acquisition and decode operations. This can lead to faster over all trigger rates. When reads occur faster than results can be sent out (the queue is full), the most recent results are discarded until space becomes available in the results queue.

If the feature is disabled, the more recent results overwrite the earlier results. The most recent results are kept. Earlier results are discarded if the PLC does not read them fast enough.

If the queue overflows while the feature is disabled:

- The difference between the TriggerID and ResultID values becomes greater than 1. This difference represents the number of reads that occurred but could not be queued because the queue was full.
- The number of lost reads equals TriggerID - ResultID - 1.
- After the next read, the ResultID value returns to the typical operating value of TriggerID - 1.

Result Fragmentation

The Result Fragmentation feature splits decode result strings into fragments if the strings are too large for the Results Data field. The fragments are then pushed to the buffer like normal results. Then, the PLC reads the fragments one by one.

Result Fragmentation depends on Result Buffering. This means that results are only fragmented if Result Buffering is enabled.

Note: For Modbus TCP, you have to enable Result Fragmentation manually.

Enabling Result Fragmentation

To enable or disable Result Fragmentation, you have the following options:

- Use the FFP.RESULT-FRAG DMCC.
- Use SoftEvent 6 to enable the Results Buffering and Result Fragmentation features from the PLC.

Operation of Result Fragmentation

The PLC does not automatically reassemble the result string. Once the PLC reads a result fragment, you have to copy the fragment to a new user-defined variable. After this, you have to set and clear the ResultAck bit, which updates the Decode Results with the next fragment. This process continues until all fragments have been sent to the PLC.

In contrast, if Results Buffering is disabled, large decode result strings are cut off to fit into the Results Data field, which means that the end of the string gets lost.

Identifying Decode Result Chunks

Decode result fragments look like normal results. To identify result fragments, look at the Result ID:

- The Result ID of the result fragments are the same.
- The Result ID of the last result fragment is different from the Result ID of the next result.

Independent of whether results fragmentation is enabled, the Result Length always represents the number of characters to be read from the Results Data plus the remaining characters that did not fit into the Results Data buffer. With results fragmentation enabled, the remaining characters will be found in follow-up fragments, the remaining characters are discarded.

You can identify the first and last fragments of a result as follows:

- The first fragment has a different Result ID from the previous Result ID.
- The Result Length of the last fragment is lower or equal to the Result Data buffer size.
- All fragments of a result share the same Result ID.

Maximum Result Data Length

Without results fragmentation, result data is truncated to fit into the results data buffer size. Depending on the protocol, the result buffer size can be fixed or configurable.

The Result Fragmentation feature allows processing of up to 50 times longer results, depending on the results buffering queue size, with an absolute limit of 65535 characters.

The maximum number of result characters for ModbusTCP is up to 4000 bytes (configurable).

SoftEvents

SoftEvents act as virtual inputs. When the value of a *SoftEvent* bit changes from 0 to 1, the action associated with the event is executed. When the action completes, the corresponding *SoftEventAck* bit changes from 0 to 1 to signal completion.

The *SoftEvent* and *SoftEventAck* form a logical handshake. After *SoftEventAck* changes to 1, make sure that the original *SoftEvent* is set back to 0. When that occurs, *SoftEventAck* is automatically set back to 0.

> **Note**: Do not execute SoftEvents that change the reader configuration at the same time that reads are being triggered. Changing configuration during acquisition/decode can lead to unpredictable results.

The “ExecuteDMCC” and “SetMatchString” SoftEvent actions require user supplied data. Make sure that this data is written to the *UserData* and *UserDataLength* area of the Input Data block prior to invoking the SoftEvent. Only one SoftEvent can be invoked at a time, because both of these SoftEvents depend on the *UserData*.

String Commands

The String Command feature allows you to execute string-based DMCCs over the SLMP or the Modbus TCP protocol connection. The DMCC is sent to the reader through the String Command block. The DMCC result is returned through the String Command Result block. Initiating a command and notification of completion is accomplished by signaling bits in the Control and Status blocks.

To execute a DMCC, the command string is placed in the data field of the String Command block. The command string consists of standard ASCII text. The same command format is used for a serial (RS-232) or Telnet connection. The string does not need to be terminated with a null character. Instead, the length of the string (that is, the number of ASCII characters) is placed in the length field of the String Command block.

After executing the DMCC, the result string is returned in the String Command Result block. Similar to the original command, the result string consists of ASCII characters in the same format that is returned through a serial or Telnet connection. Also, there is no terminating null character. Instead the length of the result is returned in the Command String Result length field. The Command String Result block also contains a numeric result code. This allows you to determine the success or failure of the command without having to parse the text string. The values of the result code are defined in the DMCC documentation.

DMCC Result Feedback from PLC

The DMCC Result Feedback feature allows the reader to send DMCC responses back to the PLC.

Note: The Modbus/TCP protocol already offers the String Commands feature to initiate DMCC commands with result feedback. The DMCC Result Feedback feature is also available, but the String Commands feature is preferred because it is simpler to use.

By default, this feature is off. To enable the feature, use the `FFP.DMCC-RESULTS` DMCC command. Changing this configuration does not require you to reboot or reconnect reader.

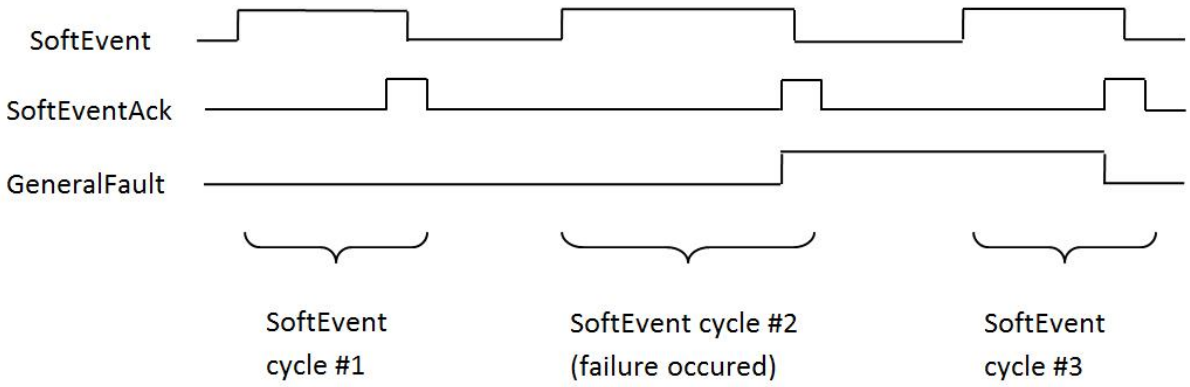
When you enable the feature, the reader sends DMCC responses to the PLC using the same mechanism it uses to send read results. To distinguish DMCC results from read results, bit 6 is of the ResultCode field is set to 1. The following Output Data Block fields contain DMCC information instead of read results:

Word	Name	Description
2	ResultID	Contains a DMCC result code: • 0: no error • 100: unidentified error • 101: invalid command • 102: invalid parameter • 103: incorrect checksum • 104: parameter rejected/altered due to reader state • 105: reader unavailable (offline)
3	ResultCode	Bit 6 is set to 1 to indicate a DMCC response.
4	ResultLength	Actual number of bytes of DMCC result data contained in the *ResultData* field (16-bit Integer).
5..N	ResultData	DMCC result data (array of bytes).

You can use DMCC Result Feedback in combination with Results Fragmentation.

General Fault Indicator

When a communication-related fault occurs, the “GeneralFault” bit changes from 0 to 1. The only fault conditions supported are SoftEvent operations. If a SoftEvent operation fails, the fault bit will be set. The fault bit will remain set until the next successful SoftEvent operation, or until *TriggerEnable* is set to 0 and then back to 1.



Modbus TCP Examples

Included with the DataMan Setup Tool installer are two example PLC programs created with CoDeSys v2.3 software. These samples are designed and tested on a Wago 750-841 PLC. These programs demonstrate the DataMan ID readers' capabilities and proper operation. You can do the same operations by using more advanced features and efficient programming practices with Wago PLCs. The examples try to show different approaches in the techniques used for the communication to the DataMan reader and they are better for demonstration purposes.

Note: All examples are designed to work only if the "Control" datablock is mapped to the Coil space and the "Status" datablock is mapped to the Discrete Input space.

ApplicationLayer Example

This sample realizes a generic data transfer between the DataMan reader and the PLC. Memory areas of the "Control", "Status" and "Output Area" are cloned in the PLC and synchronized cyclically, or as needed. Each data area is synchronized with its own instance of "ETHERNETMODBUSMASTER_TCP". This causes three TCP connections to be open simultaneously. Make sure that the Modbus TCP related setting "Maximum Connections" on the DataMan reader is set to at least 3 for this example to work.

Function

The example application demonstrates the following operations:

1. Transfer the 32-bit "Control" register data from the PLC to the reader.
2. Transfer the 32-bit "Status" register data from the reader to the PLC.
3. Transfer "Output Data" from the reader to the PLC.

All actions are started when there is a connection to the reader.

Transferring "Control" Register Data

All data gets transferred when there is a change in the local PLC data. The local PLC data can be manipulated in the visualization.

Note: No synchronization is implemented from the reader to the PLC, so the local PLC data can be incorrect after building up the connection or if another Modbus TCP client manipulates the Contol register simultaneously. There is a timeout setting that can lead to a disconnect if you do not manipulate the "Control" register during this timeframe.

Transferring Status Register Data

All data gets transferred cyclically. The poll interval can be specified in the visualization.

Transferring Output Data

All data gets transferred cyclically. The poll interval can be specified in the visualization.

DataManControl Example

This sample shows in a sequential manner the steps to do to achieve one of the functions named in the following subsection. To outline this chronological sequence, “Sequential Function Chart” is used as programming language.

Function

The example application demonstrates the following operations:

1. Triggering a read
2. Getting read results
3. Executing string commands (DMCC)
4. Executing SoftEvent operations
    a. Train code
    b. Train match string
    c. Train focus
    d. Train brightness
    e. Untrain
    f. Execute DMCC
    g. Set match string

The “Main” program contains variables to invoke each of these operations. The operation is invoked by toggling the control bool directly or from the visualization (red=0, green=1) from 0 to 1. This will invoke the associated subroutine to perform the operation. When the operation 4 is complete, the subroutine will set the control bit back to 0.

Triggering a Read

The example provides a “Continuous Trigger”. As the name implies, enabling the “xTrigger” bit will invoke a continuous series of read operations. Once enabled, the “xTrigger” control bit will remain set until you disable it.

Primarily, the trigger subroutine manages the trigger handshake operation between the PLC and the reader. The control *Result Ack* and *Trigger* bits are reset, the *Trigger Enable* bit is set, the PLC waits for the corresponding *TriggerReady* status bit from the reader, and the control Trigger bit is set. Refer to a description of handshaking in section Operation.

Getting Read Results

For this example the operation of triggering a read and getting read results was intentionally separated. This is to support the situation where the PLC is not the source of the read trigger. For example, the reader may be configured to use a hardware trigger. In such a case, only the get results subroutine would be needed.

Like the triggering subroutine, the get results subroutine manages the results handshake operation between the PLC and the reader. The routine waits for the *ResultsAvailable* status bit to become active, it copies the result data to internal storage, and then executes the *ResultsAck* handshake. Refer to a description of handshaking in section Operation.

The read result consists of a *ResultCode*, **ResultLength**, and *ResultData*. Refer to section Output Data Block Field Descriptions for details of the *ResultCode* values. The *ResultLength* field indicates how many bytes of actual result data exist in the *ResultData* field. The subroutine converts this byte length to word length before copying the results to internal storage.

Execute String Commands (DMCC)

The string command feature provides a simple way to invoke DMCCs from the PLC. The command format and command result format is exactly identical to that used for serial or Telnet DMCC operation.

This subroutine copies an example DMCC (||>GET DEVICE.TYPE) to the String Command block and then manages the string command handshake operation between the PLC and the reader to invoke the command and retrieve the command result. Any valid DMCC command may be invoked with this mechanism. Refer to the DataMan ***Command Reference*** document available through the Windows **Start** menu or the Setup Tool **Help** menu.

Execute SoftEvents

SoftEvents are used to invoke a predefined action. Each SoftEvent is essentially a virtual input signal. Each of the SoftEvent subroutines manages the handshake operation between the PLC and the reader to invoke the predefined action. The associated action is invoked when the *SoftEvent* bit toggles from 0 to 1. The subroutine then watches for the associated *SoftEventAck* bit from the reader which signals that the action is complete. For a description of handshaking, see section Operation.

Note: The “Execute DMCC” and “Set Match String” SoftEvents make use of the Input Data block. The subroutine for these two events copies the relevant data into the User Data fields of the Input Data block and then invokes the User Data subroutine to transfer the data to the reader. Only after the user data is transferred is the actual SoftEvent action invoked. It is required that the user data be transferred before invoking either of these events.

Note: The “Train Match String” SoftEvent only prepares the training mechanism. The actual training occurs on the next read operation. Therefore, a trigger must be issued following “Train Match String”.

EtherCAT

EtherCAT is an application-level protocol used in industrial automation applications. This real-time Ethernet protocol uses standard Ethernet hardware for wiring and special EtherCAT hardware on readers for efficient communication in a fieldbus segment. EtherCAT aims to maximize the data rate on the physical network by sending frames always from the EtherCAT master controller. One master device must be present on the fieldbus segment. The master device is often implemented as a Programmable Logic Controller (PLC).

DataMan readers support EtherCAT on readers that integrate an EtherCAT subdevice controller. Refer to the *Quick Start Guide* of your reader to see whether yours contains the EtherCAT hardware and for further information, for example, which ports on the reader are used for EtherCAT communication.

To use your Ethernet card for EtherCAT virtualization as part of development or demonstrations, plug the DataMan reader into a properly set environment. EtherCAT communication can create problems in the Ethernet settings of your host computer.

By default, the reader has the EtherCAT protocol enabled. You can enable or disable the protocol through DMCC, scanning a configuration code, or in the DataMan Setup Tool. You need to connect the DataMan Setup Tool through the secondary Ethernet socket if you want to use DMCC or the DataMan Setup Tool.

File Over EtherCAT (FoE) Function

You can transmit files from the PLC to the reader over EtherCAT. The following file extensions are supported:

File Extension	File Type	Comments
.bin.gz	Firmware update	The reader does not reboot automatically to apply the update. You have to reboot the reader manually, or using *EXECUTE DMCC* SoftEvent. The reader is not operational during the update.
.cdc	Device backup	You must manually reboot the EtherCAT reader to apply the backup files. This is different from non-EtherCAT DataMan readers, which restart automatically. If you do not reboot the reader, the old state persists. You can reboot the reader with the *Execute DMCC* SoftEvent.
.cfg	Device configuration	Does not require a reboot.

Note:

- If you have more than one readers connected in a ring topology, the configuration can break when you reboot more than one readers at the same time.
- Certain PLCs can remove the file extension when you upload these files through FoE. Always make sure that the file extensions are correct.
- You can find the *.cfg* and *.cdc* files by unzipping the .DMB file that the DataMan Setup Tool produces when exporting a reader configuration.

DMCC

You can use the following commands to enable or disable EtherCAT:

Enable:

```
||>SET ETHERCAT.ENABLED ON
||>CONFIG.SAVE
||>REBOOT
```

Disable:

```
||>SET ETHERCAT.ENABLED OFF
||>CONFIG.SAVE
||>REBOOT
```

You can issue these commands through RS-232 or Telnet connection. Use a third-party Telnet client, such as PuTTY, to communicate with your DataMan reader.

Reader Configuration Code

Scanning the following reader configuration codes enables or disables EtherCAT for your reader.

Note: You must reboot the reader for the change to take effect.

Enable

Disable

Setup Tool

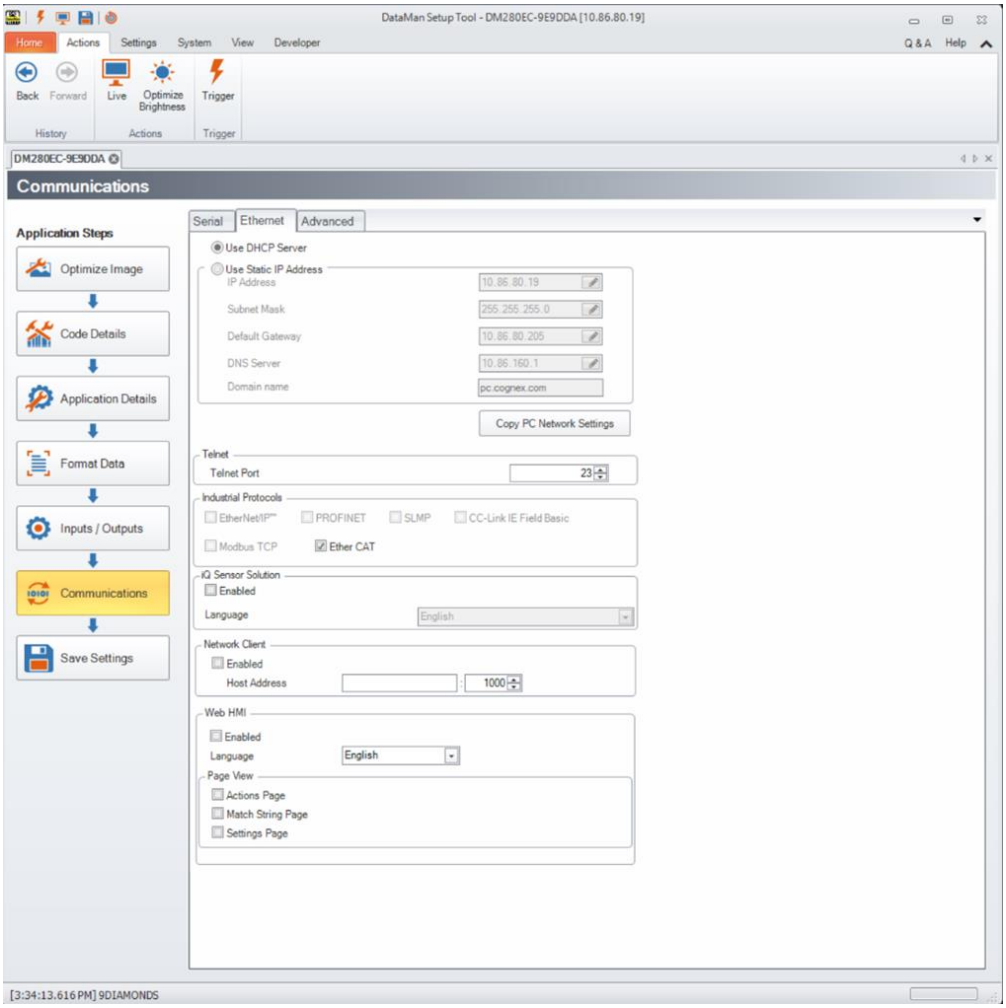
1. Open Setup Tool to discover the reader.

   Note: The reader must be in the subnet.

2. If the reader appears as misconfigured, select the reader and click **Repair & Support** to change the misconfigured IP address to the desired static IP.

   Note: The reader must have a properly configured IP address for connecting to the DataMan Setup Tool.

3. Click **Connect** to connect to the reader.
4. Navigate to the **Communications** application step.
5. Enable EtherCAT under the **Ethernet** tab.
6. Reboot the reader.



Preparing to Use EtherCAT

Preparing to use EtherCAT involves the following main steps:

- To use a DM280EC reader, you need an EtherCAT Master device or EtherCAT PLC with a run-time project developed for DM280EC. A DataMan 280EC is an EtherCAT slave. In case of a Beckhoff PLC, TWINCAT3 is the development tool for EtherCAT PLC projects. CodeSYS also supports EtherCAT.
- Integrate a DM280EC into your PLC project is to install the ESI file.

> **Note**: The ESI file delivered with firmware version 6.3.8 or later is not compatible with earlier versions.

EtherCAT Object Model

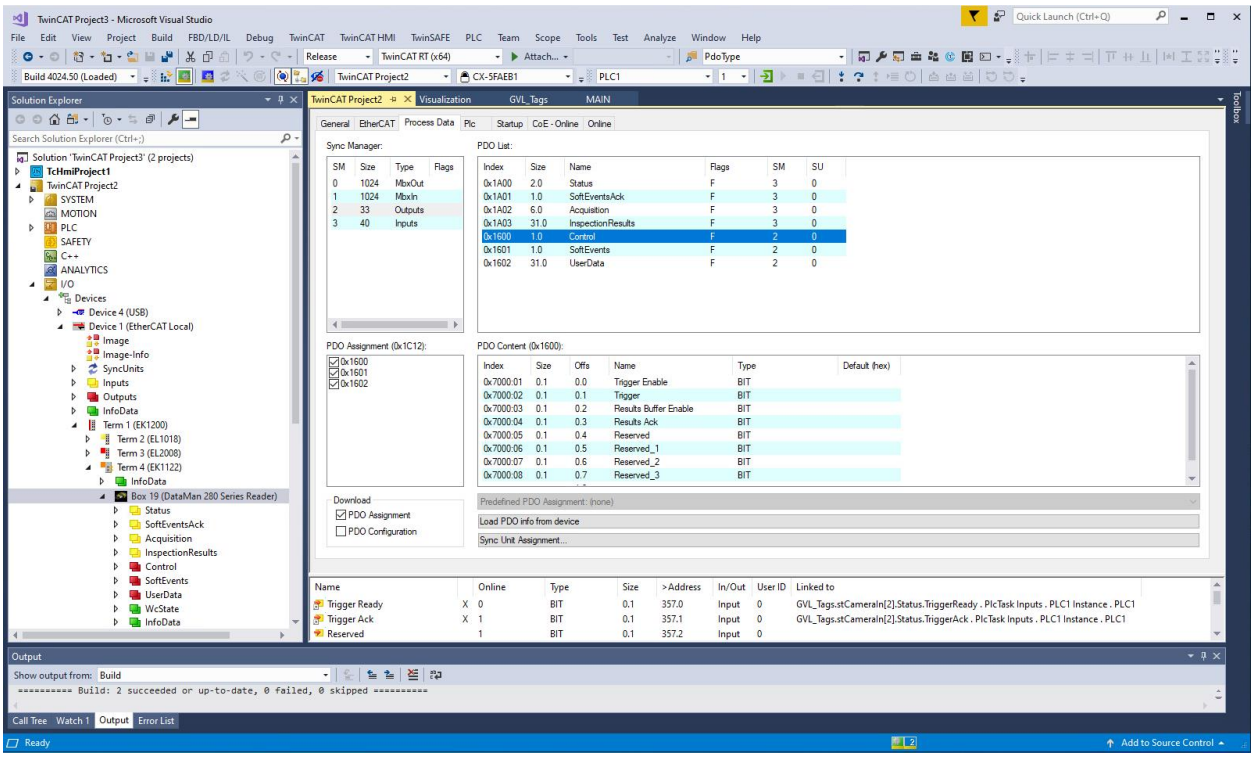
> **Note**: This section applies to readers running firmware version 6.3.10 and higher. For a complete example, see the *Cognex_DM280EC_ESI_v4.xml* ESI file delivered with DataMan Setup Tool.

The EtherCAT implementation consists of the following Process Data Objects (PDOs):

- Control
- Status

- Acquisition
- Inspection Results (configurable blocks)
- SoftEvent
- SoftEvent Ack
- User Data (configurable blocks)

The following image shows an example of PDOs:



The Cognex object model has modular PDOs to offer flexibility and maximize performance or usability on demand. Using the maximum PDO size extends usability at the expense of the performance. For better performance, reduce the size of the PDOs. You only need to select the PDO in the application.

The PDO can have a maximum size of 1048 bytes out of 1486 bytes in a single frame. You need at least 152 bytes.

Note: For more information about how frames are used for Slaves, see EtherCAT Introduction.

Control Object

Controls the reader. This object consists of data sent from the PLC to the DataMan reader.

Byte size: 1 byte

Index	Name	Description
7000:01	Trigger Enable	Enables triggering through EtherCAT. Clearing this bit disables triggering.
7000:02	Trigger	Triggers an acquisition when the following conditions are met: • *Trigger Enable* is set • No acquisition is currently in progress • The reader is ready to trigger

7000:03	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 reader responds to the acknowledge 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, received read results overwrite the content of the Result Data module. The reader can buffer up to 50 sets of read results.
7000:04	Results Ack	Used to acknowledge that the PLC has successfully read the latest result data. When set True the *Result Available* bit will be cleared. If result buffering is enabled, the next set of result data will be pulled from the queue and *Result Available* will again be set True.
7000:05	Reserved	Reserved for future use.
7000:06		
7000:07		
7000:08		

Soft Events

Used to initiate a SoftEvent.

Byte size: 1 byte

Index	Name	Description
7001:01	Train Code	Transition from 0 to 1 invokes the Train Code operation.
7001:02	Train Match String	Transition from 0 to 1 invokes the train match string operation.
7001:03	Train Focus	Transition from 0 to 1 invokes the train focus operation.
7001:04	Train Brightness	Transition from 0 to 1 invokes the train brightness operation.
7001:05	Untrain	Transition from 0 to 1 invokes the untrain operation.
7001:06	Set UserData Fragment	Transition from 0 to 1 indicates that the current *User Data* is a fragment. Fragments are accumulated until an Execute DMCC or a Set Match String Soft Event is invoked. The corresponding SoftEvent Ack bit signals when the PLC reads the fragment. Transition from 1 to 0 resets the Ack bit.
7001:07	Execute DMCC	Transition from 0 to 1 invokes the DMCC operation. You must first place a valid DMCC command string in *User Data* before invoking this event. Transition from 1 to 0 resets the corresponding Ack bit.
7001:08	Set Match String	Transition from 0 to 1 invokes the set match string operation. You must first place match string data in *User Data* before invoking this event.

User Data Object

Data sent from a PLC to a reader to support acquisition, decode, and other special operations. Currently this object is only used to support the *Execute DMCC* and *Set Match String* SoftEvents.

PDO	Index
UserData32	7002
UserData64	7003
UserData128	7004
UserData256	7005

Byte	Name	Description
0 32 64 96 128 224 256 480	User Data	Data sent from the PLC to the reader to support acquisition, decoding, and other special operations (array of bytes). **Note**: Requires a \0 termination character. **Note**: For the *Set Match String* SoftEvent, the pattern is applied to all symbologies.

Status Object

Indicates the current reader status. This object consists of data sent from the DataMan reader to the PLC.

Byte size: 2 byte

Index	Name	Description
6000:01	Trigger Ready	Indicates when the reader is ready to accept a new trigger. True when *Trigger Enable* is set and the reader is ready to accept a new trigger.
6000:02	Trigger Ack	Indicates that the reader has received a new Trigger. Remains True as long as the *Trigger* bit remains True, that is, it is interlocked with the Trigger bit.
6000:03	Acquiring	Indicates that the reader is in the process of acquiring an image.
6000:04	Acquisition Error	Indicates that the reader was unable to successfully trigger an acquisition. Cleared when the next successful acquisition occurs.
6000:05	Reserved	Reserved for future use.
6000:06		
6000:07		
6000:08		
6000:0A	Decoding	Set while decoding an image.
6000:0B	Results Buffer Overrun	Indicates that the reader has discarded a set of read results because the results queue is full. Cleared when the next set of results are successfully queued.
6000:0C	Results Available	Indicates that a new set of read results are available, that is, the contents of the Result Data module are valid. Cleared when the results are acknowledged.
6000:0D	General Fault	Indicates that a fault has occurred, for example, a SoftEvent *Set Match String* or *Execute DMCC* error has occurred.
6000:0E	Reserved	Reserved for future use.
6000:0F		
6000:0G		

Soft Events Ack

Used to acknowledge completion of a SoftEvent.

Byte size: 1 byte

Index	Name	Description
6001:01	Train Code Ack	Indicates that the *Train Code* operation is complete.
6001:02	Train Match String Ack	Indicates that the *Train Match String* operation is complete.
6001:03	Train Focus Ack	Indicates that the *Train Focus* operation is complete.
6001:04	Train Brightness Ack	Indicates that the *Train Brightness* operation is complete.
6001:05	Untrain Ack	Indicates that the *Untrain* operation is complete.
6001:06	Set UserData Fragment Ack	Indicates that the data fragment in *User Data* was read and the PLC is ready to process the next fragment.
6001:07	Execute DMCC Ack	Indicates that the *Execute DMCC* operation is complete.
6001:08	Set Match String Ack	Indicates that the *Set Match String* operation is complete.

Acquisition Object

Contains further information about the Result. This object consists of data sent from the reader to the PLC.

Byte size: 6 byte

Index	Name	Description
6002:01	TriggerID	Trigger identifier. ID of the next trigger to be issued.
6002:02	ResultID	The value of the Trigger ID when the trigger that generated these results was issued. Used to match up triggers with corresponding result data (16-bit integer).
6002:03	Read	The bit is set on a successful decode.
6002:04	Validated	The bit is set on a successful code validation.
6002:05	Verified	The bit is set on a successful code verification.
6002:06	Acquisition Trigger Overrun	The bit is set if the error trigger overrun is risen.
6002:07	Acquisition Buffer Overflow	The bit is set if the error buffer overflow is risen.
6002:08	Reserved	Reserved for future use.
6002:09	DMCC Response	The DMCC responses over EtherCAT is enabled by default. This bit is set when the Inspection result bit carries the output of a DMCC. If zero, then inspection result carries a decode result.
6002:0A	Reserved	Reserved for future use.
6002:0B		
6002:0C		
6002:0D		
6002:0E		
6002:0F		
6002:10		
6002:11		
6002:12		

Inspection Results Object

Read result data sent from a reader to a PLC.

Bytes sizes:

- 0
- 32
- 64
- 96
- 128
- 224
- 256
- 480

PDO	Index
InspectionResult32	6003
InspectionResult64	6004
InspectionResult128	6005
InspectionResult256	6006

Byte	Name	Description
0...1	Result Length	Actual number of bytes of Result Data contained in the *Result Data* field (16-bit integer).
2...	Result Data	Decoded read result data (array of bytes).

Note: If the result is bigger than the frame configured, the result is sliced into subsequent data frames. For more information, see the *Long Codes Behavior* section in the *EtherCAT Operation* on page 141.

EtherCAT Operation

Note: This section applies to readers running firmware version 6.3.10 and higher. For a complete example, see the *Cognex_DM280EC_ESI_v4.xml* ESI file delivered with DataMan Setup Tool.

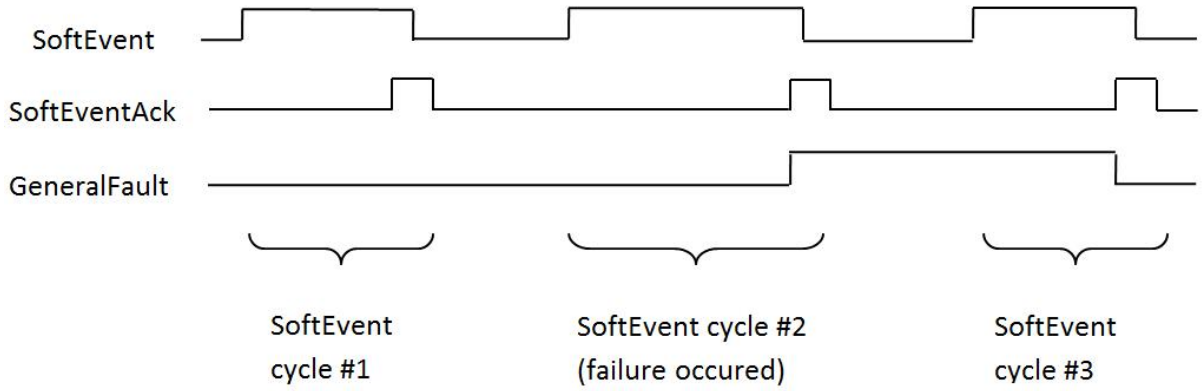
SoftEvents

SoftEvents act as “virtual” inputs. When the value of a *SoftEvent* bit changes from 0 to 1, the action associated with the event is executed. When the action completes, the corresponding *SoftEventAck* bit will change from 0 to 1 to signal completion. The acknowledge bit will change back to 0 when the corresponding SoftEvent bit is set back to 0.

The *ExecuteDMCC* and *SetMatchString* SoftEvent actions require user-supplied data. This data must be written to the UserData Module prior to invoking the SoftEvent. Since both of these SoftEvents depend on the *UserData*, only one can be invoked at a time.

General Fault Indicator

When a communication-related fault occurs, the *GeneralFault* bit will change from 0 to 1. Currently the only fault conditions supported are SoftEvent operations. If a SoftEvent operation fails, the fault bit will be set. The fault bit will remain set until the next successful SoftEvent operation, or, until *TriggerEnable* is set to 0 and then back to 1.



Acquisition Sequence

You can trigger the reader to acquire images with the following methods:

- By explicitly manipulating the Trigger bit of the Acquisition Control Module
- Through external hardwired input
- Through DMCC

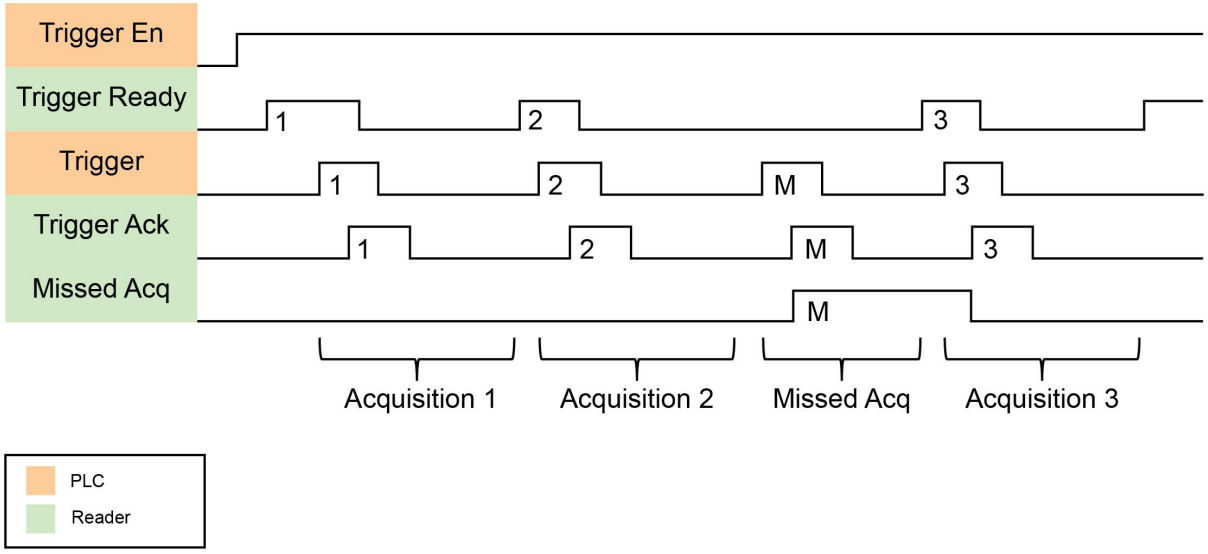
This section describes manipulating the Control Object bits.

On startup, the *TriggerEnable* bit is False. It must be set to True to enable triggering. When the reader is ready to accept triggers, the reader sets the *TriggerReady* bit to True.

While the *TriggerReady* bit is True, each time the reader detects the *Trigger* bit change from 0 to 1, it initiates a read. Make sure that the PLC holds the bit in the new state until that same state value is seen in the *TriggerAck* bit. This is a necessary handshake to guarantee that the reader detects the change.

During an acquisition, the *TriggerReady* bit is cleared. When the reader is ready to begin another image acquisition, the *TriggerReady* bit is set to True again.

If results buffering is enabled, the reader allows overlapped acquisition and decoding operations. The *TriggerReady* bit is set high after the acquisition is complete but while the decoding is still in process. You can use this to achieve faster overall trigger rates. If result buffering is not enabled, the *TriggerReady* bit remains low until both the acquisition and decode operations are complete.



To force a reset of the trigger mechanism, set the *TriggerEnable* bit to False until the *TriggerReady* bit is also set to False. Then, the *TriggerEnable* bit can be set to True to reenable acquisition.

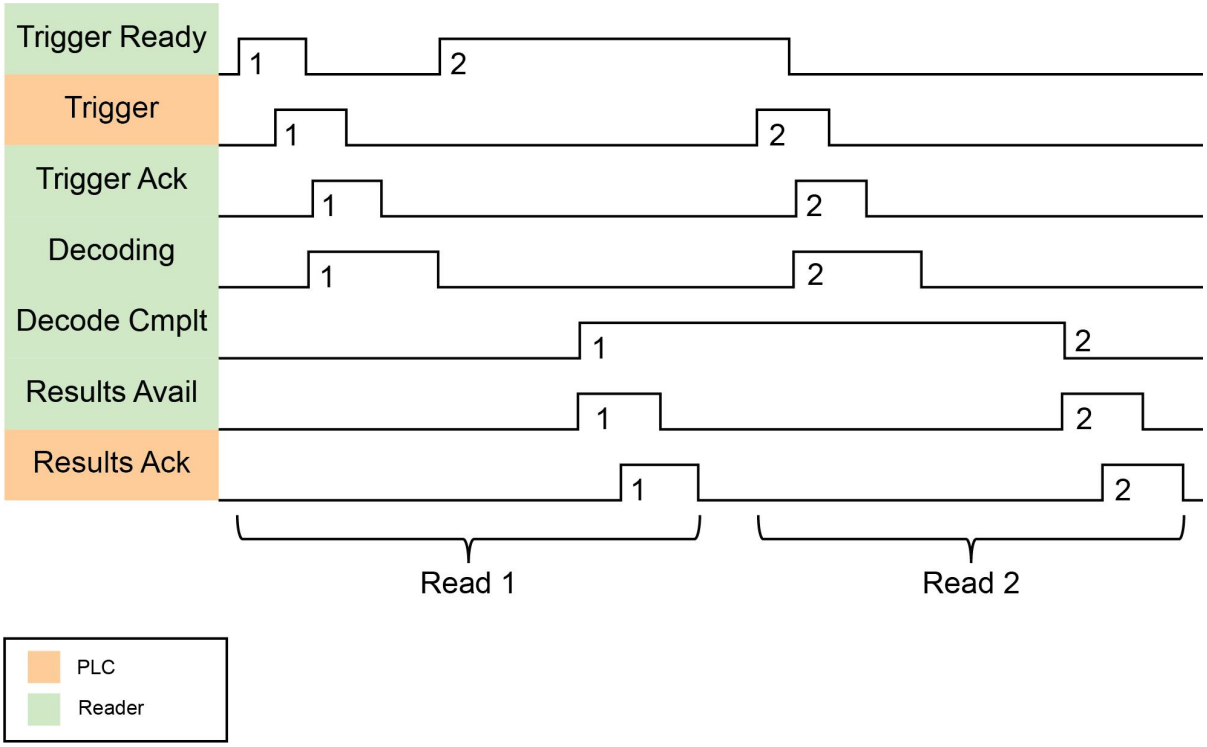
As a special case, you can cancel an acquisition by clearing the *Trigger* signal before the read operation is complete. This allows for canceling reads in Continuous and Manual mode if no code is in the field of view. To ensure that a read is not unintentionally canceled, make sure that the PLC holds the Trigger signal True until both the *TriggerAck* and *ResultsAvailable* bits are True (or the *DecodeComplete* bit toggles state).

Decode / Result Sequence

After an image is acquired, it is decoded. When the decoding is complete and a result is ready to be delivered, the reader toggles the *DecodeComplete* bit of the Status Object.

The *Results Buffer Enable* bit determines how the reader handles decode results. If BufferResultsEnable is set to False, then the decode results are immediately placed into DecodeResults, and the Results Available bit is set to True.

If the Results Buffer Enable bit is set to True, the new results are queued. The earlier decode results remain in the Results Object until the client acknowledges them by setting the *Results Ack* bit to True. After the Results Available bit is cleared, make sure that the client sets 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 DataMan client (PLC).



Results Buffering

The Results Buffering feature allows a queue for decode results. When enabled, the feature queues a finite number of result data sets until the PLC (client) has time to read them. This smooths out data flow if the PLC slows down for short periods or if there are surges in read activity.

If the feature is enabled, the reader overlaps acquisition and decode operations. This can lead to faster over all trigger rates. When reads occur faster than results can be sent out (the queue is full), the most recent results are discarded until space becomes available in the results queue.

If the feature is disabled, the more recent results overwrite the earlier results. The most recent results are kept. Earlier results are discarded if the PLC does not read them fast enough.

If the queue overflows while the feature is disabled:

- The difference between the TriggerID and ResultID values becomes greater than 1. This difference represents the number of reads that occurred but could not be queued because the queue was full.
- The number of lost reads equals TriggerID - ResultID - 1.
- After the next read, the ResultID value returns to the typical operating value of TriggerID - 1.

Long Codes Behavior with Results Buffering for EtherCAT

During non-buffered operation, only the selected result data array that is available in the Inspection Result object can store read results. If you need to read codes longer than the selected result data array, enable Results Buffering and perform the steps below. Before doing so, make sure the *ResultsAvailable* bit in the Status object is 1 and *Result length* contains a longer length than bytes available in the array.

You can choose from the following frame sizes in bytes:

- 32
- 64
- 96
- 128
- 224
- 256
- 480

The process is the following:

1. Read out the selected result data array.
2. Set *ResultsAck* bit in the Control object to 1 to acknowledge reading.
    - *ResultsAvailable* bit in the Status object goes to 0.
3. Lower *ResultsAck* bit in the control object to 0 again.
    - *ResultValid* bit in the status object goes back to 1.
    - *ResultID* in the Inspection Result Header stays the same because no new result is available.
    - *ResultID* in the Inspection Result Header stays the same because no new result is available.
    - *Result length* now contains the length of the remaining data of the full code: last length of the result data array.
4. Read out the remaining data.
5. If not finished, continue with step 2. Don't forget to also acknowledge the last readout, so that the next Result can be delivered.

Result Fragmentation

The Result Fragmentation feature splits decode result strings into fragments if the strings are too large for the Results Data field. The fragments are then pushed to the buffer like normal results. Then, the PLC reads the fragments one by one.

Result Fragmentation depends on Result Buffering. This means that results are only fragmented if Result Buffering is enabled.

Note: For EtherCAT, the Result Fragmentation feature is automatically enabled when you enable Results Buffering.

Operation of Result Fragmentation

The PLC does not automatically reassemble the result string. Once the PLC reads a result fragment, you have to copy the fragment to a new user-defined variable. After this, you have to set and clear the ResultAck bit, which updates the Decode

Results with the next fragment. This process continues until all fragments have been sent to the PLC.

In contrast, if Results Buffering is disabled, large decode result strings are cut off to fit into the Results Data field, which means that the end of the string gets lost.

Identifying Decode Result Chunks

Decode result fragments look like normal results. To identify result fragments, look at the Result ID:

- The Result ID of the result fragments are the same.
- The Result ID of the last result fragment is different from the Result ID of the next result.

Independent of whether results fragmentation is enabled, the Result Length always represents the number of characters to be read from the Results Data plus the remaining characters that did not fit into the Results Data buffer. With results fragmentation enabled, the remaining characters will be found in follow-up fragments, the remaining characters are discarded.

You can identify the first and last fragments of a result as follows:

- The first fragment has a different Result ID from the previous Result ID.
- The Result Length of the last fragment is lower or equal to the Result Data buffer size.
- All fragments of a result share the same Result ID.

Maximum Result Data Length

Without results fragmentation, result data is truncated to fit into the results data buffer size. Depending on the protocol, the result buffer size can be fixed or configurable.

The Result Fragmentation feature allows processing of up to 50 times longer results, depending on the results buffering queue size, with an absolute limit of 65535 characters.

The maximum number of result characters sent in a single EtherCAT frame is 478 bytes.

DMCC Result Feedback from PLC

The DMCC Result Feedback feature allows the reader to send DMCC responses back to the PLC.

For EtherCat, this feature is enabled by default. To disable the feature, use the `FFP.DMCC-RESULTS` DMCC command. Changing this configuration does not require you to reboot or reconnect reader.

When you enable the feature, the reader sends DMCC responses to the PLC using the same mechanism it uses to send read results. To distinguish DMCC results from read results, the DMCC Response bit is set to 1. The following PDOs contain DMCC information instead of read results:

Acquisition Object:

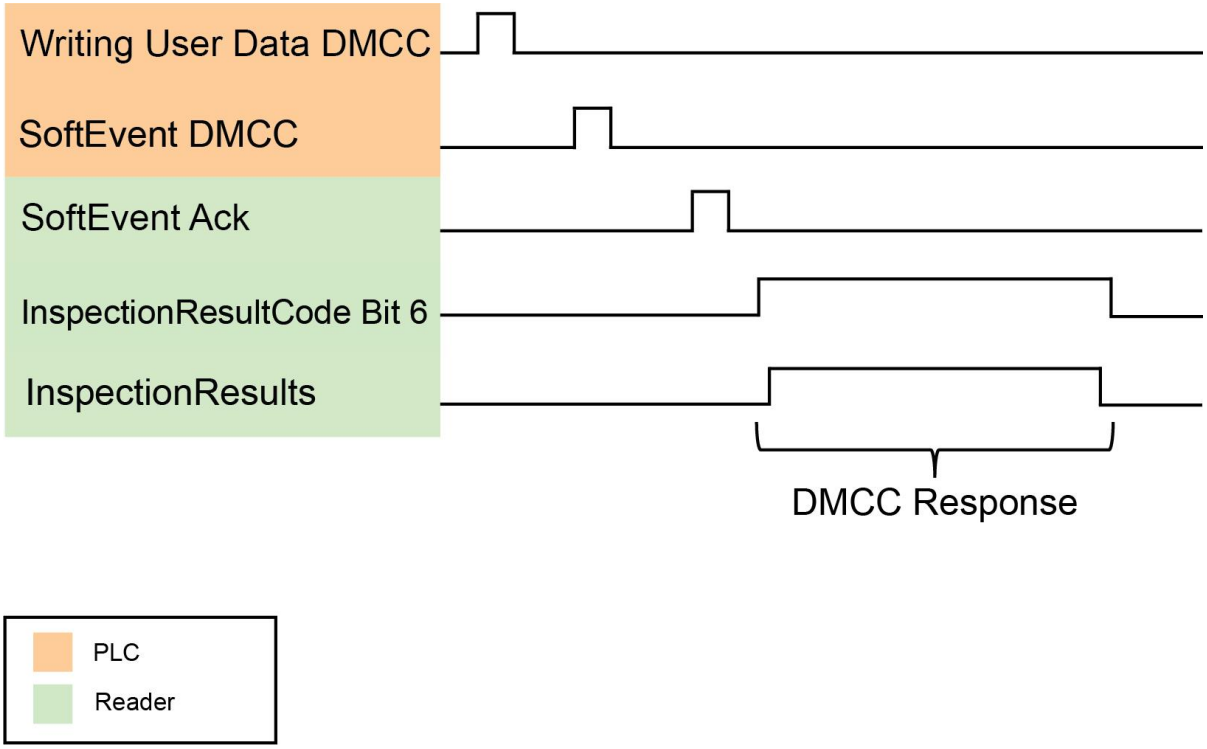
Index	Name	Description
6002:02	ResultID	Contains a DMCC result code: • 0: no error • 100: unidentified error • 101: invalid command • 102: invalid parameter • 103: incorrect checksum • 104: parameter rejected/altered due to reader state • 105: reader unavailable (offline)
6002:09	DMCC Response	Set to 1 to indicate a DMCC response.

Inspection Results Object:

Byte	Name	Description
0-1	Result Length	Actual number of bytes of DMCC result data contained in the *Result Data* field (16-bit integer).
2...	Result Data	DMCC result data (array of bytes).

You can use DMCC Result Feedback in combination with Results Fragmentation.

Note: The delivery of a result data can be interrupted and interlaced with the response of a DMCC if they are requested too close in time.



Extended User Data Mode

The Extended User Data mode allows you to transfer user data that exceeds the 480-byte limit. Typical uses cases include large DMCC commands, set match strings, or scripts.

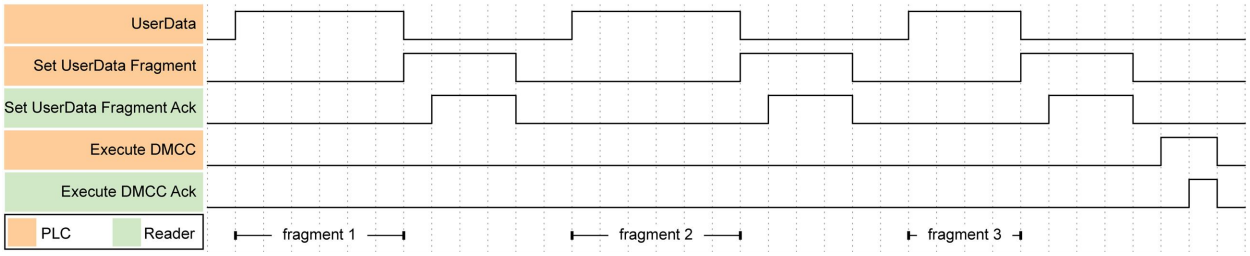
The reader receives a large payload in fragments of up to 480 bytes. Each fragment is written sequentially into the User Data object. The transfer is coordinated using the Set UserData Fragment and the Set UserData Fragment Ack bits. The reader accumulates all incoming fragments into a single User Data object until the PLC sends an Execute DMCC or Set Match String SoftEvent.

To use Extended User Data Mode:

1. Fill a fragment to its operational size. For more information on size options, see *User Data Object* on page 138.
2. Set the Set UserData Fragment bit to 1.
3. Wait until the reader sets the Set UserData Ack bit.
4. Reset the acknowledgment by setting the Set UserData Fragment bit to 0.
5. Repeat the previous steps until all data is sent.
6. Add the remainder of the data to the last fragment. Terminate the string with \0.

7. Invoke an Execute DMCC or Set MatchString SoftEvent.
8. Wait until the reader sets the Execute DMCC Ack or Set Match String Ack bit.

The following image illustrates the process:



If the Extended User Data transfer is interrupted, send an Execute DMCC SoftEvent with empty user data to reset the data mode.

Industrial Protocols for the Wireless DataMan

Special considerations must be taken into account when using any of the Industrial Ethernet protocols with DataMan wireless readers. Industrial protocol connections are established not between the PLC and the wireless device, but between the PLC and the DataMan Base Station.

The DataMan wireless reader may be, at times:

- powered down
- in hibernation mode
- physically out of range

Automatic power down and hibernation are features that preserve power and extend the time period between recharging.

The following sections detail the necessary configurations you have to set to communicate with the DataMan Base Station (through which information is routed to the DataMan wireless reader).

Protocol Operation

Because the wireless reader can become unavailable at times, the Industrial Ethernet protocols are actually run from the Base Station. The Base Station is directly wired to the Ethernet network and is online continuously (even when the reader is unavailable). In this way the Ethernet protocol operation is not disrupted when the reader is unavailable. This results in a few special considerations.

Ethernet Address

Because the Ethernet protocols are run from the Base Station, you must use the Base Station Ethernet address when configuring the protocol. For example, when configuring EtherNet/IP using the RSLogix5000 software package, you select the DM8000 reader but you must enter the IP address of the Base Station (not the reader). The PLC using the protocol must communicate with the Base Station.

PLC Triggering

Because the wireless reader can become unavailable at times, triggering from the PLC is not supported on wireless readers. The PLC will receive all read results triggered from the reader. Also, SoftEvents and DMCC commands are supported (when the reader is available).

SoftEvents

SoftEvents triggered from the PLC are supported. However, these events will fail if the reader is unavailable. Such a failure will be signaled by the ‘General Fault’ bit.

Note: Some operations assigned to SoftEvents are in general not supported on any handheld DataMan readers (for example, ‘Train Code’, ‘Train Focus’, ‘Train Brightness’). Unsupported operations will also result in the ‘General Fault’ signal being set.

DMCC

DMCC commands issued from the PLC are supported. However, these events will fail if the reader is unavailable. Depending on which Industrial Ethernet protocol is in use, the failure will be signaled either with the ‘General Fault’ signal or an error status return code. The new DMCC error return code ‘105’ has been established to signal that the reader is unavailable.

Note: When issuing DMCC commands from the PLC to a wireless reader, do not change the DMCC response format (that is, do not issue the command 'COM.DMCC-RESPONSE'). Doing this may disable DMCC operation on the wireless reader. This command can be used on non-Industrial Ethernet connections such as Telnet.

Offline Buffering

When the wireless reader is physically out of range of the Base Station, it will automatically buffer all reads. Buffer capacity varies according to a number of factors. However, typically several hundred reads can be stored. These buffered reads are also preserved if the reader enters hibernation (sleep mode). When the reader is brought back into range of the Base Station the buffered reads will automatically be downloaded.

If this offline buffering situation is anticipated in the user application, then Industrial Ethernet protocol buffering should also be enabled. Protocol buffering will allow a PLC to control the rate that read results are returned to the PLC. This prevents the PLC from becoming overwhelmed by a rapid flow of read results. Refer to the specific Industrial Protocol section of this document for details of enabling and utilizing protocol buffering.

Note: The Base Station can buffer up to 500 results.

Status of Industrial Protocols

The Status field of the industrial protocols (in Setup Tool, under Industrial Protocols tab) displays the last logged message of the status dialog. The message will remain displayed until another message is logged. If no industrial protocol is enabled, the status field remains blank.

Status messages vary by protocol, and some protocols do not log any messages. See the table below for a summary of messages for protocols that display them:

Protocol/Reader	Message	Description
Wi-Fi readers	Industrial ethernet protocols shutdown	Displayed if no protocol is running
	Proxy %s connection opening	Opening connection to Wi-Fi reader, s == connection type ("Command", "Result", or "User DMCC")
	Proxy %s open	Connection open, s == connection type ("Command", "Result", or "User DMCC")
	Proxy %s not open	failed to open connection, s == connection type ("Command", "Result", or "User DMCC")
	Proxy %s closed	connection has been closed, s == connection type ("Command", "Result", or "User DMCC")

Protocol/Reader	Message	Description
Modbus TCP	Modbus TCP starting...	Protocol starting
	Modbus TCP: Server Setup Failed	Failed to start server
	Modbus TCP running	Protocol running
	Modbus TCP: Add block %s failed	Data block configuration error
	Modbus TCP control block %s	Status of ModbusTCP data block, s == “info” or “error”
SLMP Protocol	Opening SLMP scanner connection %s:%x	starting SLMP connection, s==IP address, x==port number
	SLMP scanner connection established %s:%x	SLMP connection established, s==IP address, x==port number
	SLMP scanner connection %s:%x rejected	PLC has rejected the connection, s==IP address, x==port number
	SLMP scanner connection %s:%x faulted	Miscellaneous connection fault has occurred, s==IP address, x==port number
	SLMP scanner poll rate set too low	Configured SLMP poll rate is set too low
	SLMP scanner connection normal %s:%x	SLMP connection has been restored, s==IP address, x==port number
	SLMP Scanner control block %s	Status of SLMP data block, s == “info” or “error”
	SLMP Scanner failed transfering '%s' block	Comm error, failed to transfer a data block, s==block name

Copyright © 2026
Cognex Corporation. All Rights Reserved.