Datalogic Matrix & built-in EtherNet/IP

Installation and User Guide

December 2011

REVISION 1.00

Master Revision History

Revision	Date	Author(s)	Change Description
1.00	01/12/2011	J.Wendorf, D.Natati	Initial Revision

Table of Contents

I. Introduction

A. Overview

The Matrix 410™ is the modular, flexible and versatile compact bar code 2D reader for industrial applications embedding 1.3 and 2.0 megapixel sensors. Matrix 410™ features excellent performances in reading and verifying, easy setup, thanks to the X-PRESS™ interface and patented Blue Diamonds™ system, ease of use, extreme flexibility, high versatility and industrial strength. The integration of EtherNet/IP and TCP-IP protocol expands the networking and remote diagnostic capabilities of the reader. The possibility of sending diagnostic or statistical messages, even through the Web, provides a great advantage for service and maintenance and reduces plant downtime costs.

B. Reference Terms

Matrix – Refers to the Matrix family of devices

Reader – Refers to the Matrix device

Client – Refers to the ControlLogix PLC

C. References

- Volume I: CIP Common Specification, Release 1.0, ©2003 ODVA
- Volume 2: EtherNet/IP Adaptation of CIP, Release 1.0, ©2003 ODVA

- To find more information on the ControlLogix system, including EtherNet/IP go to http://ab.rockwellautomation.com/

II. EtherNet/IP Overview

A LITTLE BACKGROUND

Most people who work in an office associate the term “Ethernet” with the physical cable behind their desk. This cable connects their office PC to the printers and servers of the local network and the infinite web sites on the Internet. This cable is only the physical part of Ethernet, the media carrying Ethernet messages to your PC. On this wire is a whole series of communication protocols such as IP, the Internet Protocol; TCP, the Transport Control Protocol; and various Microsoft protocols such as NetBEUI. This suite of protocols works well for the office environment. It allows users to share files, access printers, send email, search the Internet and perform all the other communications used in the office environment.

The needs of the factory floor are much different with some very special requirements. Instead of accessing files and printers, factory floor controllers must access data embedded in drive systems, operator workstations and I/O devices. Instead of letting a user wait while a task is being performed, factory floor data communications needs are real-time or very close to real time. Terminating the fill operation on a bottle requires much more time-precise communications than accessing the next page of an Internet site.

Traditionally, Ethernet had only limited acceptance in Industrial Automation. Until recently the expense, lack of intelligent switches and routers and the domination of large vendors with proprietary protocols prevented the wide acceptance of Ethernet on the factory floor. Now with prices falling, PCs with inherent Ethernet capability moving in droves onto the factory floor and intelligent switches and routers, Ethernet is gaining acceptance. Only the lack of a widely accepted, flexible application layer targeted to Industrial Automation has prevented its complete acceptance.

ETHERNET/IP

Ethernet/IP is the application layer protocol that can meet this challenge. Four independent groups have joined forces to develop and promote EIP as a public domain Ethernet application layer for Industrial Automation. These groups include the Open DeviceNet Vendor Association (ODVA), the Industrial Open Ethernet Association (IOANA), Control Net International (CI) and the Industrial Ethernet Association (IEA). The goals of this effort illustrate how EIP provides a wide-ranging, comprehensive, certifiable standard suitable to a wide variety of automation devices:

Ethernet/IP uses all the transport and control protocols used in traditional Ethernet including the Transport Control Protocol (TCP), the Internet Protocol (IP) and the media access and signaling technologies found in off-the-shelf Ethernet interface cards. Building on these standard PC technologies means that EIP works transparently with all the standard off-the-shelf Ethernet devices found in today’s marketplace. It also means that EIP can be easily supported on standard PCs and all their derivatives. Even more importantly, basing EIP on a standard technology platform ensures that EIP will move forward as the base technologies evolve in the future.

ETHERNET/IP IS A CERTIFIABLE STANDARD

The groups supporting EIP plan to ensure a comprehensive, consistent standard by careful, multi-vendor attention to the specification and through certified test labs as has been done with DeviceNet and ControlNet. Certification programs modeled after the programs for DeviceNet and ControlNet will ensure the consistency and quality of field devices.
EIP is built on a widely accepted protocol layer

EIP is constructed from a very widely implemented standard used in DeviceNet and ControlNet called the Control and Information Protocol (CIP) and is illustrated on the attached drawing. This standard organizes networked devices as a collection of objects. It defines the access, object behavior and extensions which allow widely disparate devices to be accessed using a common mechanism. Over 300 vendors now support the CIP protocol in present day products. Using this technology in EIP means that EIP is based on a widely understood, widely implemented standard that does not require a new technology shakedown period.

CIP OVERVIEW

The Communications and Information Protocol (CIP) is a communications protocol for transferring automation data between two devices. In the CIP Protocol, every network device represents itself as a series of objects. Each object is simply a grouping of the related data values in a device. For example, every CIP device is required to make an Identity object available to the network. The identity object contains related identity data values called attributes. Attributes for the identity object include the vendor ID, date of manufacture, device serial number and other identity data. CIP does not specify at all how this object data is implemented, only what data values or attributes must be supported and that these attributes must be available to other CIP devices.

The Identity object is an example of a required object. There are three types of objects defined by the CIP protocol:

REQUIRED OBJECTS

Required objects are required by the specification to be included in every CIP device. These objects include the Identity object, a Message Router object and a Network object.
The identity object contains related identity data values called attributes. Attributes for the identity object include the vendor ID, date of manufacturer, device serial number and other identity data.

A Network object contains the physical connection data for the object. For a CIP device on DeviceNet the network object contains the MacID and other data describing the interface to the CAN network. For EIP devices, the network object contains the IP address and other data describing the interface to the Ethernet port on the device.

APPLICATION OBJECTS

Application objects are the objects that define the data encapsulated by the device. These objects are specific to the device type and function. For example, a Motor object on a Drive System has attributes describing the frequency, current rating and motor size. An Analog Input object on an I/O device has attributes that define the type, resolution and current value for the analog input.
These application layer objects are predefined for a large number of common device types. All CIP devices with the same device type (Drive Systems, Motion Control, Valve Transducer…etc) must contain the identical series of application objects. The series of application objects for a particular device type is known as the device profile. A large number of profiles for many device types have been defined. Supporting a device profile allows a user to easily understand and switch from a vendor of one device type to another vendor with that same device type.

A device vendor can also group Application Layer Objects into assembly objects. These super objects contain attributes of one or more Application Layer Objects. Assembly objects form a convenient package for transporting data between devices. For example, a vendor of a Temperature Controller with multiple temperature loops may define assemblies for each of the temperature loops and an assembly with data from both temperature loops. The user can than pick the assembly that is most suited for the application and how often to access each assembly. For example, one temperature assembly may be configured to report every time it changes state while the second may be configured to report every one-second regardless of a change in state.
Assemblies are usually predefined by the vendor but CIP also defines a mechanism in which the user can dynamically create an assembly from application layer object attributes.

VENDOR SPECIFIC OBJECTS

Objects not found in the profile for a device class are termed Vendor Specific. These objects are included by the vendor as additional features of the device. The CIP protocol provides access to these vendor extension objects in exactly the same method as either application or required objects. This data is strictly of the vendors choosing and is organized in whatever method makes sense to the device vendor.
In addition to specifying how device data is represented to the network, the CIP protocol specifies a number of different ways in which that data can be accessed such as cyclic, polled and change-of-state.

ADVANTAGES TO EIP

The advantages of the CIP protocol layer over EIP are numerous. The consistent device access means that a single configuration tool can configure CIP devices on different networks from a single access point without using vendor specific software. The classification of all devices as objects decreases the training and startup required when new devices are brought online. EIP provides improved response time and greater data throughput than DeviceNet and ControlNet. EIP links devices from the sensor bus level to the control level to the enterprise level with a consistent application layer interface.

There are numerous application layer competitors to EIP including Modbus/TCP from Groupe Schneider, PROFInet from Siemens, HSE Fieldbus from the Fieldbus foundation and other vendors. Unfortunately space prevents a detailed review of each of these products. However, none of these

competitors can provide the vendor support, flexibility and total architecture support offered by the implementation of CIP over Ethernet.

USER CHALLENGES

EIP implementation is not without challenges. Two of the most important challenges to the first time user include training and network configuration. One common problem is the lack of trained staff who understand both the IT fundamentals and the automation network. A collaborative effort between the IT and Automation staffs is required to successfully implement the first Ethernet/IP system. A second challenge is proper network configuration. Planning your Ethernet factory automation infrastructure is essential. Careful identification of all your control loops, choosing the correct routers, switches and paths and documenting your network properly are requisites for a communications network which meets your production goals and requires little ongoing maintenance.

III. Object Model

The Object Model is the logical grouping of attributes accessible from the Matrix.

The MATRIX supports 6 required objects

- Identity Object (0x01)
- Message Router Object (0x02)
- Assembly Object (0x04)
- Connection Manager Object (0x06)
- TCP Object (0xF5)
- Ethernet Link Object (0xF6)

The MATRIX supports 5 vendor specific objects

- Item Object (0x64)
- General Purpose Input Object (0x65)
- General Purpose Output Object (0x66)
- Statistics Object (0x67)
- Diagnostics Object (0x68)

The following are the ODVA data types

Data Type	Description
USINT	Unsigned Short Integer (8-bits)
UINT	Unsigned Integer (16-bit)
UDINT	Unsigned Double Integer (32-bit)
STRING	Character String (1 byte per character)
BYTE	Bit String (8-bits)
WORD	Bit String (16-bits)
DWORD	Bit String (32-bits)

A. Identity Object (0x01)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Vendor ID	UINT	850_{DEC}	Get
2	Device Type	UINT	00_{HEX}	Get
3	Product Code	UINT	3000_{DEC}	Get
4	Product Major Revision Product Minor Revision	USINT USINT	01 01	Get
5	Status Word (see below for definition)	WORD	See Below	Get
6	Serial Number	UDINT	Unique 32 Bit Value	Get
7	Product Name Structure of: Product Name Size Product Name String	 USINT USINT [ ]	 26 “Unattended Scanning System”	Get
64_{HEX}	Product Model Number Structure of: Product Model Number Size Product Model Number String	 USINT USINT [ ]	 20 “Product Model Number”	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get_Attribute_Single
05_{HEX}	No	Yes	Reset

4. Instance Attribute Semantics

Vendor ID

Vendor IDs are used to identify the manufacturer of a product. Vendor IDs are managed by ODVA. The Vendor ID for Datalogic. is 850.

Device Type

Device Types are used to identify the device profile used for a product. Device profiles define the minimum set of attributes and objects required for conformance. The list of Device Types is managed by ODVA. 0 (Generic Device) is the Device Type for this product.

Product Code

The Product Code is a number (0-65535) used to identify a vendor's product within the device type. The product code refers to the behavior of the product on a given network and doesn't affect functionality not seen by the network. The Product Code for this series of devices is 3000.

Product Major/Minor Revision

The Major and Minor Revision identify the revision of the item the Identity Object represents. Zero is invalid for either field. The current revision of the product is 1.01.

Status Word

The Status Word represents the status of the complete device. Only bit zero ("Owned") is monitored for this device.

Bit	Name	Definition
0	Owned	0 - No I/O Connection Allocated 1 - I/O Connection Allocated
1 – 15	Unused	Unused

Serial Number

The Serial Number is a 32-bit number used in conjunction with the Vendor ID to form a unique number on DeviceNet. Each vendor is responsible for guaranteeing the uniqueness of the serial number across all of its devices.

Product Name

The Product Name is a string (up to 32 characters) that identifies a product on the network. The same Product Code may have a variety of product name strings. The Product Name for this family of products is “Unattended Scanning System”. The first byte in the access of this attribute contains the length of the string (26 bytes).

Product Model Number

The Product Model Number is a vendor specific attribute used to identify the reader. The string length varies from 0 to 128 characters. The Product Model Number is set prior to shipping the product. The default string is “Product Model Number” with a length of 20 bytes.

B. Message Router Object (0x02)

<<< This object has no supported attributes or services >>>

C. Assembly Object (0x04)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get
2	Max Instance	UINT	81_{HEX}	Get

2. Instance Attributes (Instance 0x64 – "Input Instance 1")

Attribute ID	Name	Data Type				Access Rule
3	Polled Input Data Structure of:		*(Structure item location)*			Get
			Class	Instance	Attribute	
	Item Sequence Number	USINT	0x64	0x01	0x04	
	Item Status	UINT	0x64	0x01	0x02	
	Item Data Size	UINT	0x64	0x01	0x03	
	Local Presence And Input Bits	BYTE	0x65	0x01	0x03	
	Failure Mask	BYTE	0x68	0x01	0x01	

3. Instance Attributes (Instance 0x65 – "Input Instance 2")

Attribute ID	Name	Data Type				Access Rule
3	Polled Input Data Structure of:		*(Structure item location)*			Get
			Class	Instance	Attribute	
	Item Sequence Number	USINT	0x64	0x01	0x04	
	Item Status	UINT	0x64	0x01	0x02	
	Item Data Size	UINT	0x64	0x01	0x03	
	Local Presence And Input Bits	BYTE	0x65	0x01	0x03	
	Failure Mask	BYTE	0x68	0x01	0x01	
	Fragment Sequence Number	USINT	0x64	0x01	0x05	
	Fragment Data Size	UINT	0x64	0x01	0x08	
	Fragment Data [ ]	BYTES [ ]	0x64	0x01	0x09	

4. Instance Attributes (Instance 0x70 – "Output Instance 1")

Attribute ID	Name	Data Type				Access Rule
3	Polled Output Data Structure of:		*(Structure item locaiton)*			Get / Set
			Class	Instance	Attribute	
	Last Item Sequence Number Received	USINT	0x64	0x01	0x06	
	Remote Presence And Output Bits	UINT	0x66	0x01	0x03	

5. Instance Attributes (Instance 0x71 – "Output Instance 2")

Attribute ID	Name	Data Type				Access Rule
3	Polled Output Data Structure of:		*(Structure item location)*			Get / Set
			Class	Instance	Attribute	
	Last Item Sequence Number Received	USINT	0x64	0x01	0x06	
	Remote Presence And Output Bits	UINT	0x66	0x01	0x03	
	Last Fragment Sequence Number Received	USINT	0x64	0x01	0x07	

6. Instance Attributes (Instance 0x80 – "Configuration Instance")

Many I/O clients include a configuration path when opening an I/O connection to the server. There is no configuration data, but the instance number is necessary.

7. Instance Attributes (Instance 0x81 – "Heartbeat / Input Only Instance")

This instance allows clients to monitor input data without providing output data. Since there is no consume data, no attributes are supported.

8. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get_Attribute_Single
10_{HEX}	No	Yes	Set_Attribute_Single

9. Class Attribute Semantics

Max Instance

The Max Instance attribute lists the highest instance number (currently 0x81) that exists in the Assembly Object.

10. Instance Attribute Semantics

All Instance attributes in the Assembly Object are composed of attributes from other objects. See the attribute definitions in their respective objects.

D. Connection Manager Object (0x06)

<<< This object has no supported attributes or services >>>

E. TCP Object (0xF5)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Status	DWORD	1	Get
2	Configuration Capability	DWORD	0	Get
3	Configuration Control	DWORD	0	Get
4	Physical Link Object Structure of: Path Size Path	 UINT WORDS [ ]	 2 0x20F6 0x2401	Get
5	Interface Configuration Structure of: IP Address Network Mask Gateway Address Name Server Name Server 2 Domain Name Size Domain Name	 UDINT UDINT UDINT UDINT UDINT UINT STRING	 0 0 0 0 0 0 0	Get
6	Host Name Structure of: Host Name Size Host Name	 UINT STRING	 0 0	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get_Attribute_Single

4. Instance Attribute Semantics

Status

The Status attribute is a bitmap that indicates the status of the TCP/IP network interface. This attribute value is fixed at 1.

Bit(s)	Name	Definition
0 – 3	Interface Configuration Status	0 – Interface Configuration attribute not configured 1 – Interface Configuration attribute contains a valid configuration 2 – 15 Reserved for future use
4 – 31	Reserved	Unused

Configuration Capability

The Configuration Capability attribute is a bitmap that indicates the device's support for optional network configurations. This attribute value is fixed at 0 since network configuration information is not available to the 6x00 EtherNet/IP Reader.

Bit(s)	Name	Definition
0	BOOTP Client	1 (TRUE) indicates the device is capable of obtaining its network configuration via BOOTP
1	DNS Client	1 (TRUE) indicates the device is capable of resolving host names by querying a DNS server
2	DHCP Client	1 (TRUE) indicates the device is capable of obtaining its network configuration via DHCP
3	DHCP-DNS Update	1 (TRUE) indicates the device is capable of sending its host name in the DHCP request
4	Configuration Settable	1 (TRUE) indicates the Interface Configuration attribute is settable. This device does not allow this
5 – 31	Reserved	Unused

Configuration Control

The Configuration Control attribute is a bitmap used to control network configuration attributes. This attribute value is fixed at 0 since network configuration information is not available to the 6x00 EtherNet/IP Reader.

Bit(s)	Name	Definition
0 – 3	Startup Configuration	0 – The device uses the interface configuration values stored in non-volatile memory 1 – The device obtains the interface configuration values via BOOTP at startup 2 – The device obtains the interface configuration values via DHCP at startup 3 – 15 Reserved for future use
4	DNS Enable	1 (TRUE) the device shall resolve host names by querying a DNS server
5 – 31	Reserved	Unused

Physical Link Object

This attribute identifies the object associated with the underlying physical communications interface. The first byte is the path size in words, followed by the path to the object. Ethernet is always used for this application, so the path value is fixed.

Interface Configuration

This attribute contains the configuration parameters required to operate as a TCP/IP node. The following are the fields of the Interface Configuration structure. These values are modifiable via Visiset only, so the attributes are read only.

Name	Data Type	Meaning
IP Address	UDINT	The device's IP Address
Network Mask	UDINT	The device's network mask. The network mask is used when the IP network has been partitioned into subnets. The network mask is used to determine whether an IP address is located on another subnet.
Gateway Address	UDINT	The IP address of the device's default gateway. When a destination IP address is on a different subnet, packets are forwarded to the default gateway for routing to the destination subnet.
Name Server	UDINT	The IP address of the primary name server. The name server is used to resolve host names. For example, that might be contained in a CIP connection path.
Name Server 2	UDINT	The IP address of the secondary name server. The secondary name server is used when the primary name server is not available, or is unable to resolve a host name.
Domain Name Size	UINT	The length of the Domain Name in bytes.

Name	Data Type	Meaning
Domain Name	STRING	The default domain name. The default domain name is used when resolving host names that are not fully qualified. For example, if the default domain name is “odva.org”, and the device needs to resolve a host name of “plc”, then the device will attempt to resolve the host name as “plc.odva.org”.

Host Name

The Host Name attribute contains the device’s host name. The host name is used when the device supports DHCP-DNS. Since this device doesn’t support DHCP-DNS, this attribute is NULL.

F. Ethernet Link Object (0xF6)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Interface Speed	UDINT	100	Get
2	Interface Flags	DWORD	3	Get
3	Physical Address	USINT [ ]	0	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get_Attribute_Single

4. Instance Attribute Semantics

Interface Speed

The Interface Speed attribute indicates whether the device is running at 10Mbps, 100Mbps, 1Gbps, etc… The attribute resolution is in Mbps, so if the interface is running at 100Mbps, the attribute value is 100.

Interface Flags

The Interface Flags attribute contains status and configuration information about the physical interface as follows:

Bit(s)	Name	Definition
0	Link Status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. 0 indicates an inactive link; 1 indicates an active link. The determination of link status is implementation specific. In some cases devices can tell whether the link is active via hardware/driver support. In other cases, the device may only be able to tell whether the link is active by the presence of incoming packets.
1	Half/Full Duplex	0 indicates the interface is running half duplex; 1 indicates full duplex. Note that if the Link Status flag is 0, then the value of the Half/Full Duplex flag is indeterminate.
2 – 31	Reserved	Set to zero.

Physical Address

The Physical Address attribute contains the interface's MAC layer address. The Physical Address is an array of octets (bytes). The recommended display format is "XX-XX-XX-XX-XX-XX" starting with the first octet. This attribute is read only.

G. Item Object (0x64)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get
2	Maximum Item Data Buffer Size	UINT	450	Get
3	Maximum Fragment Data Buffer Size	UINT	450	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Expected Number Of Barcodes Per Item	USINT	1	Get
2	Item Status	UINT	0	Get
3	Item Data Size	UINT	0	Get
4	Item Sequence Number	USINT	0	Get
5	Fragment Sequence Number	USINT	0	Get
6	Last Item Sequence Number Received	USINT	0	Get / Set
7	Last Fragment Sequence Number Received	USINT	0	Get / Set
8	Fragment Data Size	UINT	0	Get
9	Fragment Data [ ]	BYTES [ ]	0	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
05_{HEX}	No	Yes	Reset[1]
$0E_{HEX}$	Yes	Yes	Get Attribute Single
10_{HEX}	No	Yes	Set Attribute Single

4. Class Attribute Semantics

Maximum Item Data Buffer Size

The Maximum Item Data Buffer Size is the maximum length of Item Data. If this attribute is greater than the Maximum Fragment Data Buffer Size, fragmentation is used to pass the Item Data. The valid range is 1 – 65535. The default size is 450.

[1] This Service Code is used to flush the Item Buffer Queue.

Maximum Fragment Data Buffer Size

The Maximum Fragment Data Buffer Size is the length of the fragment buffer. This value must be less than or equal to the Maximum Item Data Buffer Size. The valid range is 1 – 450. The default size is 450 (no fragmentation is used).

5. Instance Attribute Semantics

Expected Number Of Barcodes Per Item

The Matrix supports Item Data packets with multiple barcodes. The Expected Number of Barcodes Per Item is the number of barcodes embedded in the Item Data. The default size is one barcode.

Item Status

The Item Status Code is the status of the current Item Data packet and is returned with every Item Data transfer. The table below shows the status codes and their meanings.

Item Status Code	Item Status Name
0x0000	Good Read
0x0001	Complete, No Read
0x0002	Partial Read
0x0003	Multiple Read
0x0004	Wrong Read

Item Data Size

The Item Data Size is the total size of the Item Data. If the Item Data Size is greater than the Maximum Fragment Data Buffer Size, fragmentation is used (see the fragmentation example at the end of this section).

Item Sequence Number

The Item Sequence Number is incremented by one on every new Item Data production. The Item Sequence Number is set to zero at power up. Once an Item Data packet is ready to transmit, the Item Sequence Number is set to one. The Item Sequence Number reloads to one since zero is an invalid number.

Fragment Sequence Number

The Fragment Sequence Number is set to one on the first fragment of the Item Data production. The Fragment Sequence Number is incremented by one on every new fragment. If fragmentation isn't used, this value is fixed at one.

Last Item Sequence Number Received

The Last Item Sequence Number Received is written with the Item Sequence Number by the EtherNet/IP client to acknowledge the receipt of the Item Data. If fragmentation is used, this value isn't written until the complete message is received.

Last Fragment Sequence Number Received

The Last Fragment Sequence Number Received is written with the Fragment Sequence Number by the EtherNet/IP client to acknowledge the receipt of an individual fragment. If fragmentation isn't used, this value doesn't need to be written.

Fragment Data Size

The Fragment Data Size is the length of the data (in bytes) stored in the Fragment Data attribute. If fragmentation is used, this value equals the Maximum Fragment Data Buffer Size until the last fragment.

Fragment Data

This attribute stores the Fragment Data. If the Item Data Size is less than the Maximum Fragment Data Buffer Size, this attribute stores the complete Item Data. If the Item Data Size is greater than the Maximum Fragment Data Buffer Size, this attribute stores the individual fragments of data.

6. Item Data Handshaking Example (No Fragmentation)

The following is an example of how to send 3 Item Data packets, each 300 bytes, with a fragment size of 450.

To Datalogic barcode reader from EIP Client		**To EtherNet/IP Client from Datalogic Barcode Reader**					
Last Item Sequence Number	Last Fragment Sequence Number	Item Sequence Number	Fragment Sequence Number	Item Size	Fragment Size	Fragment Data Buffer	**Description**
0	0	0	0	0	0	NULL	Power Up
		1	1	300	300	[0-299]	Datalogic sends complete Item Data 1
1	1						EIP Client acknowledges Item Data 1
		2	1	300	300	[0-299]	Datalogic sends complete Item Data 2
2	1						EIP Client acknowledges Item Data 2
		3	1	300	300	[0-299]	Datalogic sends complete Item Data 3
3	1						EIP Client acknowledges Item Data 3

7. Item Data Handshaking Example (With Fragmentation)

The following is an example of how to send 2 Item Data packets, each 800 bytes, with a fragment size of 128.

To Datalogic barcode reader from EIP Client		**To EtherNet/IP Client from Datalogic Barcode Reader**					
Last Item Sequence Number	Last Fragment Sequence Number	Item Sequence Number	Fragment Sequence Number	Item Size	Fragment Size	Fragment Data Buffer	**Description**
0	0	0	0	0	0	NULL	Power Up
		1	1	800	128	[0-127]	Datalogic sends fragment 1, Item Data Buffer 1
0	1						EIP Client acknowledges fragment 1
		1	2	800	128	[128-255]	Datalogic sends fragment 2, Item Data Buffer 1
0	2						EIP Client acknowledges fragment 2
		1	3	800	128	[256-383]	Datalogic sends fragment 3, Item Data Buffer 1
0	3						EIP Client acknowledges fragment 3
		1	4	800	128	[384-511]	Datalogic sends fragment 4, Item Data Buffer 1
0	4						EIP Client acknowledges fragment 4
		1	5	800	128	[512-639]	Datalogic sends fragment 5, Item Data Buffer 1
0	5						EIP Client acknowledges fragment 5
		1	6	800	128	[640-767]	Datalogic sends fragment 6, Item Data Buffer 1
0	6						EIP Client acknowledges fragment 6
		1	7	800	32	[768-799]	Datalogic sends fragment 7, Item Data Buffer 1
1	7						EIP Client acknowledges whole Item Data Buffer 1
		2	1	800	128	[0-127]	Datalogic sends fragment 1, Item Data Buffer 2
1	1						EIP Client acknowledges fragment 1
		2	2	800	128	[128-255]	Datalogic sends fragment 2, Item Data Buffer 2
1	2						EIP Client acknowledges fragment 2
		2	3	800	128	[256-383]	Datalogic sends fragment 3, Item Data Buffer 2
1	3						EIP Client acknowledges fragment 3
		2	4	800	128	[384-511]	Datalogic sends fragment 4, Item Data Buffer 2
1	4						EIP Client acknowledges fragment 4
		2	5	800	128	[512-639]	Datalogic sends fragment 5, Item Data Buffer 2
1	5						EIP Client acknowledges fragment 5
		2	6	800	128	[640-767]	Datalogic sends fragment 6, Item Data Buffer 2
1	6						EIP Client acknowledges fragment 6
		2	7	800	32	[768-799]	Datalogic sends fragment 7, Item Data Buffer 2
2	7						EIP Client acknowledges whole Item Data Buffer 2

H. General Purpose Input Object (0x65)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
3	Presence and Input Bits	BYTE	0	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get Attribute Single

4. Instance Attribute Semantics

Presence and Input Bits

The Presence and Input Bits attribute is a bitmap used to monitor the status of the discrete inputs on the Matrix reader.

Bit(s)	Name	Definition
0	State of Input Bit 0	1 = ON; 0 = OFF
1	State of Input Bit 1	1 = ON; 0 = OFF
2	State of Input Bit 2	1 = ON; 0 = OFF
3	State of Input Bit 3	1 = ON; 0 = OFF
4	State of Input Bit 4	1 = ON; 0 = OFF
5	State of Input Bit 5	1 = ON; 0 = OFF
6	State of Input Bit 6	1 = ON; 0 = OFF
7	Local Presence Bit	1 = ON; 0 = OFF *(Used when the presence input is connected to the Matrix reader.)*

I. General Purpose Output Object (0x66)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
3	Presence and Output Bits	BYTE	0	Get / Set

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get Attribute Single
10_{HEX}	No	Yes	Set Attribute Single

4. Instance Attribute Semantics

Presence and Output Bits

The Presence and Output Bits attribute is a bitmap used to control the state of the discrete outputs on the Matrix reader.

Bit(s)	Name	Definition
0	State of Output Bit 0	1 = ON; 0 = OFF
1	State of Output Bit 1	1 = ON; 0 = OFF
2	State of Output Bit 2	1 = ON; 0 = OFF
3	State of Output Bit 3	1 = ON; 0 = OFF
4	State of Output Bit 4	1 = ON; 0 = OFF
5	State of Output Bit 5	1 = ON; 0 = OFF
6	State of Output Bit 6	1 = ON; 0 = OFF
7	Remote Presence Bit	1 = ON; 0 = OFF *(Used when the presence input is provided by the EtherNet/IP client.)*

J. Statistics Object (0x67)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Good Read Count	UDINT	0	Get
2	No Read Count	UDINT	0	Get
3	Partial Read Count	UDINT	0	Get
4	Multiple Read Count	UDINT	0	Get
5	Wrong Read Count	UDINT	0	Get
6	Item Count	UDINT	0	Get
7	Missed Item Count	UDINT	0	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
05_{HEX}	No	Yes	Reset[1]
$0E_{HEX}$	Yes	Yes	Get Attribute Single

4. Instance Attribute Semantics

Attribute	Description
Good Read Count	Successful read count
No Read Count	Presence indicated a barcode, but no barcode data was read
Partial Read Count	Only part of the barcode was read
Multiple Read Count	Multiple barcodes were successfully read
Wrong Read Count	Unexpected number of barcodes read
Item Count	Number of items processed
Missed Item Count	Number of items lost due to queue overflows

[1] Reset Instance 1, Attributes l-7 to 0.

K. Diagnostics Object (0x68)

1. Class Attributes (Instance 0)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Revision	UINT	1	Get

2. Instance Attributes (Instance 1)

Attribute ID	Name	Data Type	Data Value	Access Rule
1	Failure Mask 0x01 – "Input Failure 0x02 – "Communications Failure" 0x04 – "Reader Failure" 0x08 – "SW Error" 0x10 – "Remote Failure"	USINT	0	Get
2	Failure Subcause	UINT	0	Get
3	Failure String **Structure of:** String Length Message String	 USINT USINT[128]	 0 0	Get

3. Common Services

Service Code	Implemented for		Service Name
	Class Level	Instance Level	
$0E_{HEX}$	Yes	Yes	Get Attribute Single

4. Instance Attribute Semantics

Failure Mask

The Failure Mask is set when an error occurs with the reader. Below is the table of Failure Mask codes.

Failure Mask Code	Name
0x01	Input Failure
0x02	Communications Failure
0x04	Reader Failure
0x08	Software Error
0x10	Remote Failure

Failure Subcause

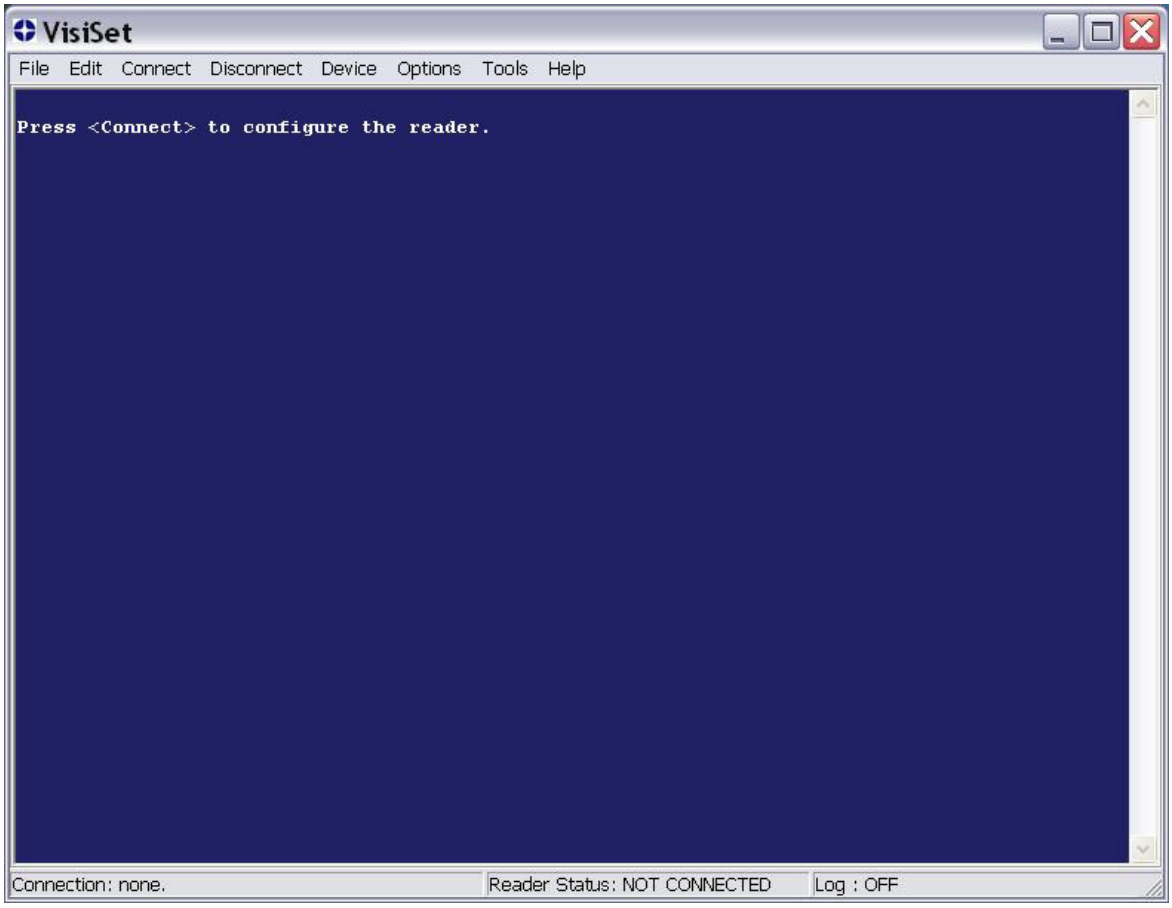
The Failure Subcause is an integer value set by Datalogic on a failure. See Datalogic for a list of Failure Subcauses and their meanings.

Failure String

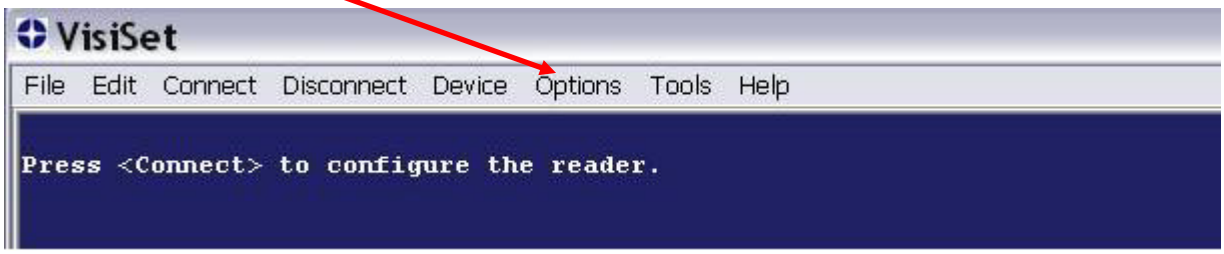
The Failure String is set by Datalogic on a failure. See Datalogic for a list of Failure Strings and their meanings.

IV. Configuring the Matrix for EtherNet/IP

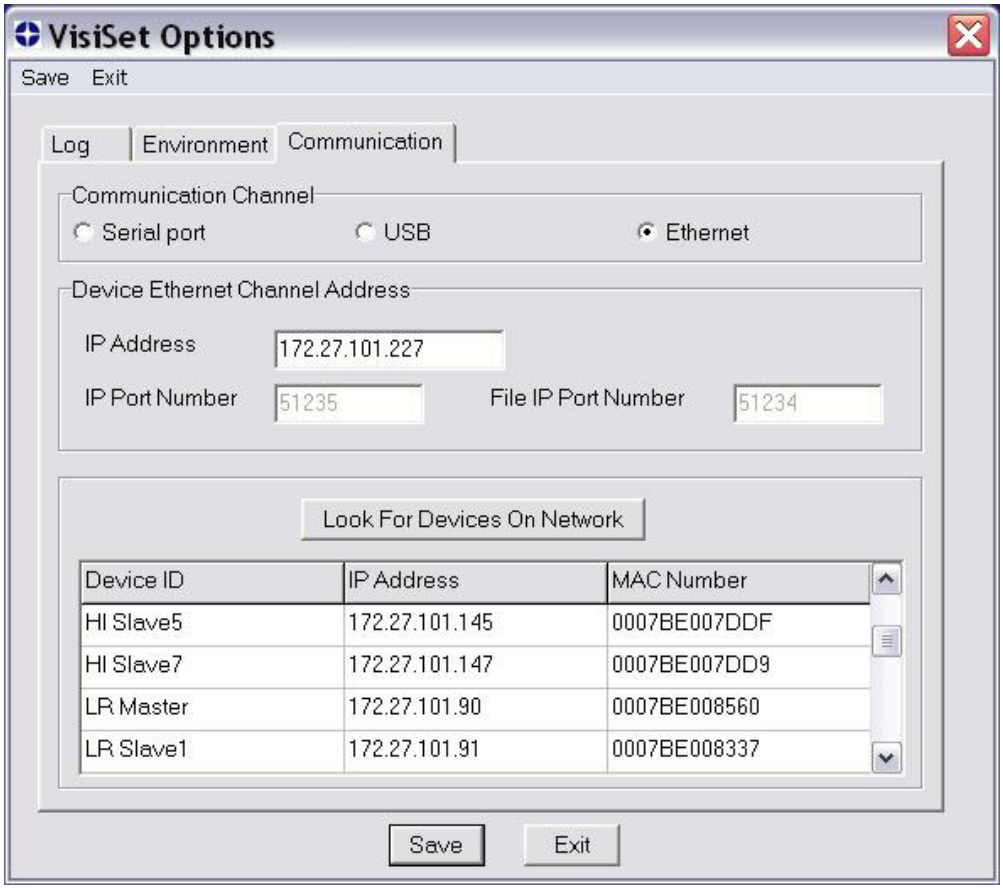
- **VISISET** is the Windows application used to configure Matrix: opening it the following initial window appears



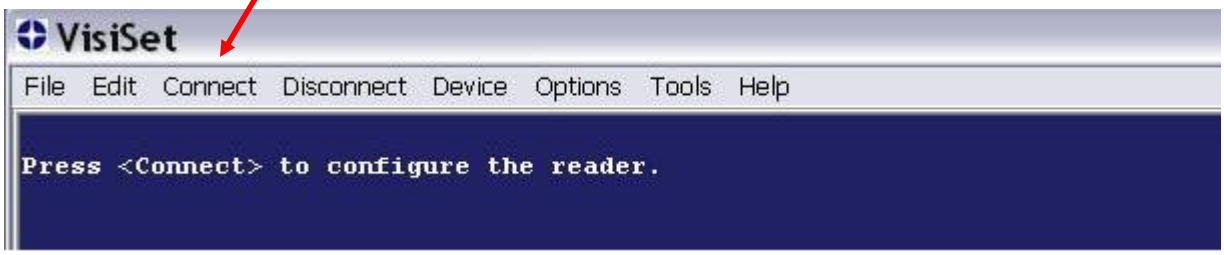
- Select "**Options**" on the main bar



then the desired communication option (the example below shows an Ethernet setup)

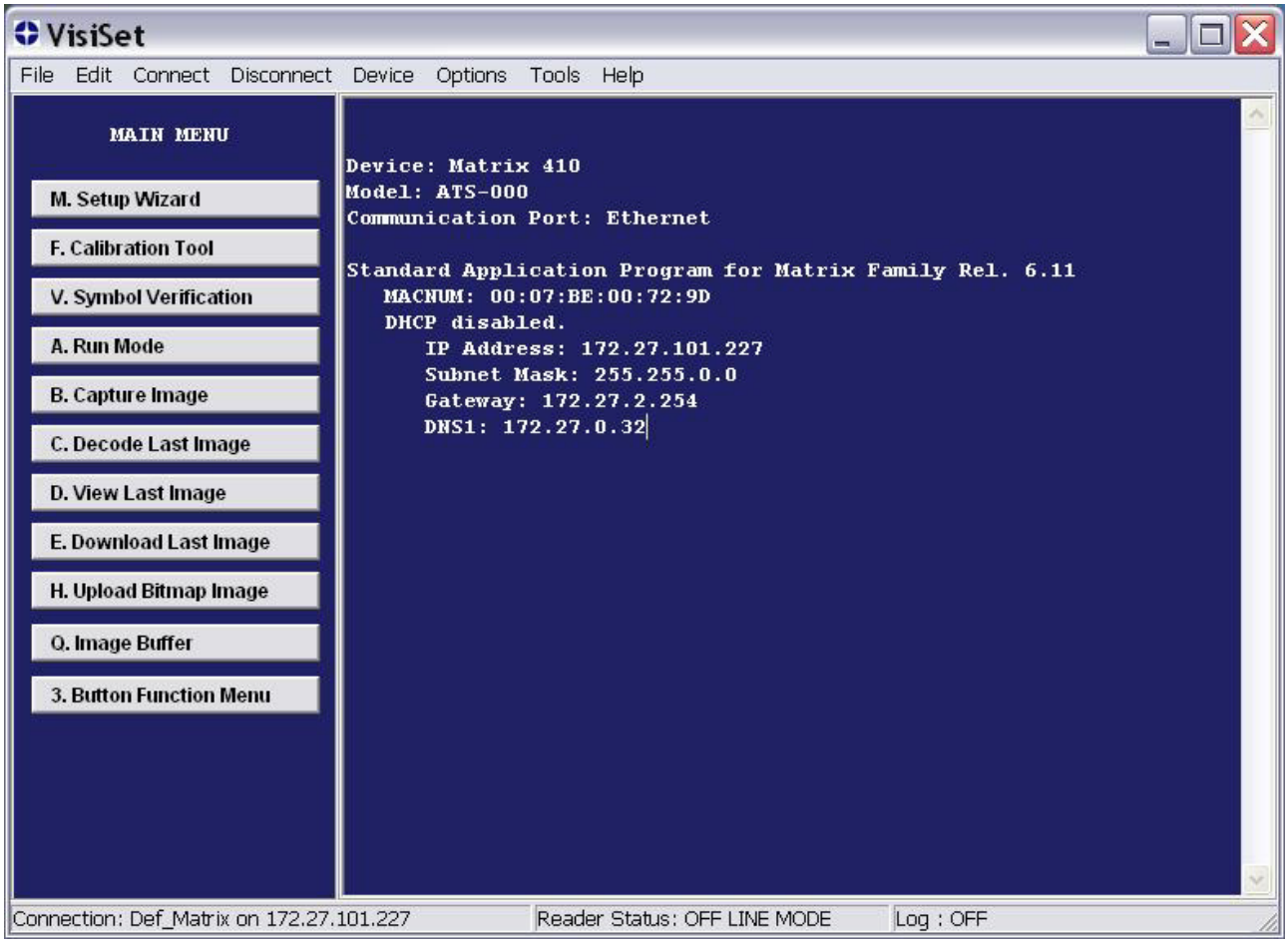


- then click on "**Connect**":

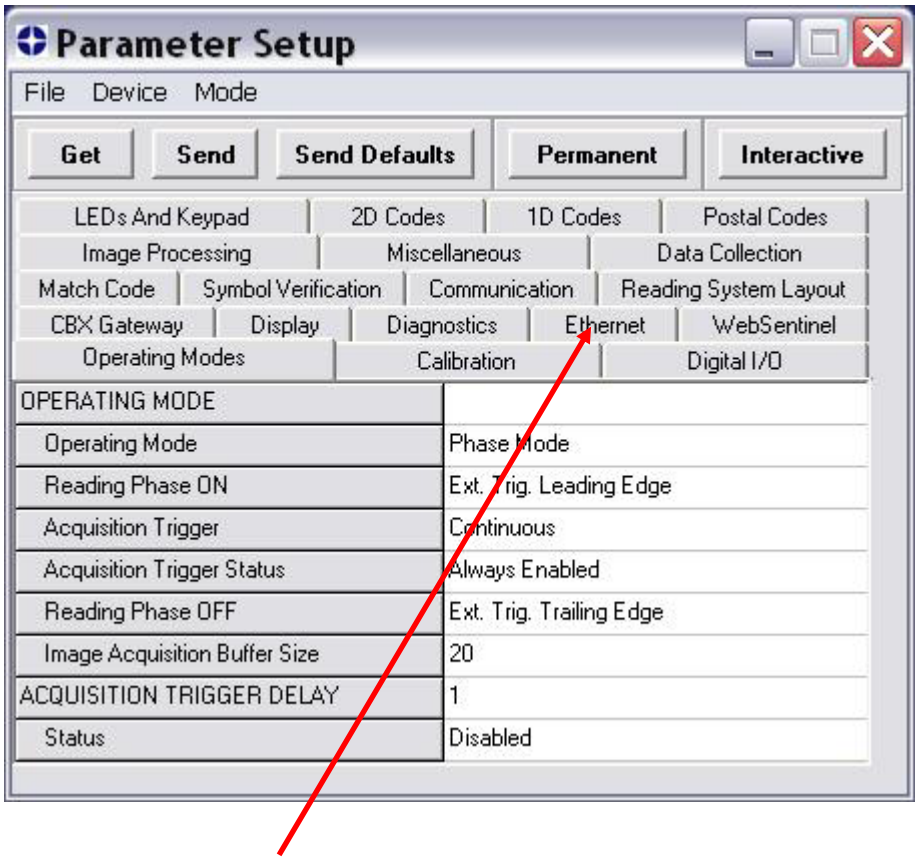


Visiset connects the device and the "**Welcome**" window appears,

.



- Upon opening Visiset, click on "**Device**", then "**Get Configuration from Temporary Memory**". The "Parameter Setup" window appears:



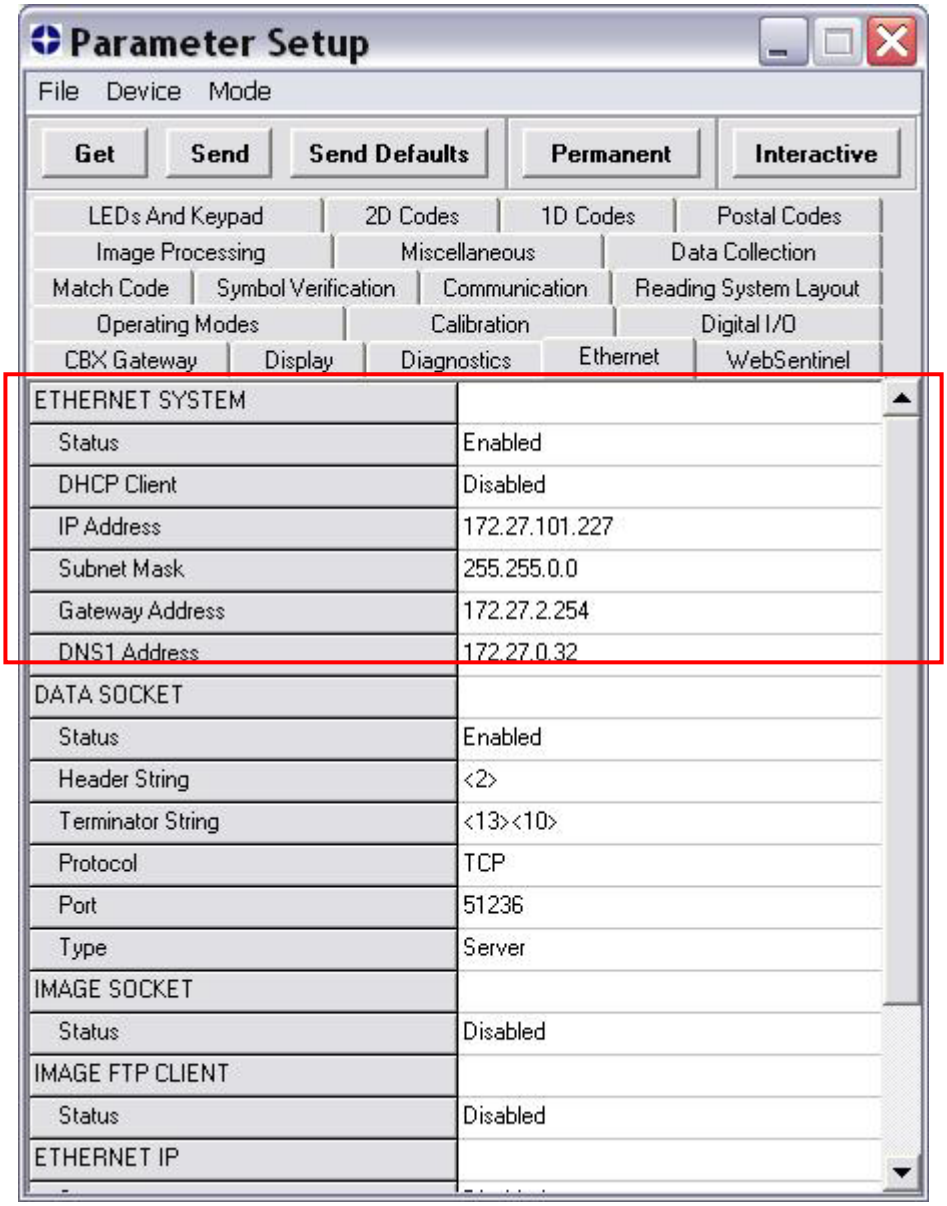
- Select the "**Ethernet**" tab; here check/set the correct "**Ethernet System**" parameters, according your LAN.

Note:

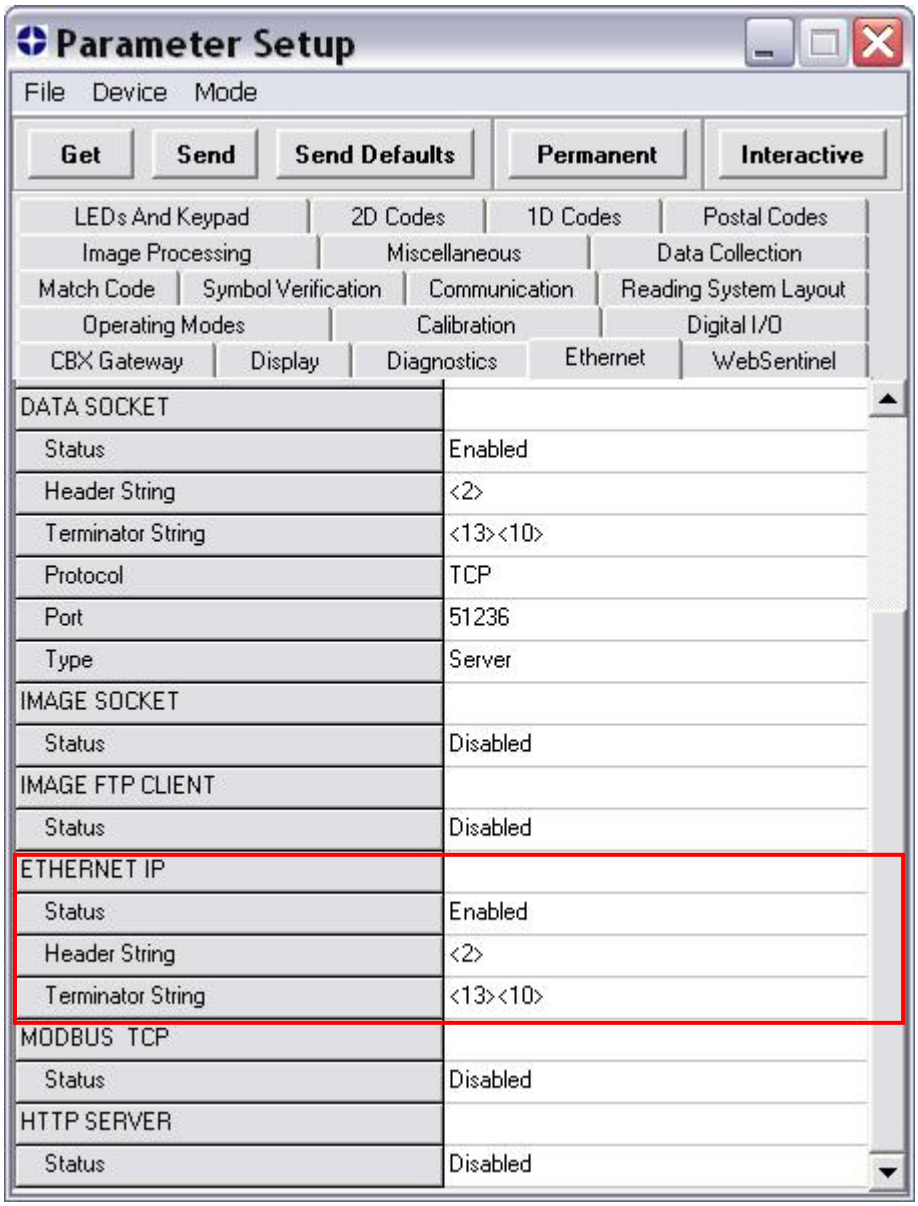
- If the selected connection option is "Ethernet" (like on pictures of pag.36), the device has already properly ethernet-connected then keep the "Ethernet System" parameters unchanged;

- If the selected connection option is "Serial port", the "Ethernet System" parameters have the default values, then the user has to properly set them.

Below an example of a static IP addressing.



- Go to the "**ETHERNET IP**" section, then select the option "**Status = Enabled**": the window below appears



- Select the desired "**Header String**" and "**Terminator String**" parameters to format the EIP string according to the application requirements. The example above formats the string as. <STX>……string……<CR><LF> (2, 13 and 10 are as decimal values)

- To save the parameters to the device, click on "**Send**" button.

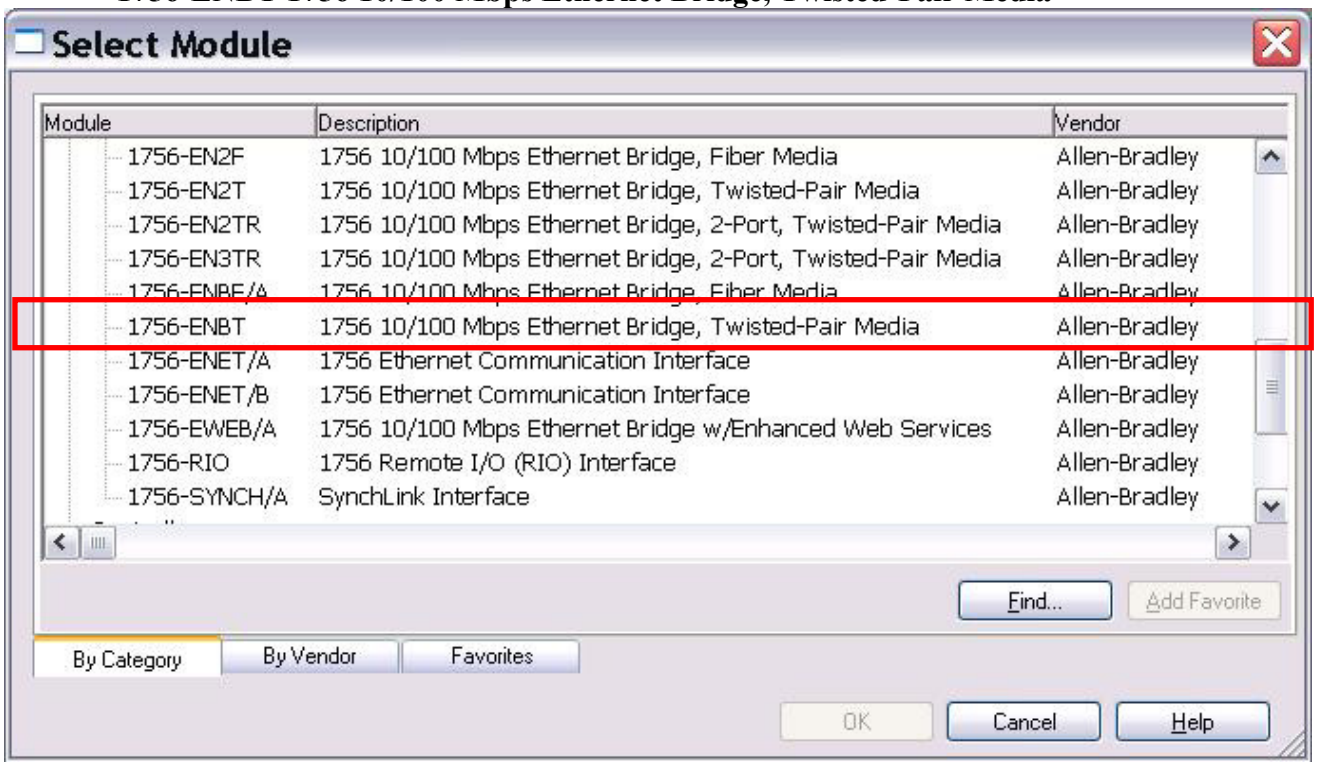
Your Datalogic Matrix is now configured to use EtherNet/IP.

V. Configuring Logix5561™ to use EtherNet/IP

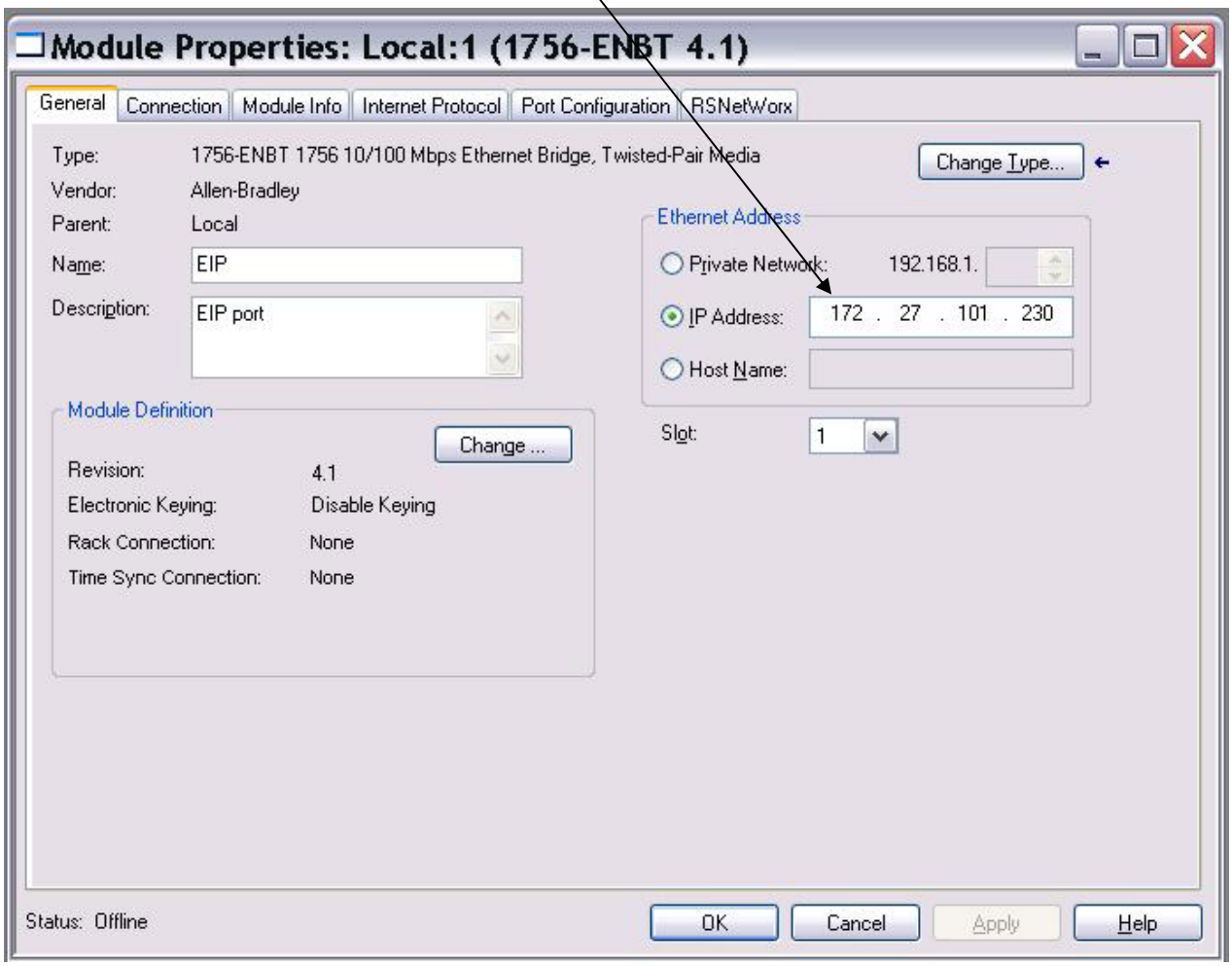
A. Configuring the Ethernet Adapter

- Right click on the I/O Configuration Folder and select "New Module"

- Choose the appropriate Ethernet Module for your application. For this example:
  "1756-ENBT 1756 10/100 Mbps Ethernet Bridge, Twisted-Pair Media"

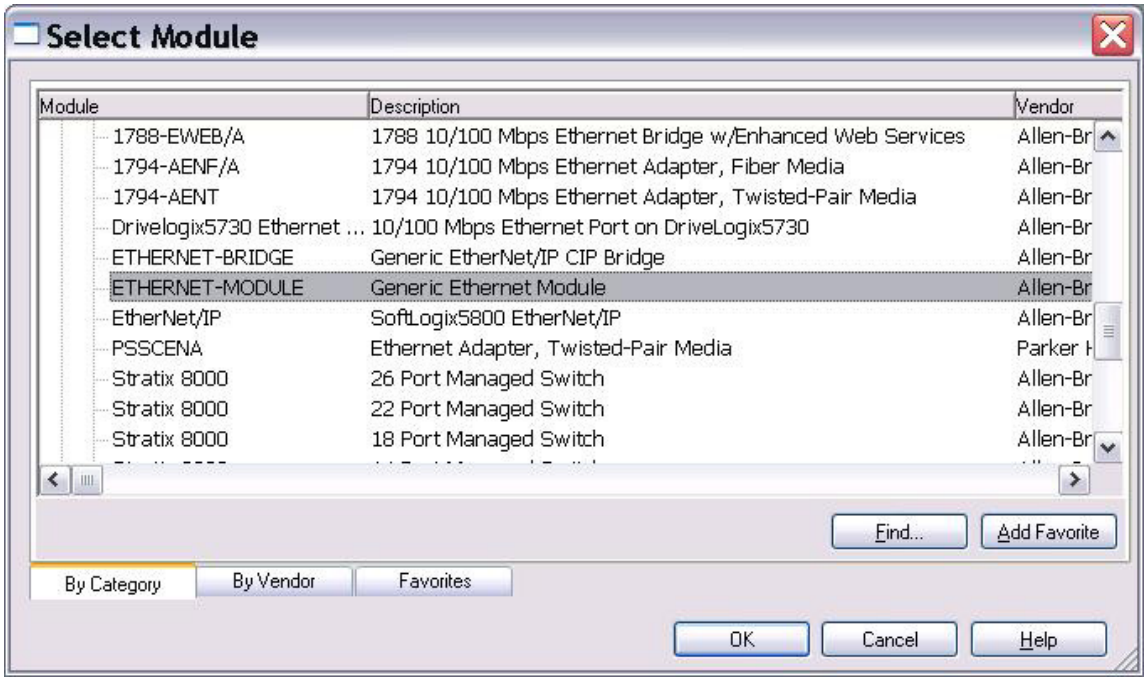


- Fill in the desired IP Address for the Ethernet adapter and assign a name to the adapter. For this example the IP Address is "172.27.101.230" and the name is "EIP".

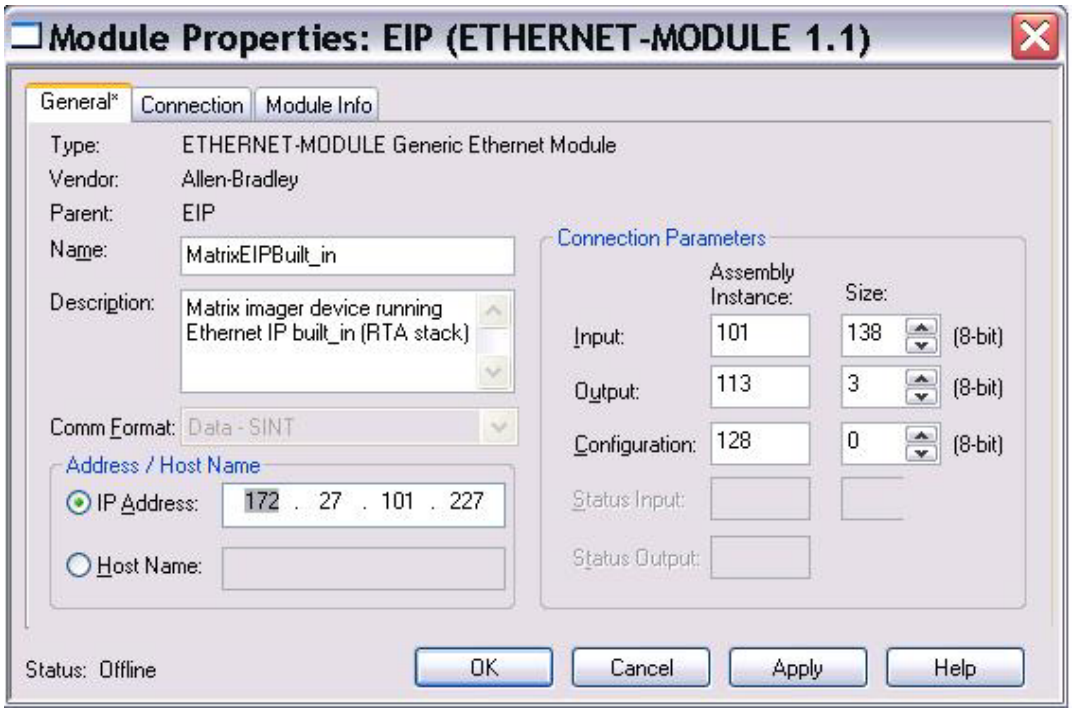


- Click on "OK"

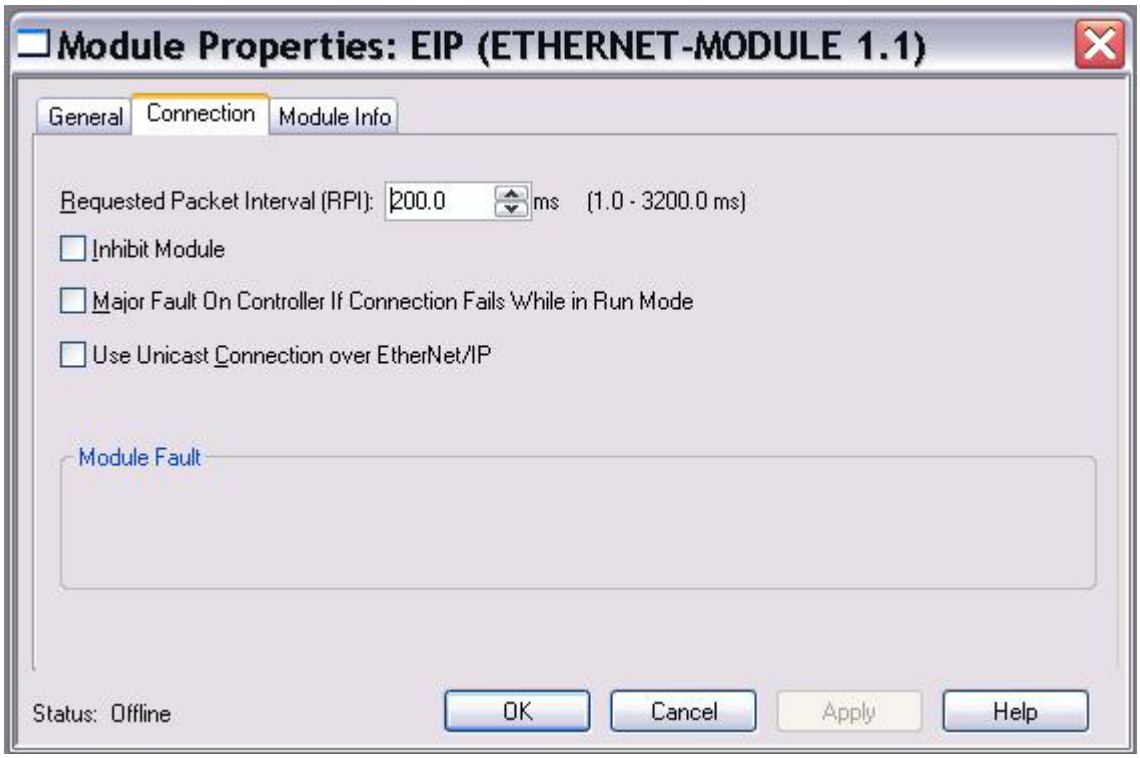
- Right click on the new adapter in the I/O Configuration list and choose **"ETHERNET-MODULE Generic Ethernet Module"**



- Fill in the Connection Parameters and IP Address for the Datalogic Reader and assign a Name. For this example, the IP Address is "172.27.101.227" and the Name is "MatrixEIPBuilt_in". This example configures ControlLogix for access Assembly Instance 0x65 (138 bytes) for inputs and Assembly instance 0x71 (3 bytes) for outputs.



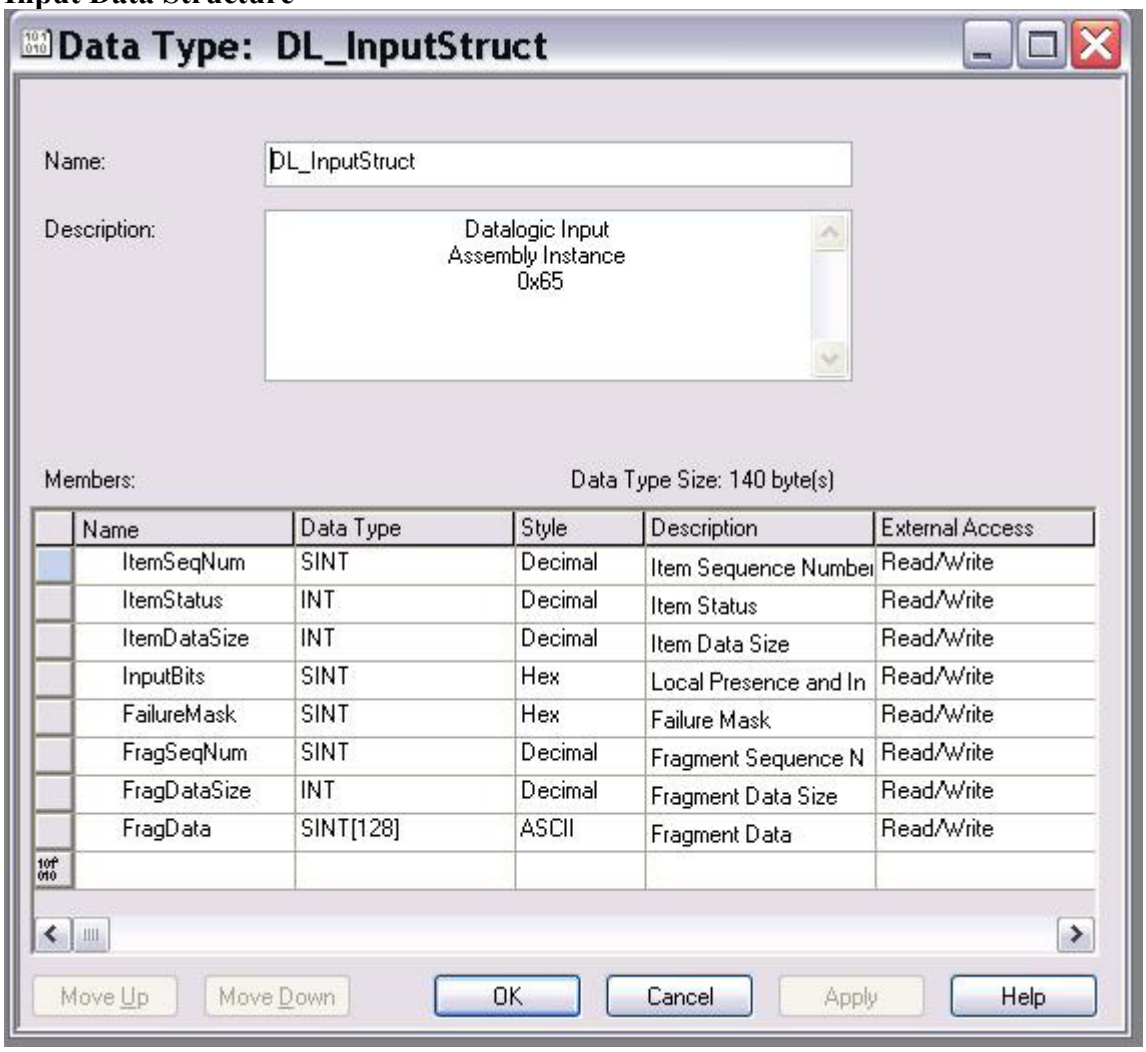
- Select the Request Packet Interval to 200 milliseconds. The range supported by the Datalogic reader is 25 – 3200 milliseconds.



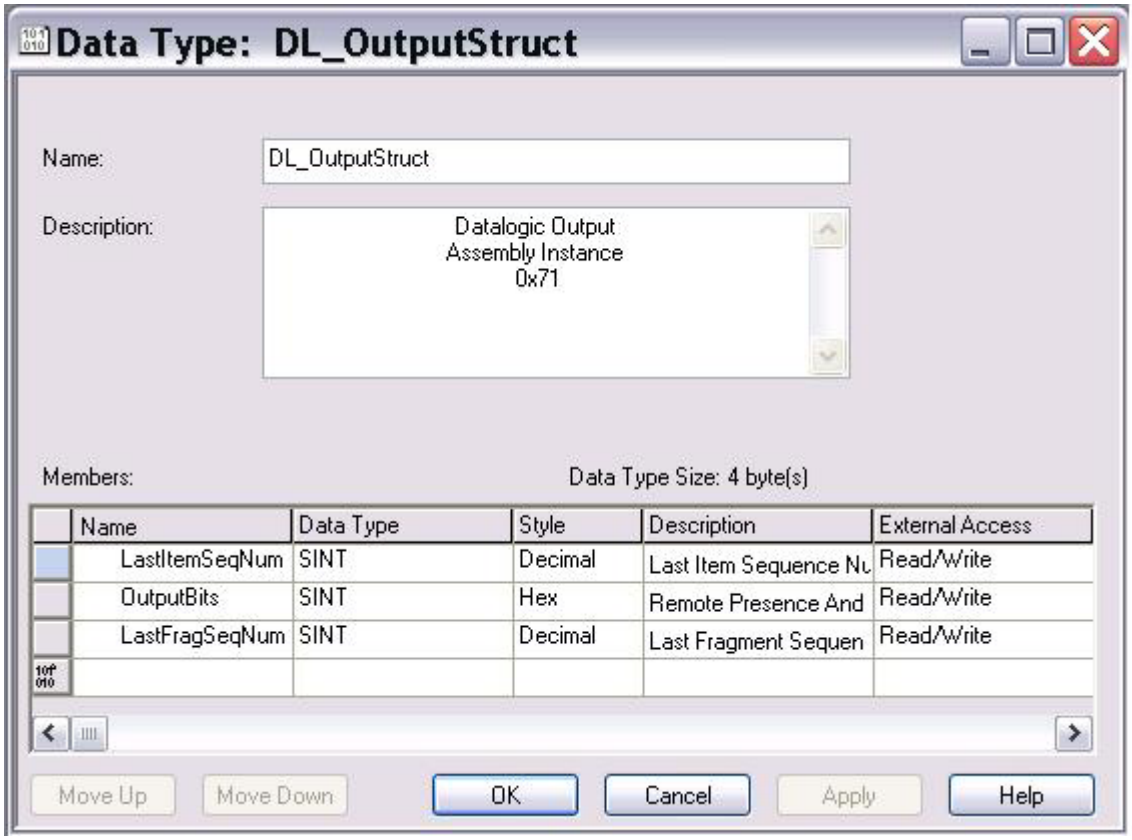
B. Accessing the I/O Data

By default, the Datalogic input data is stored in an array of bytes “DL.I.Data[]” and the Datalogic output data is stored in an array of bytes “DL.O.Data[]”. To store the data in a useful data structures, User-Defined data structures need to be defined.

Input Data Structure

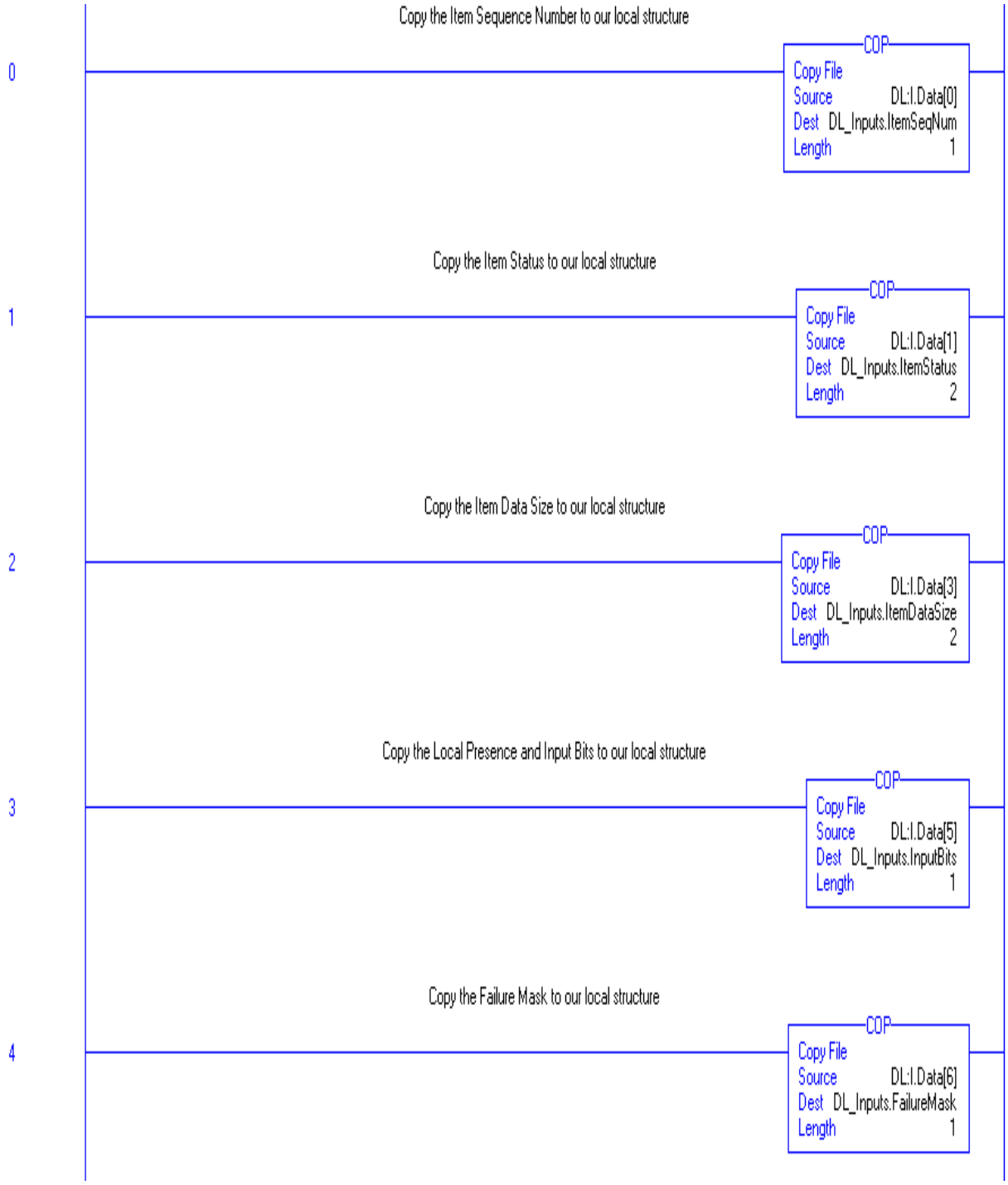


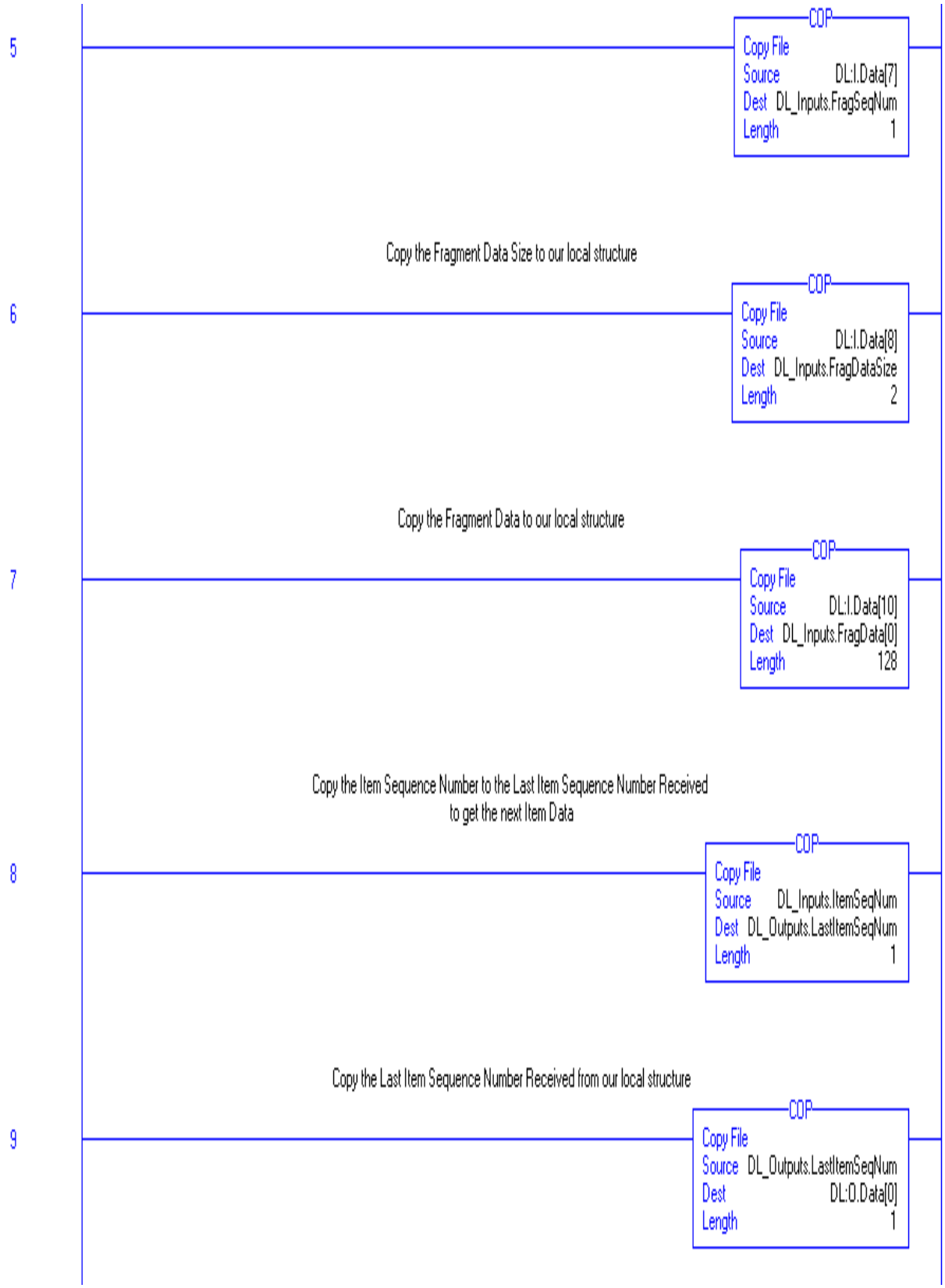
Output Data Structure

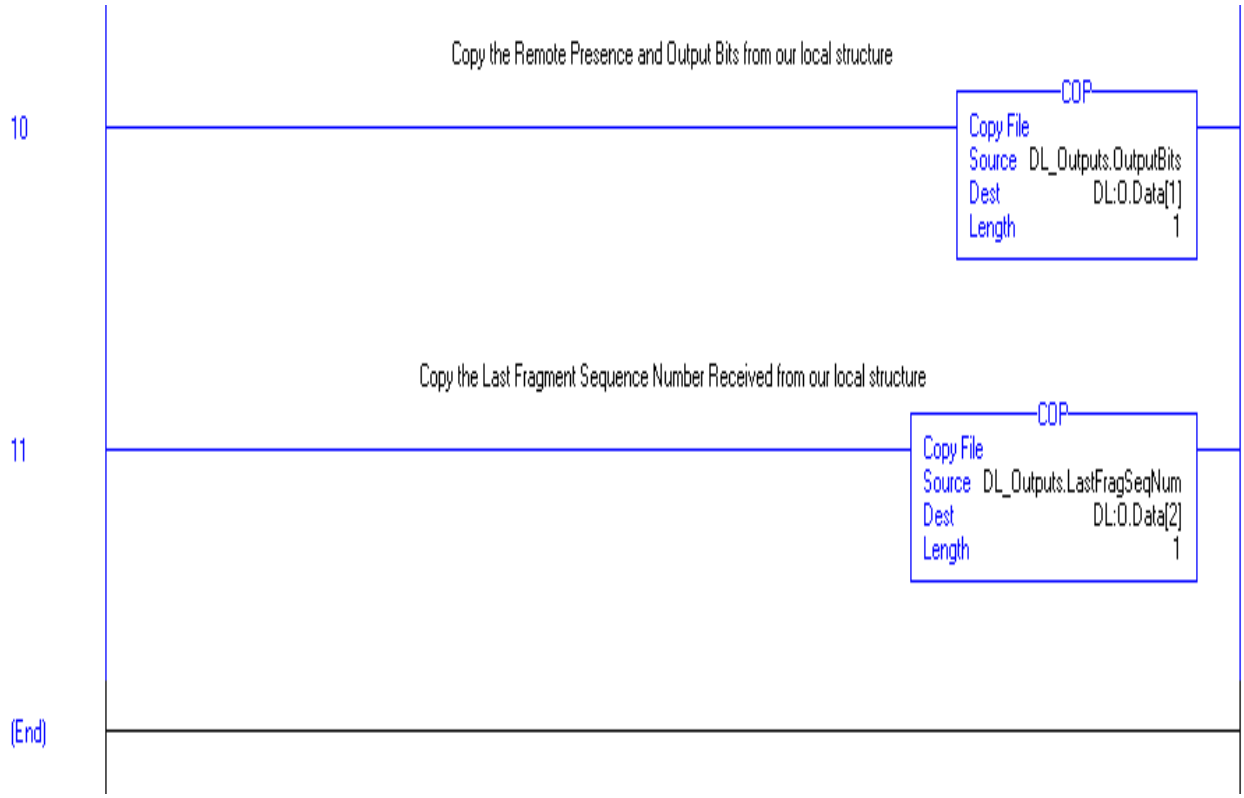


C. Sample Ladder Logic

The sample ladder logic “IO_Sample.ACD” stores all input data into the user-defined structure and handles the handshake required to read barcodes out of the Datalogic Reader.



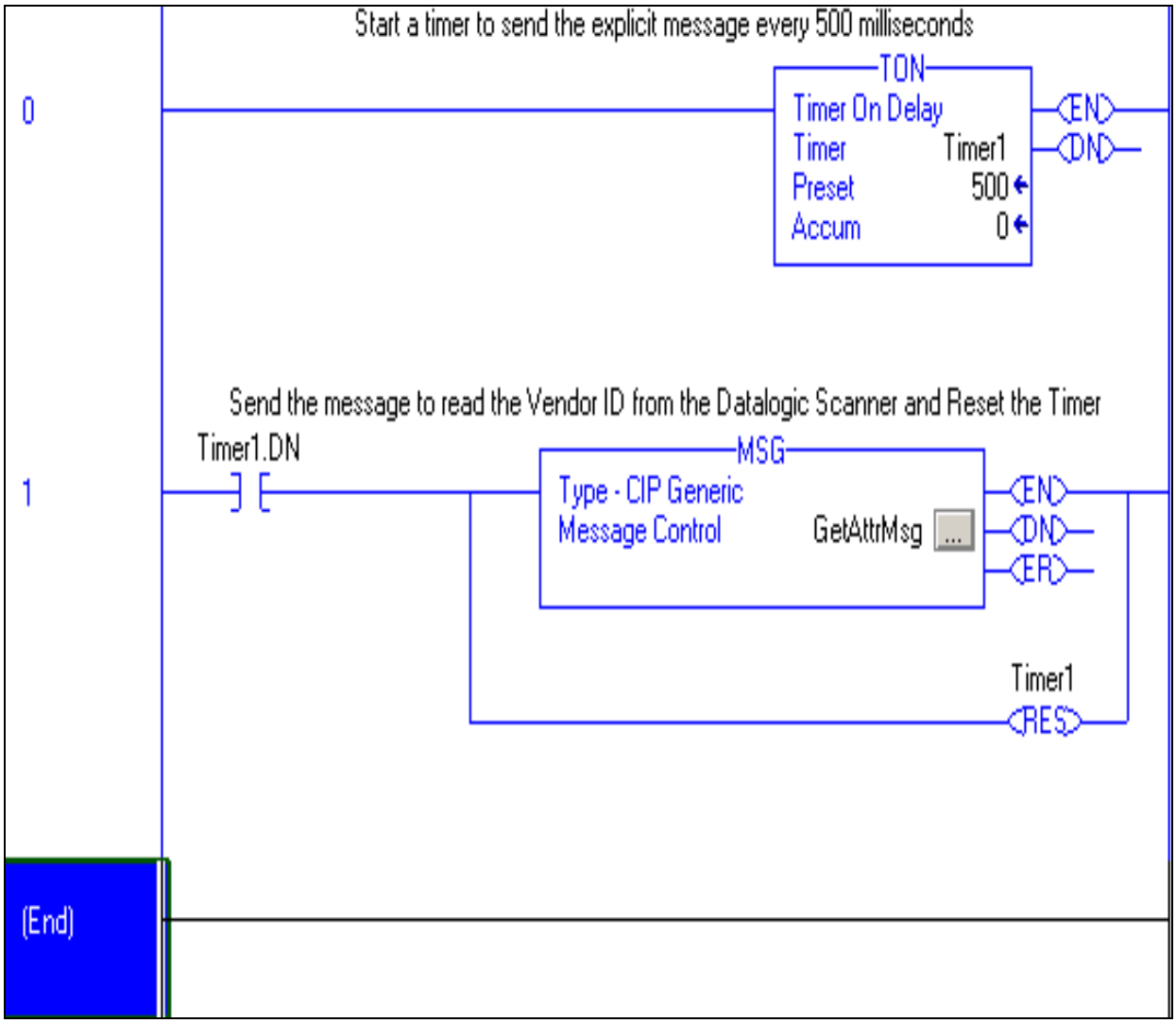




VI. Using Explicit Messaging

A. Sample Ladder Logic

The sample ladder logic “EM_Sample.ACD” reads the Vendor ID from the Datalogic reader every 500 milliseconds.

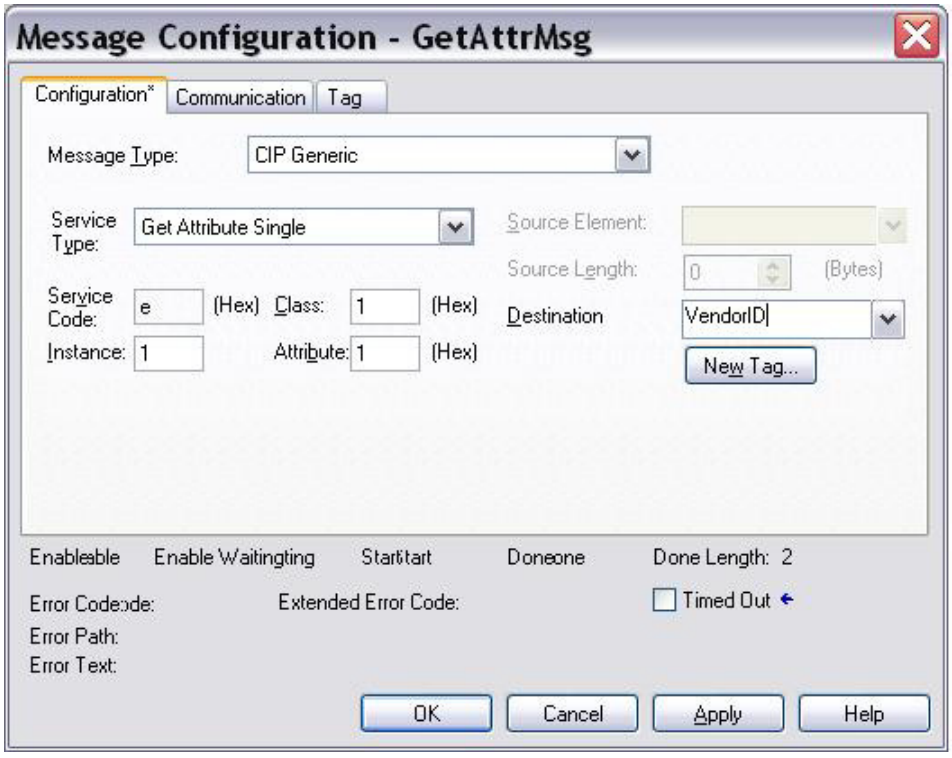


B. Configuring the MSG instruction

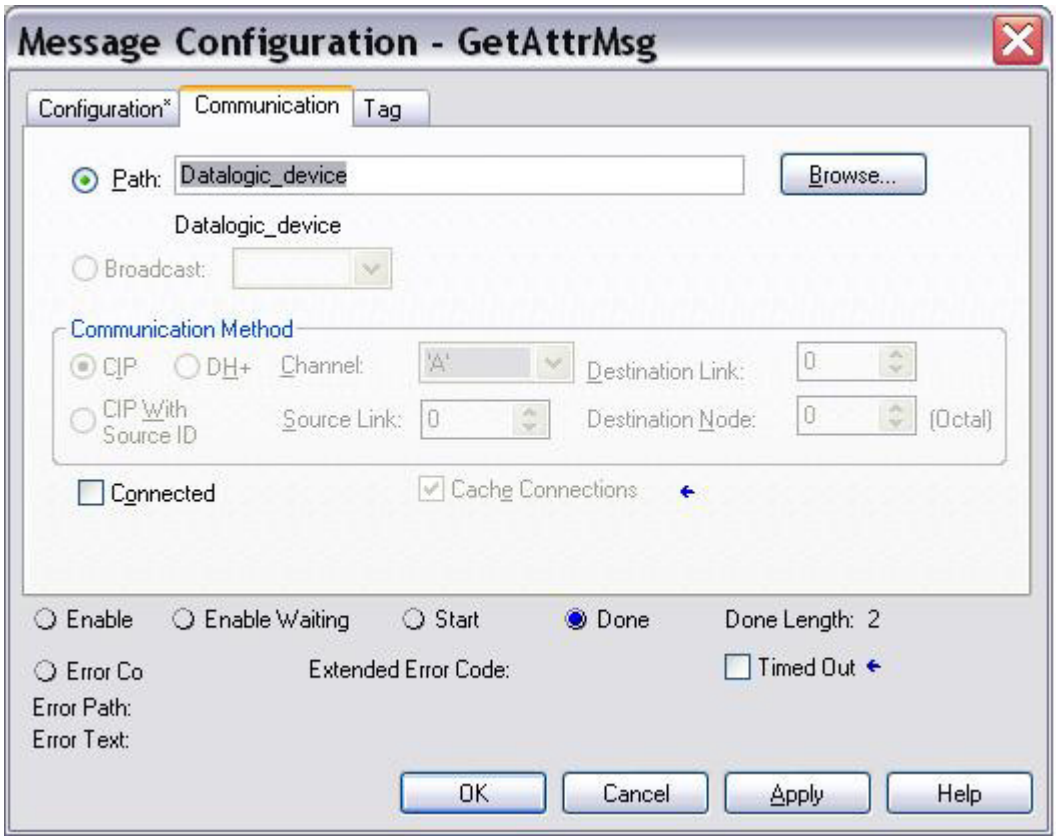
The MSG instruction is used to send explicit messages to node on a given network. This example performs a Get_Attribute_Single to Class 1, Instance 1, Attribute 1 to the Datalogic Reader. The result is stored in a unsigned integer tag called “VendorID”.

To check the MSG configuration settings, select “Controller Tags” on the left column of the sample project, then double-click on it:

Select “GetAttrMsg”, right-click on it then select “Configure GetAttrMsg”:



The communication tab configures the path to “Datalogic_device”. This is the name of the Datalogic Reader device.



VII. Troubleshooting Procedures

The Diagnostic Object supplies a Failure Mask, Failure Subcause, and Failure String for diagnostics. Contact Datalogic for the meaning of the Failure Subcause and Failure String. Datalogic defines the particular causes of the Failure Mask. The Failure Mask has 5 defined values:

- 0x01 – "Input Failure"
- 0x02 – "Communications Failure"
- 0x04 – "Reader Failure"
- 0x08 – "Software Error"
- 0x10 – "Remote Failure"

This following section covers general EtherNet/IP issues. For issues related to Ethernet networking (other than general TCP/IP configuration of the Datalogic reader), contact your Information Technology (IT) department. For issues related to the Matrix reader, contact Datalogic.

Problem:	TCP Connect / Ping Failure
Possible Causes:	IP Address Incorrect Subnet Mask Incorrect Gateway Address Incorrect
Solution:	Using Visiset, verify the Ethernet configuration. Test the configuration via a ping to the device.

Problem:	I/O Connection Failed
Possible Causes:	Connection configuration incorrect Connection is already allocated
Solution:	Make sure the connection isn't already allocated (see error code section). Verify the path and size is correct for both the inputs and outputs.

Problem:	I/O Connection times out
Possible Causes:	Multicast Traffic not routed properly Requested Packet Interval (RPI) set too fast
Solution:	Make sure the RPI is greater than 25 milliseconds. Make sure Multicast traffic is routed properly.

Problem:	Barcode doesn't update
Possible Causes:	Handshaking protocol isn't working
Solution:	Make sure the Last Item Sequence Number Received is equal to the Item Sequence Number. Make sure trigger is working properly using the EtherNet/IP Reader Demo.

Appendix A – EtherNet/IP Error Codes

A. General Status Codes

(The following is from Volume 1, Appendix B of the ODVA CIP Specification.)

The following table lists the Status Codes that may be present in the General Status Code field of an Error Response message. Note that the Extended Code Field is available for use in further describing any General Status Code. Extended Status Codes are unique to each General Status Code within each object. Each object shall manage the extended status values and value ranges (including vendor specific). All extended status values are reserved unless otherwise indicated within the object definition.

General Status Code (in hex)	Status Name	Description of Status
00	Success	Service was successfully performed by the object specified.
01	Connection failure	A connection related service failed along the connection path.
02	Resource unavailable	Resources needed for the object to perform the requested service were unavailable
03	Invalid parameter value	See Status Code 0x20, which is the preferred value to use for this condition.
04	Path segment error	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
05	Path destination unknown	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered.
06	Partial transfer	Only part of the expected data was transferred.
07	Connection lost	The messaging connection was lost.
08	Service not supported	The requested service was not implemented or was not defined for this Object Class/Instance.
09	Invalid attribute value	Invalid attribute data detected
0A	Attribute list error	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
0B	Already in requested mode/state	The object is already in the mode/state being requested by the service
0C	Object state conflict	The object cannot perform the requested service in its current mode/state
0D	Object already exists	The requested instance of object to be created already exists.
0E	Attribute not settable	A request to modify a non-modifiable attribute was received.
0F	Privilege violation	A permission/privilege check failed
10	Device state conflict	The device's current mode/state prohibits the execution of the requested service.
11	Reply data too large	The data to be transmitted in the response buffer is larger than the allocated response buffer
12	Fragmentation of a primitive value	The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type.
13	Not enough data	The service did not supply enough data to perform the specified operation.
14	Attribute not supported	The attribute specified in the request is not supported
15	Too much data	The service supplied more data than was expected
16	Object does not exist	The object specified does not exist in the device.
17	Service fragmentation sequence not in progress	The fragmentation sequence for this service is not currently active for this data.
18	No stored attribute data	The attribute data of this object was not saved prior to the requested service.

General Status Code (in hex)	Status Name	Description of Status
19	Store operation failure	The attribute data of this object was not saved due to a failure during the attempt.
1A	Routing failure, request packet too large	The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service.
1B	Routing failure, response packet too large	The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service.
1C	Missing attribute list entry data	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
1D	Invalid attribute value list	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
1E	Embedded service error	An embedded service resulted in an error.
1F	Vendor specific error	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
20	Invalid parameter	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.
21	Write-once value or medium already written	An attempt was made to write to a write-once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
22	Invalid Reply Received	An invalid reply is received (e.g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.
23 – 24		Reserved by CIP for future extensions
25	Key Failure in path	The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
26	Path Size Invalid	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
27	Unexpected attribute in list	An attempt was made to set an attribute that is not able to be set at this time.
28	Invalid Member ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute
29	Member not settable	A request to modify a non-modifiable member was received
2A	Group 2 only server general failure	This error code may only be reported by DeviceNet group 2 only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
2B – CF		Reserved by CIP for future extensions
D0 – FF	Reserved for Object Class and service errors	This range of error codes is to be used to indicate Object Class specific errors. Use of this range should only be performed when none of the Error Codes presented in this table accurately reflect the error that was encountered.

B. Forward Open (Connection Allocation) Error Codes

(The following is from Volume 1, Chapter 3, Section 3-5.6.1 of the ODVA CIP Specification.)

The following error codes are returned with the reply to a Connection Manager Service Request that resulted in an error. These error codes shall be used to help diagnose the problem with a Service Request. The error code shall be split into an 8 bit general status and one or more 16-bit words of extended status. Unless specified otherwise, only the first word of extended status shall be required.

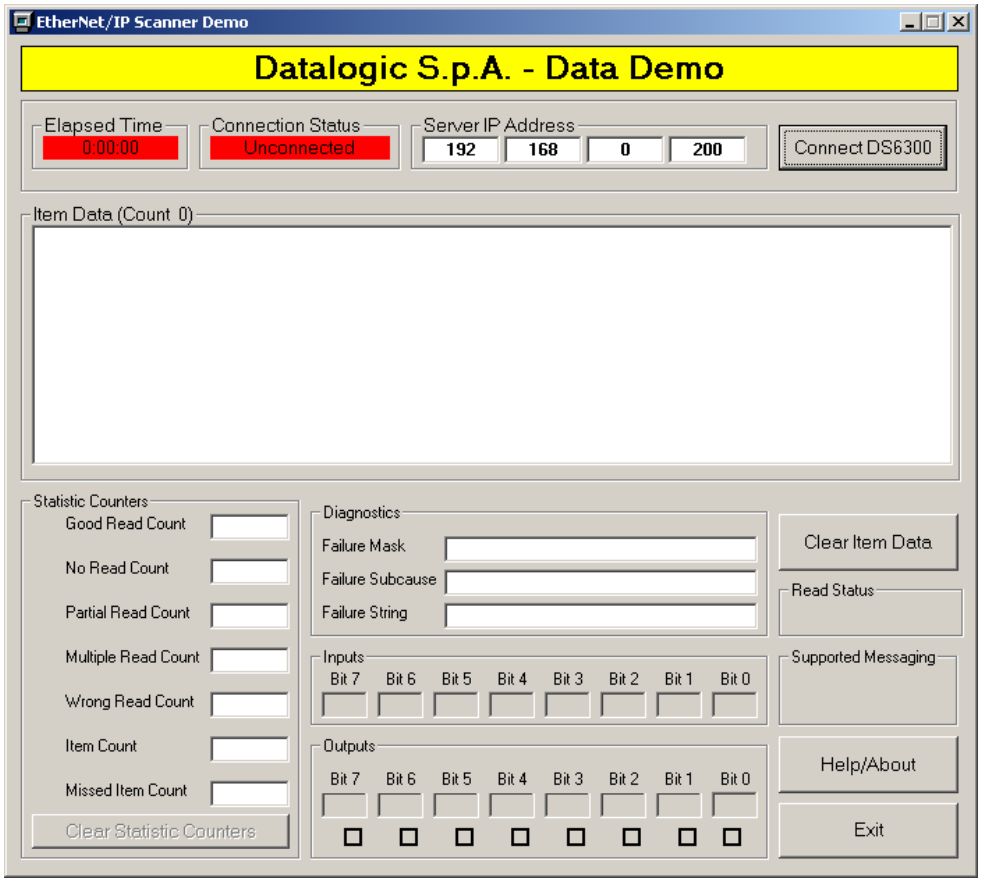
General Status	Extended Status	Explanation
0x00		Service completed successfully.
0x01	0x0100	Connection in Use or Duplicate Forward Open.
0x01	0x0103	Transport Class and Trigger combination not supported
0x01	0x0106	Ownership Conflict
0x01	0x0107	Connection not found at target application.
0x01	0x0108	Invalid Connection Type. Indicates a problem with either the Connection Type or Priority of the Connection.
0x01	0x0109	Invalid Connection Size
0x01	0x0110	Device not configured
0x01	0x0111	RPI not supported. May also indicate problem with connection time-out multiplier, or production inhibit time.
0x01	0x0113	Connection Manager cannot support any more connections
0x01	0x0114	Either the Vendor Id or the Product Code in the key segment did not match the device
0x01	0x0115	Product Type in the key segment did not match the device
0x01	0x0116	Major or Minor Revision information in the key segment did not match the device
0x01	0x0117	Invalid Connection Point
0x01	0x0118	Invalid Configuration Format
0x01	0x0119	Connection request fails since there is no controlling connection currently open.
0x01	0x011A	Target Application cannot support any more connections
0x01	0x011B	RPI is smaller than the Production Inhibit Time.
0x01	0x0203	Connection cannot be closed since the connection has timed out
0x01	0x0204	Unconnected Send timed out waiting for a response.
0x01	0x0205	Parameter Error in Unconnected Send Service
0x01	0x0206	Message too large for Unconnected message service
0x01	0x0207	Unconnected acknowledge without reply
0x01	0x0301	No buffer memory available
0x01	0x0302	Network Bandwidth not available for data
0x01	0x0303	No Tag filters available
0x01	0x0304	Not Configured to send real-time data
0x01	0x0311	Port specified in Port Segment Not Available
0x01	0x0312	Link Address specified in Port Segment Not Available
0x01	0x0315	Invalid Segment Type or Segment Value in Path
0x01	0x0316	Path and Connection not equal in close
0x01	0x0317	Either Segment not present or Encoded Value in Network Segment is invalid.
0x01	0x0318	Link Address to Self Invalid
0x01	0x0319	Resources on Secondary Unavailable
0x01	0x031A	Connection already established
0x01	0x031B	Direct connection already established
0x01	0x031C	Miscellaneous
0x01	0x031D	Redundant connection mismatch
0x01	0x031E	No more consumer resources available in the producing module
0x01	0x031F	No connection resources exist for target path
0x01	0x320 –0x7FF	Vendor specific

Appendix B – EtherNet/IP Scanner Demo

1. Overview

The "Ethernet/IP Scanner Demo" simulates some basic functions of the Master EIP and tests the following:

- Item Data
- Statistic Counters
- Diagnostics (Failure Mask, Failure Sub Cause, Failure String)
- Discrete Outputs
- Supported Messaging (I/O and/or Explicit)

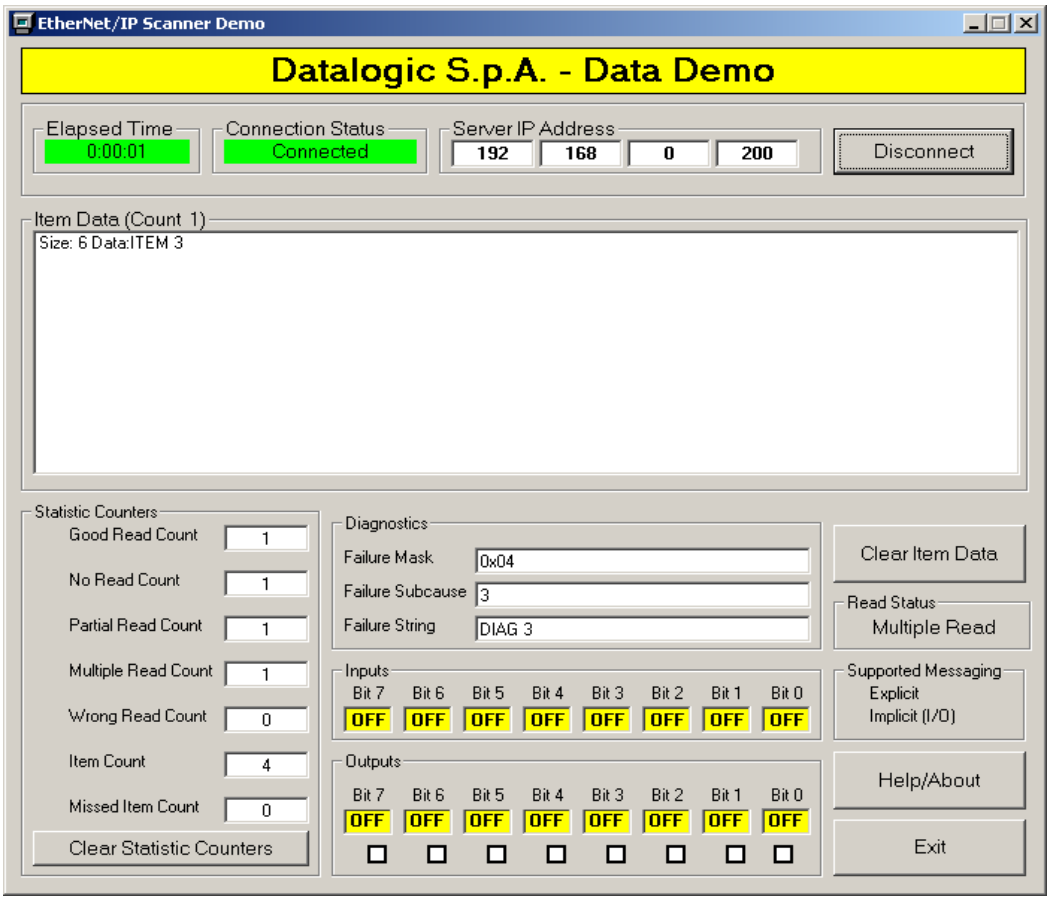


If EIP enabled on Matrix, click on the "Connect…."[1] button to start the communication with the device

[1] The connection button has the "**Connect DS6300**" label because it refers the first Datalogic reader supporting the Ethernet/IP implementation. No trouble at all connecting all the others EIP devices

2. Successful Communications

If all communications are successful, the screen should be similar to the following.



The data strings coming from the reader are shown on the “Item Data” window, with data size and content. The example above shows the received string “ITEM 3”, 6 bytes length.

3. Matrix triggering through Ethernet/IP

Matrix readers allow to start the reading phase through Ethernet/IP, running the Phase Mode or the One Shot operating mode.

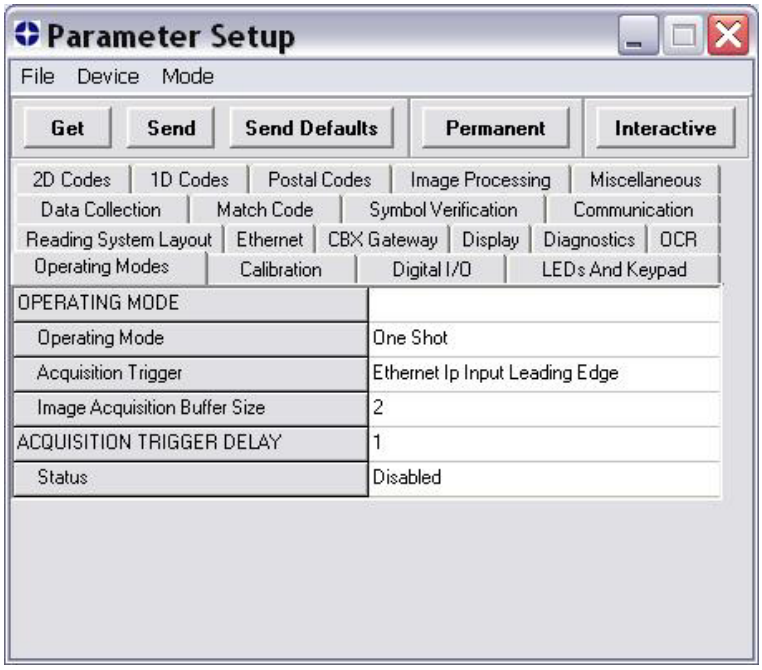
In order to use this feature the following steps are necessary:

1. open the Matrix Parameter Setup
2. enable Ethernet/IP

ONE SHOT

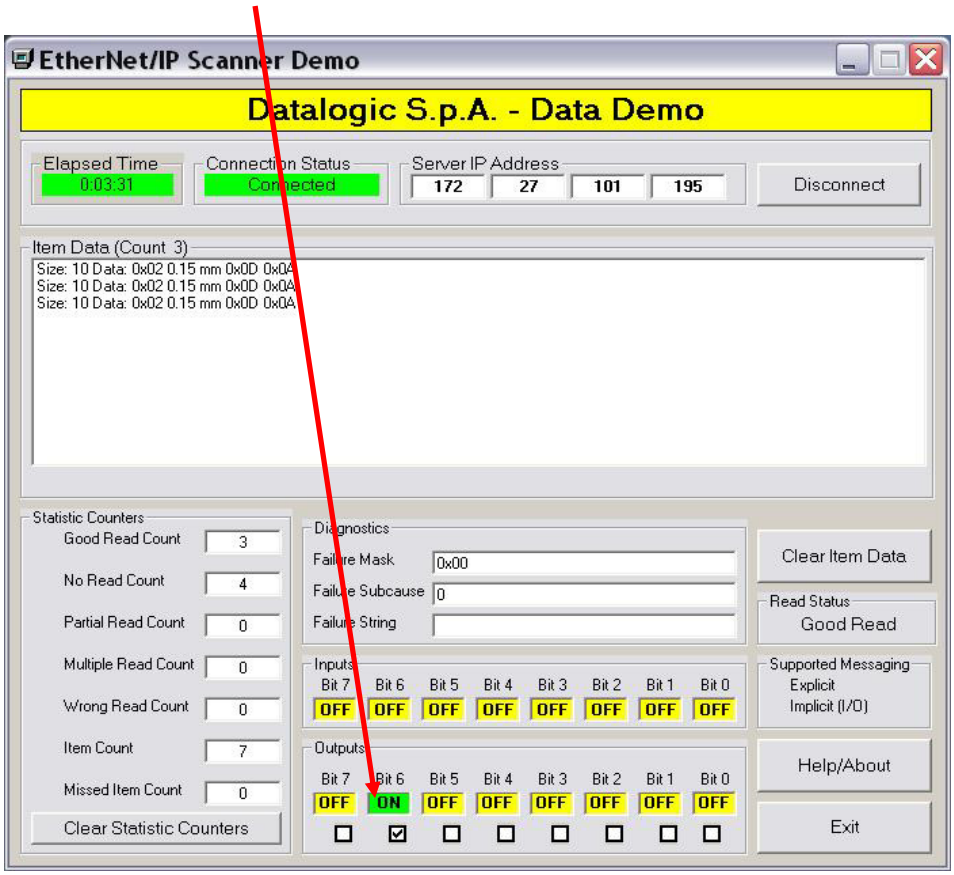
3. select the "Operating Modes" tab then set
    a. "Operating Mode = One Shot"
    b. "Acquisition Trigger = Ethernet IP Input Leading Edge"

(the picture below shows the setup)



4. click on "Send" to save the configuration and run the device
5. launch the Ethernet/IP Scanner Demo and verify the good connection

6. click on the "**Bit 6**" box of the "Outputs" area: the box toggles to the "ON" status, the reading phase runs and the data string comes to the "Item Data" window



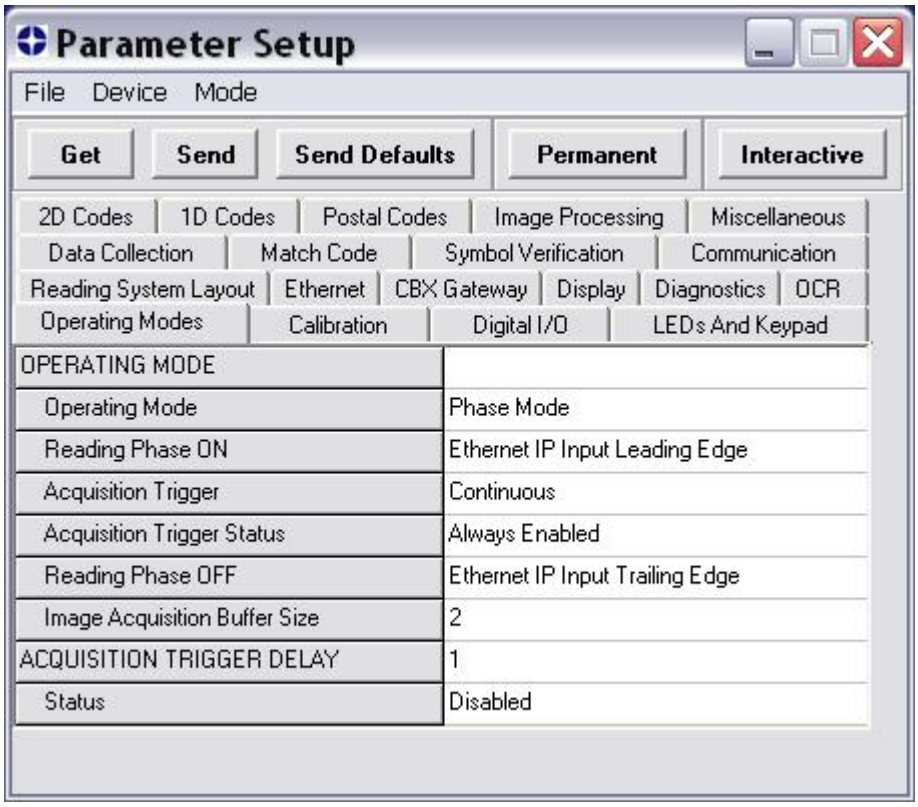
7. click on the "Bit 6" box again to come back to the "OFF" status
8. repeat the steps 6 and 7 to read again. The picture above shows 3 data strings, 10 bytes long: <02hex>0.15 mm<0Dhex><0Ahex>

Note that:
IF the step 3b is
"Acquisition Trigger = Ethernet IP Input Leading Edge"
THEN
the "bit 6 ON→OFF change" triggers the Matrix
ELSE
IF the step 3b is
"Acquisition Trigger = Ethernet IP Input Trailing Edge"
THEN
the "bit 6 OFF→ON change" triggers the Matrix

PHASE MODE

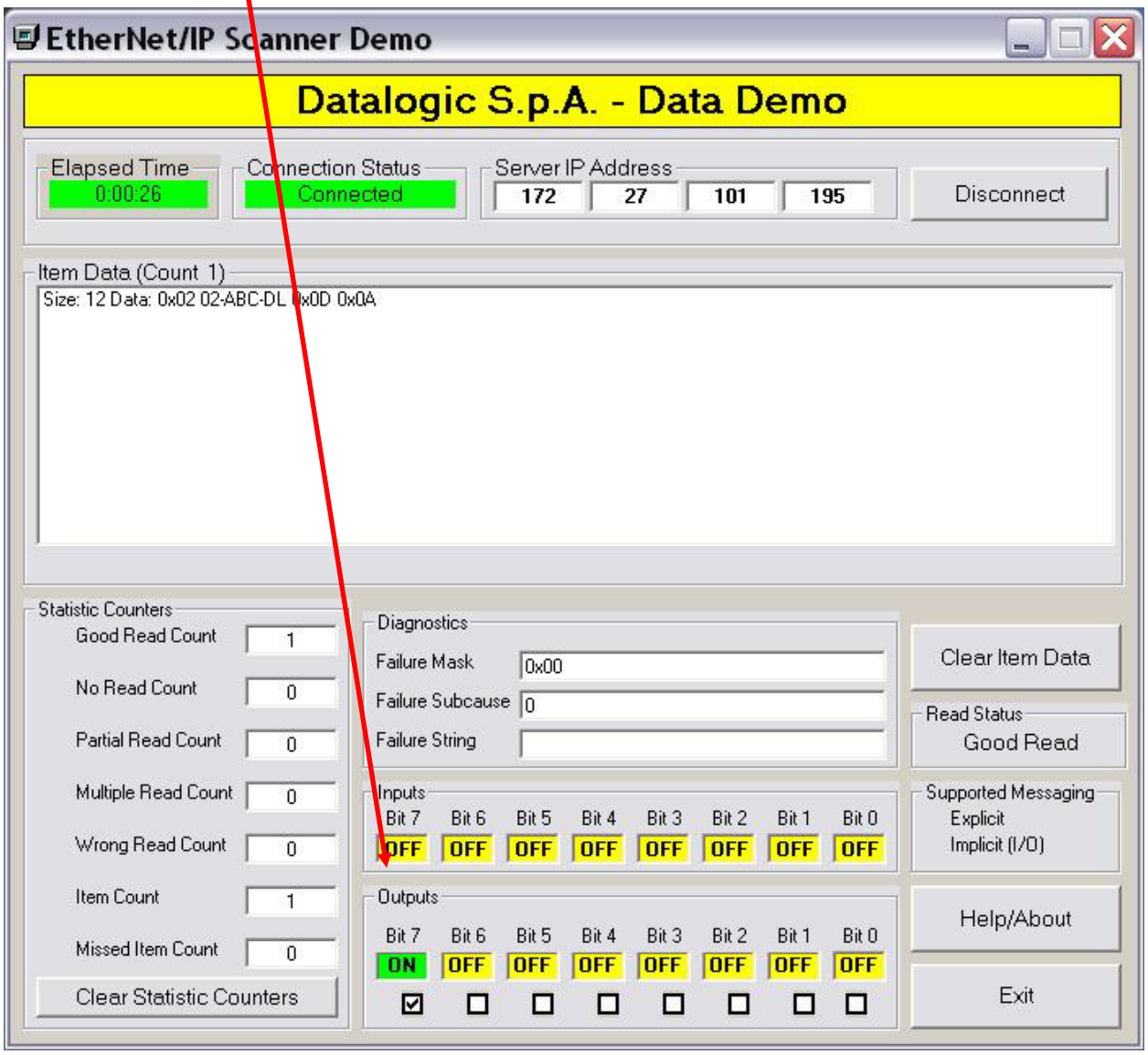
3. select the “Operating Modes” tab then set
    a. “Operating Mode = Phase Mode”
    b. “Reading Phase ON = Ethernet IP Input Leading Edge”
    c. “Reading Phase OFF = Ethernet IP Input Trailing Edge”

(the picture below shows the setup)



4. click on “Send” to save the configuration
5. launch the Ethernet/IP Scanner Demo and verify the good connection

6. click on the "**Bit 7**" box of the "Outputs" area: the box toggles to the "ON" status, the reading phase starts



7. click on the "**Bit 7**" box again: the box toggles to the "OFF" status, the reading phase ends.

The data string comes to the "Item Data" window now or on the previous step according to the Operating Modes options.
The picture above shows 3 data strings, 12 bytes long:
<02hex>02-ABC-DL<0Dhex><0Ahex>

Note that:
IF the step 3b and 3c are
"Reading Phase ON = Ethernet IP Input Leading Edge"
"Reading Phase OFF = Ethernet IP Input Trailing Edge"
THEN
the "bit 7 ON→OFF starts the reading phase
the "bit 7 OFF→ON ends the reading phase
ELSE
IF the step 3b and 3c are
"Reading Phase ON = Ethernet IP Input Trailing Edge"
"Reading Phase OFF = Ethernet IP Input Leading Edge"
THEN
the "bit 7 OFF→ON starts the reading phase
the "bit 7 ON→OFF ends the reading phase

Summarizing:

- **The EIP Master can drive the Matrix reading through Output bits**
  - **The bit 6 controls the "One Shot" Operating Mode**
  - **The bit 7 controls the "Phase Mode" Operating Mode**
- **The "Outputs" area of the EIP Scanner Demo refers the 1st byte of the Output Area of the EIP Master**
- **The "Outputs" boxes of the EIP Scanner Demo refer the bits of the 1st byte of the Output Area of the EIP Master**

4. I/O Connection Failure

If the I/O connection allocation fails, an error message pop up window appears. See the error code section of this document for the cause of the error. The error code in the example indicates the I/O connection is allocated already.