MANUAL

ICDM-RX/MOD

Modbus/TCP

PEPPERL+FUCHS

PEPPERL+FUCHS

Table of Contents

PEPPERL+FUCHS

11/4/20

PEPPERL+FUCHS

PEPPERL+FUCHS

1. Introduction

This manual provides detailed information about the following topics:

- *Programming Interface* on Page 12
- *Configuration Overview* on Page 28
- *Serial Menus* on Page 38
- *Ethernet Menus* on Page 55
- *Network Menus* on Page 72
- *Diagnostic Menus* on Page 78
- *System Menus* on Page 90

The *ICDM-RX Hardware Installation and Configuration Manual* provides the following information:

- Connecting the hardware and devices
- Programming the ICDM-RX/MOD IP address,
- Uploading Modbus/TCP firmware

The *ICDM-RX/MOD Modbus/TCP Quick Start* provides embedded web page configuration procedures if you have *read-only or read/write* devices, which provides procedures for your devices.

1.1. Audience

The primary audience of this document is the person responsible for installing the ICDM-RX/MOD and programming the PLC. This manual assumes you are familiar with the following topics:

- Windows operating system
- Modbus/TCP
- Modbus/TCP, Modbus/RTU, and/or Modbus/ASCII
- A PLC, SCADA System, or OPC Server that communicates with Modbus/TCP, Modbus/RTU, or Modbus/ASCII
    - Raw/ASCII devices such as barcode scanners, weigh scales, and printers
    - Modbus/RTU and/or Modbus/ASCII slave devices

1.2. Locating the Latest Software and Documents

Download software or documentation from: **https://www.pepperl-fuchs.com**.

PEPPERL+FUCHS

1.3. Pepperl+Fuchs Modbus Solutions

If you ordered ICDM-RX/MOD, Modbus Router is loaded on the ICDM-RX/MOD by default. You may want to review our other Modbus solutions to make sure that the feature rich Modbus/TCP application is what you need to use before uploading the Modbus/TCP firmware. Optionally, Modbus Router or Modbus Server may by more effective for your particular environment.

You can contact the Pepperl+Fuchs technical support team for information about the differences between the three Modbus solutions and download the Modbus/TCP firmware.

1.4. Product Overview

The ICDM-RX/MOD has Modbus Router firmware loaded at the factory by default. You must upload the Modbus/TCP firmware. Contact Pepperl+Fuchs for the latest version of the firmware.

The ICDM-RX/MOD operates as a highly versatile Modbus gateway when the Modbus/TCP firmware is uploaded to the ICDM-RX/MOD. The ICDM-RX/MOD provides Modbus/TCP, Modbus/RTU, Modbus/ASCII, and Ethernet TCP/IP controller interfaces to both serial and Ethernet TCP/IP raw/ASCII devices, and both Modbus/RTU and Modbus/ASCII slave devices.

- ICDM-RX/MOD 1-port models provide Modbus/TCP support for one raw/ASCII or Modbus/RTU serial device and one raw/ASCII Ethernet device for a total of two devices.
- ICDM-RX/MOD 2-port models provide Modbus/TCP support for two raw/ASCII or Modbus/RTU serial device and two raw/ASCII Ethernet device for a total of four devices.
- The ICDM-RX/MOD 4-port provides Modbus/TCP support for four raw/ASCII or Modbus/RTU serial devices and four raw/ASCII Ethernet devices for a total of eight devices.

You can connect any application, such as a configuration, database, or control application, via the application socket port to raw/ASCII serial and/or Ethernet devices while the device(s) are attached to the PLC via Modbus/TCP.

1.5. Modbus/TCP Multi-Mode Connectivity

The Modbus/TCP firmware 7.x supports the following Modbus/TCP communication modes:

- *PLC Master/ICDM-RX/MOD Slave Mode on Page 8*
- *PLC Slave/ICDM-RX/MOD Master Mode on Page 8*
- *Dual Master (Virtual Peer-to-Peer) - Write Mode on Page 9*
- *Dual Master (Virtual Peer-to-Peer) - Read Mode (Dual Polling) on Page 9*
- *Filtering and Data Extraction Functionality on Page 10*

PEPPERL+FUCHS

1.5.1. PLC Master/ICDM-RX/MOD Slave Mode

PLC Master/ICDM-RX/MOD Slave mode:

- Standard Modbus master to slave device method of communication. All read and write messages are initiated by the Modbus master.

- Raw/ASCII, Modbus/RTU slave, and Modbus/ASCII slave devices are supported in this mode.

- For raw/ASCII mode, the Receive Transfer mode and Transmit Transfer mode are both set to Slave (In Modbus/RTU-to-Slaves and Modbus/ASCII-to-Slaves mode, the ICDM-RX/MOD port only operates in To-Slave mode.
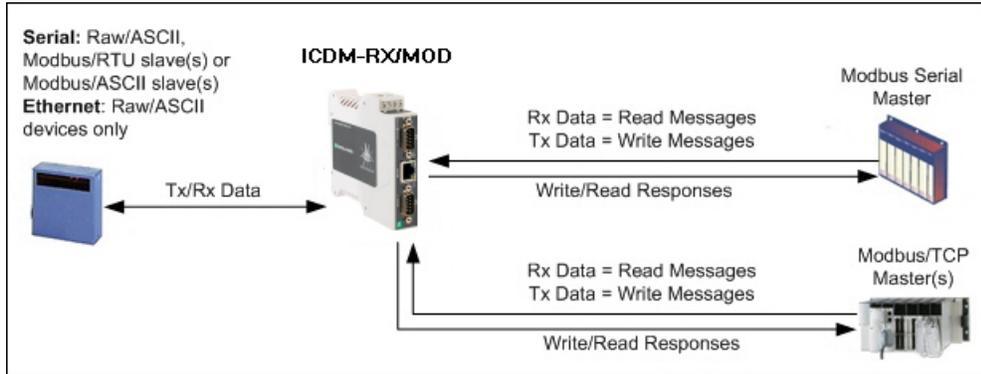
1.5.2. PLC Slave/ICDM-RX/MOD Master Mode

PLC slave/ICDM-RX/MOD master mode:

- The ICDM-RX/MOD initiates all read and write messages.

- The ICDM-RX/MOD writes received serial and/or Ethernet device data directly into PLC memory with minimal latency.

- The ICDM-RX/MOD polls the PLC for transmit data for serial and/or Ethernet devices.

- PLC programs can be simplified to eliminate both polling for received data and sending of write messages to transmit data.

- Only raw/ASCII devices are supported in this mode.

- The ICDM-RX/MOD *Receive Transfer* mode and *Transmit Transfer* mode are both set to *Master*.

11/4/20

PEPPERL+FUCHS
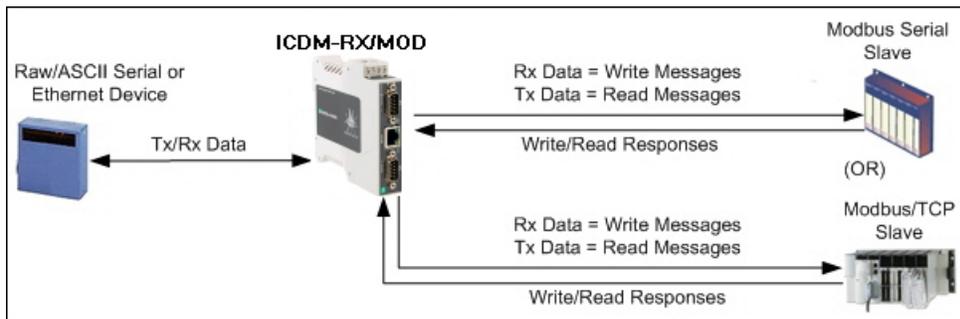
1.5.3. Dual Master (Virtual Peer-to-Peer) - Write Mode

Dual master (virtual peer-to-peer) - write mode:

- The ICDM-RX/MOD and PLC initiate only write messages to each other.
- The ICDM-RX/MOD writes received serial and/or Ethernet device data directly into PLC memory with minimal latency.
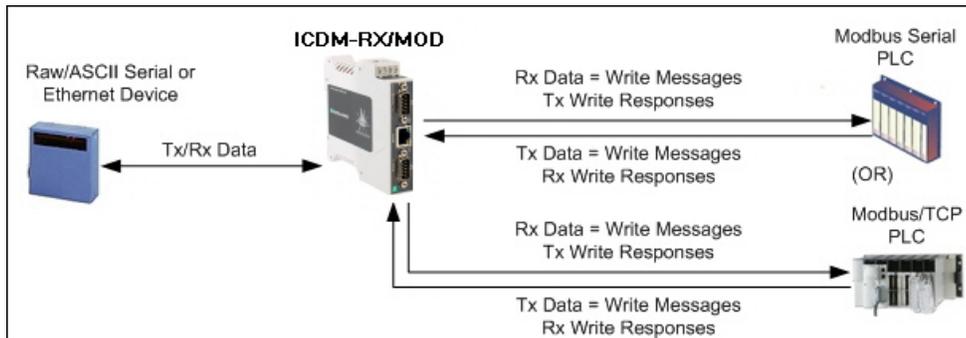
- The PLC can write to serial and/or Ethernet devices through the ICDM-RX/MOD with minimal latency.
- This mode provides the lowest possible Ethernet bandwidth usage and most efficient usage of PLC and ICDM-RX/MOD processing power.
- Only raw/ASCII devices are supported in this mode.
- The *ICDM-RX/MOD Receive Transfer* mode is set to *Master* and *Transmit Transfer* mode is set to *Slave*.

1.5.4. Dual Master (Virtual Peer-to-Peer) - Read Mode (Dual Polling)

Dual master (virtual peer-to-peer) - read mode (dual polling):

- This is provided for programmers who strongly prefer polling.
- The ICDM-RX/MOD and PLC initiate only read messages to each other.
- The PLC polls for received serial and/or Ethernet device data.
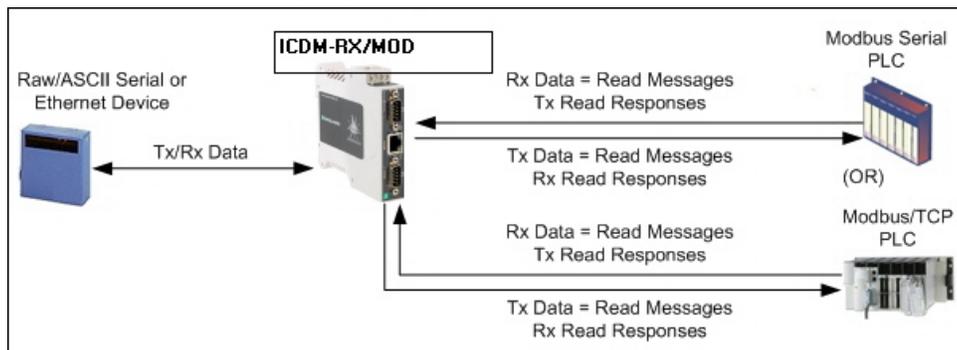
- The ICDM-RX/MOD polls for transmit data to serial and/or Ethernet devices.
- This mode requires the highest possible Ethernet bandwidth usage and provides the least efficient usage of PLC and ICDM-RX/MOD processing power.
- Only raw/ASCII devices are supported in this mode.
- The ICDM-RX/MOD *Receive Transfer* mode is set to *Slave* and *Transmit Transfer* mode is set to *Master*.

PEPPERL+FUCHS

1.5.5. Filtering and Data Extraction Functionality

The ICDM-RX/MOD provides the following filtering and data extraction functionality.
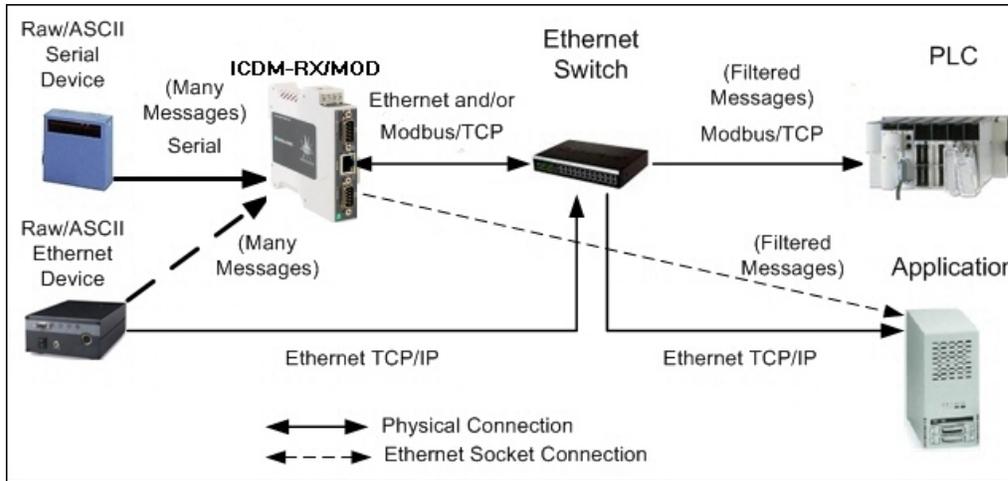
- Filtering:
  - String Filtering of up to 128 bytes of raw/ASCII data to both the PLC and/or application.
  - RFID filtering of EPCglobal formatted RFID tag data to both the PLC and/or application.
  - Barcode filtering of all UPC/EAN formatted barcodes data to both the PLC and/or application.
  - Simplifies PLC and application programming tasks.
- Data Extraction:
  - RFID data extraction extracts all parameters, such as company code, product code, and serial numbers, from any or all of the 43 EPCglobal tag formats. It then transfers the data to the PLC and/or application in a consistent and simple format.
  - Barcode data extraction extracts the company, product, and numbering codes from UPC/EAN formatted barcodes. It then transfers the data to the PLC and/or application in a consistent and simple format.
  - Simplifies PLC and application programming tasks.
- Environment specific support:
  - Support for multiple RFID reader tag formats.
  - RFID antenna grouping.
  - Aging of filtered string/RFID/barcode entries.
  - Discarding of unrecognized RFID and barcode messages.

For detailed information about filtering and data extraction, see the *ICDM-RX/EN and ICDM-RX/MOD with Modbus/TCP Filtering and Data Extraction Manual*.

1.6. Definitions and Terms

This section describes the Modbus/TCP definitions and terms included in the Modbus/TCP interface and supported by the ICDM-RX/MOD.

PEPPERL+FUCHS

1.6.1. Data Type Definitions

The following list defines the available data types.

Data Type	Definition
BYTE	Bit String (8-bits)
DINT	Signed Double Integer (32-bits)
DWORD	Bit String (32-bits)
INT	Signed Integer (16-bits)
STRING	Character String (1-byte per character)
UDINT	Unsigned Double Integer (32-bits)
USINT	Unsigned Short Integer (8-bits)
WORD	Unsigned Integer (16-bits)

1.6.2. Glossary

The following list defines terms associated with Modbus/TCP.

Term	Definition
Alias Device ID	The device ID that the original received ID is changed to when an Alias Device ID is configured.
Device ID	The address of the slave device and the term is identical to *Unit Identifier* and *Slave Address*.
Ethernet Device	A device that communicates through an Ethernet TCP/IP connection.
Master Device	A device that transmits Modbus/TCP messages to slave devices and receives the corresponding responses.
Modbus	An application layer messaging protocol that provides client/server communications between devices connected on different types of buses.
Modbus Serial	The Modbus protocol over a serial connection.
Modbus/ASCII	Modbus Serial in ASCII format. This form of Modbus communication requires two characters for each byte.
Modbus/RTU	Modbus Serial in binary format.
Modbus/TCP	The Modbus protocol over an Ethernet TCP/IP connection, also known as Modbus over Ethernet.
Raw Serial Device	A common serial device that communicates over serial ports through plain byte or ASCII data messages.
Slave Address	The address of the slave device. This term is identical to Unit Identifier and Device ID.
Slave Device	A device that only responds to Modbus messages.
Socket Port	The Ethernet socket port that is used to communicate to an Ethernet device.
Unit Identifier	The address of the slave device and the term is identical to *Device ID* and *Slave Address*.

PEPPERL+FUCHS

2. Programming Interface

2.1. Overview

The ICDM-RX/MOD provides highly flexible Modbus connectivity.

• Modbus masters supported include Modbus/TCP and Modbus/RTU and Modbus/ASCII serial masters.

• Both serial Modbus/RTU and Modbus/ASCII slave devices are supported.

• All Modbus masters can communicate with all Modbus slave devices.

• The Modbus/RTU and Modbus/ASCII Protocol Interface is defined in (*Modbus/RTU and Modbus/ASCII To-Slaves Protocol Interface* on Page 24).

The ICDM-RX/MOD provides highly advanced raw/ASCII device functionality:

• Both serial and Ethernet TCP/IP devices are supported.

• Modbus interfaces include Modbus/TCP masters, Modbus/TCP slaves, and both Modbus/RTU and Modbus/ASCII serial masters.

• Up to six Ethernet TCP/IP Application connections per serial or Ethernet TCP/IP device.

• The raw/ASCII interface is defined in (*Raw Data Interface* on Page 14).

You must configure the ICDM-RX/MOD through its embedded web pages discussed in the following chapters.

The ICDM-RX/MOD uses normal Modbus addressing conventions and provides receive, transmit, and statistical data.

Programming the PLC via Concept on Page 97 describes the Concept™ PLC programming examples provided with the ICDM-RX/MOD. It describes how to configure the ICDM-RX/MOD for raw serial data and start running the example programs using the embedded web pages and the example PLC program code.

Note: *While the Concept PLC example programs directly apply only to the Schneider Electric Momentum, Quantum, and Compact PLCs, they can be used as a guide for programming other PLCs.*

2.1.1. Modbus Master Requirements

Modbus Masters (Modbus/TCP, Modbus/RTU serial, and Modbus/ASCII) must meet these requirements:

• The Modbus Master must support the corresponding protocol.

• For raw/ASCII data, the Modbus Master must support the Read Holding Registers and Write Multiple Registers commands or, alternatively, the Read/Write Multiple Registers command.

• The Modbus Master must be able to write enough data in one message to handle the maximum sized messages required for the serial or Ethernet device.

PEPPERL+FUCHS

2.1.2. What is Modbus/RTU?

Modbus/RTU is native Modbus in hexadecimal format. These are the base Modbus messages that contain simple read and write requests. The format is as follows:

1 Byte	1 Byte	(Length dependent on message)	2 Bytes
Device ID	Function Code	Message specific parameters	CRC
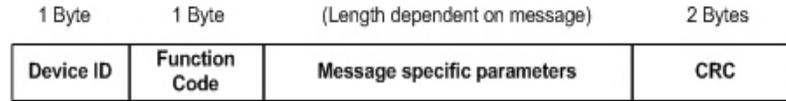
Modbus/RTU Message Format

Where:

• The terms Master or Client are used to identify the sender of the message.

• The terms Slave or Server are used to identify the devices responding to the message.

Modbus/RTU is used primarily for:

• *Serial port connectivity*. RS-485 is the most common serial mode, but RS-232 and RS-422 are also widely used. Commonly used by both Master and Slave devices.

• *Ethernet TCP/IP socket connections.* This is not the same as Modbus/TCP (please see section on Modbus/TCP), but does provide a very simple method of interfacing to remote devices. It is used by many applications and some OPC servers.

   Note: *This communication method typically is not supported by PLCs.*

2.1.3. What is Modbus/ASCII?

Modbus/ASCII is native Modbus in ASCII format. This protocol is used primarily by legacy devices and is no longer supported as widely as Modbus/RTU.

Like Modbus/RTU, Modbus/ASCII contains the base Modbus messages that contain simple read and write requests. The differences between Modbus/ASCII and Modbus/RTU are:

1. The message data is sent in ASCII format, so the message length is twice as long. It requires two ASCII characters for each byte of data.

2. An 8 bit LRC is attached to verify the message instead of a 16 bit CRC. The LRC is transmitted in ASCII format.

3. There are defined starting and ending characters to determine a Modbus/ASCII messages.

The format is as follows:

1 Start Char	2 Chars	2 Chars	(Length dependent on message)	2 Chars	2 Stop Chars
:	Device ID	Function Code	Message specific parameters	LRC	CR, LF

Modbus/ASCII Message Format

Where:

• The terms Master or Client are used to identify the sender of the message.

• The terms Slave or Server are used to identify the devices responding to the message.

Modbus/ASCII is used primarily for:

• *Serial port connectivity*. RS-485 is the most common serial mode, but RS-232 and RS-422 are also used. Used primarily by legacy Slave devices.

• *Ethernet TCP/IP socket connections*. This is not the same as Modbus/TCP (please see section on Modbus/TCP), but does provide a very simple method of interfacing to remote devices. It is used by some applications and some OPC servers.

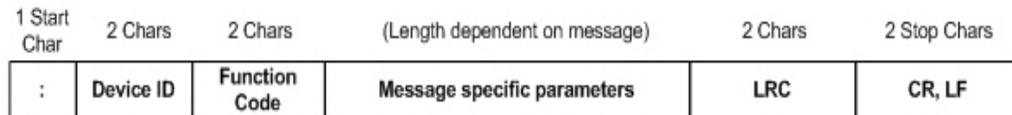   ***Note:*** *This communication method typically is not supported by PLCs.*

PEPPERL+FUCHS

2.1.4. What is Modbus/TCP?

Modbus/TCP is an Ethernet network based protocol that contains a Modbus/RTU message, with the exception of the 2 byte CRC. The Modbus/TCP message contains a header with information designed to provide message identification and routing information. The format is as follows:
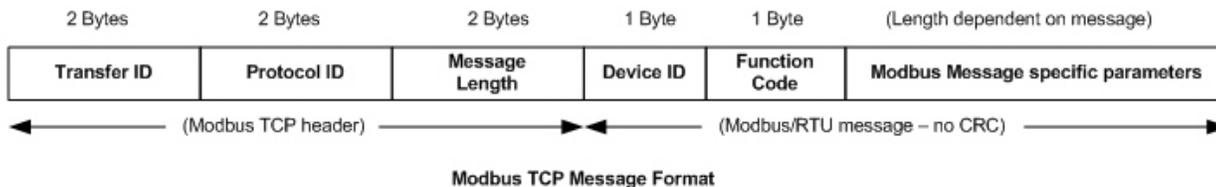
2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	(Length dependent on message)
Transfer ID	Protocol ID	Message Length	Device ID	Function Code	Modbus Message specific parameters

(Modbus TCP header) ◄────────► (Modbus/RTU message – no CRC) ────────►

Modbus TCP Message Format

Where:

• The terms Master or Client are used to identify the sender of the message.

• The terms Slave or Server are used to identify the devices responding to the message.

• Modbus TCP messages are typically sent to and received on a defined Ethernet TCP/IP socket of 502.

• Modbus TCP implementations provide more capability, but also require more processing than simpler Modbus/RTU implementations.

Modbus TCP is used for connecting advanced Ethernet based devices, such as PLCs, HMIs, SCADA Systems, and most OPC Servers to:

• Other Ethernet devices supporting Modbus TCP.

• Serial Modbus/RTU and/or Modbus/ASCII devices through gateways (such as the ICDM-RX/MOD running the Modbus/TCP or Modbus Router applications).

• Serial or Ethernet TCP/IP raw/ASCII devices (barcode scanners, printers, RFID readers, visions systems, etc) through a gateway (such as the ICDM-RX/MOD running the Modbus/TCP application).

2.2. Raw Data Interface

This subsection contains the following topics:

11/4/20

PEPPERL+FUCHS

2.2.1. Supported Modbus Messages

ICDM-RX/MOD supports the following Modbus messages over Modbus/TCP for raw data transfer.

Message Type	Function Code	Maximum Message Size	Maximum Serial Packet Size
Read Holding Registers	3	250 BYTEs (125 WORDs)	246 BYTEs (123 WORDs)
Write Multiple Registers	16 (10 hex)	240 BYTEs (120 WORDs)	236 BYTEs (118 WORDs)
Read/Write Multiple Registers	23 (17 hex)	236 BYTEs (118 WORDs)	232 BYTEs (116 WORDs)

Note: *Your PLC programming software may not allow maximum size serial packets.*

2.2.2. Serial Port Raw/ASCII Interface

Serial Port Raw/ASCII Addressing	Serial Port 1	Serial Port 2	Serial Port 3	Serial Port 4	Access Rule
Unit ID	255 (FF hex)	255 (FF hex)	255 (FF hex)	255 (FF hex)	N/A
Receive Data Address	1000 (Base 0) 1001 (Base 1)	2000 (Base 0) 2001 (Base 1)	3000 (Base 0) 3001 (Base 1)	4000 (Base 0) 4001 (Base 1)	Read Only
Receive Data Sequence Number Address	1256 (Base 0 1257 (Base 1)	2256 (Base 0) 2257 (Base 1)	3256 (Base 0) 3257 (Base 1)	4256 (Base 0) 4257 (Base 1)	Read/Write
Transmit Data Address	1300 (Base 0) 1301 (Base 1)	2300 (Base 0) 2301 (Base 1)	3300 (Base 0) 3301 (Base 1)	4300 (Base 0) 4301 (Base 1)	Read/Write
Transmit Data Sequence Number Address	1556 (Base 0) 1557 (Base 1)	2556 (Base 0) 2557 (Base 1)	3556 (Base 0) 3557 (Base 1)	4556 (Base 0) 4557 (Base 1)	Read/Write
Statistics Address	1600 (Base 0) 1601 (Base 1)	2600 (Base 0) 2601 (Base 1)	3600 (Base 0) 3601 (Base 1)	4600 (Base 0) 4601 (Base 1)	Read/Write

PEPPERL+FUCHS

2.2.3. Ethernet Device Raw/ASCII Interface

Socket Port Raw Data Addressing	Socket Port 1	Socket Port 2	Socket Port 3	Socket Port 4	Access Rule
Unit ID	254 (FE hex)	254 (FE hex)	254 (FE hex)	254 (FE hex)	N/A
Receive Data Address	1000 (Base 0) 1001 (Base 1)	2000 (Base 0) 2001 (Base 1)	3000 (Base 0) 3001 (Base 1)	4000 (Base 0) 4001 (Base 1)	Read Only
Receive Data Sequence Number Address	1256 (Base 0 1257 (Base 1)	2256 (Base 0 2257 (Base 1)	3256 (Base 0 3257 (Base 1)	4256 (Base 0 4257 (Base 1)	Read/Write
Transmit Data Address	1300 (Base 0) 1301 (Base 1)	2300 (Base 0) 2301 (Base 1)	3300 (Base 0) 3301 (Base 1)	4300 (Base 0) 4301 (Base 1)	Read/Write
Transmit Data Sequence Number Address	1556 (Base 0) 1557 (Base 1)	2556 (Base 0) 2557 (Base 1)	3556 (Base 0) 3557 (Base 1)	4556 (Base 0) 4557 (Base 1)	Read/Write

2.2.4. Raw/ASCII Transfer Modes

The ICDM-RX/MOD supports two different raw/ASCII message transfer modes. The default *Data-Stream* mode is the traditional transfer mode that asynchronously transmits messages and returns received data/responses. The *Command/Response* mode provides a *synchronous transfer* mode for sending and returning responses.

2.2.4.1. Data-Stream Transfer Mode

The *Data-Stream transfer* mode is the default transfer mode that asynchronously transmits messages from all Modbus and Application interfaces and returns received data/responses to all Modbus and Application interfaces. This mode is typically used in installations that utilize only one controller and for receive-only devices such as barcode scanners, RFID readers, weigh scales, and position encoders.

11/4/20

PEPPERL+FUCHS
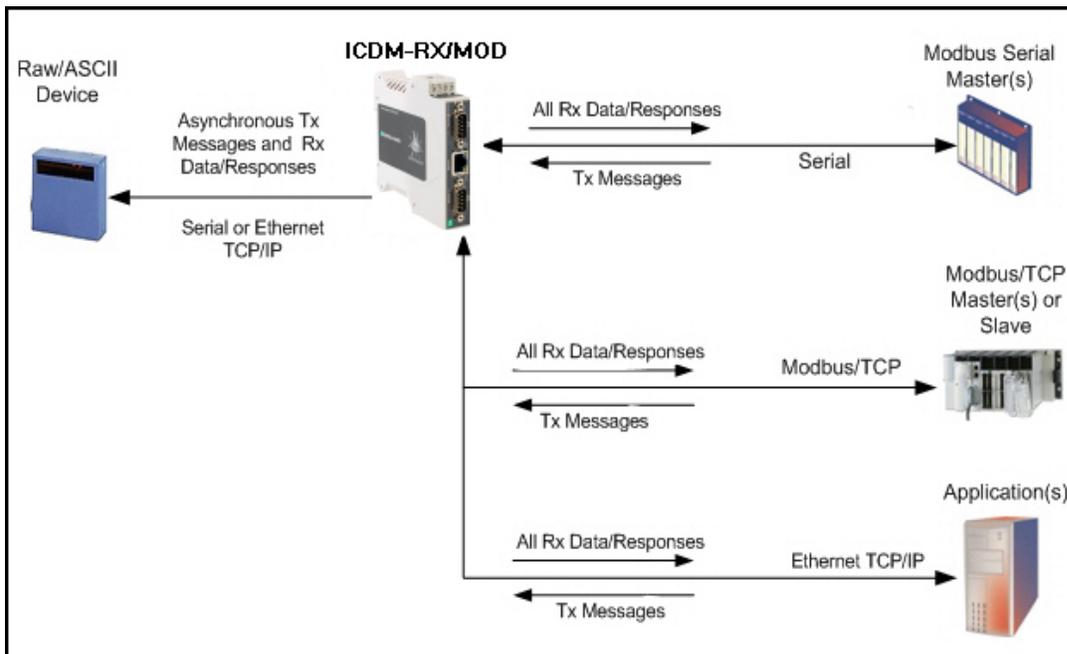
2.2.4.2. Command/Response Mode

The *Command/Response* mode provides the following functionality:

- A *synchronous transfer* mode for sending and returning responses from all Modbus and Application Ethernet TCP/IP interfaces to serial and Ethernet TCP/IP devices.
- Only one command message is transmitted at a time. Command messages are queued if a command message is active.
- Responses are routed only to the message sender.
- Responses are timed out and old responses, (ones not requested within a certain time frame), destined for the Modbus interface are discarded.
- The expected response count is configurable. While this is typically one, some devices may return multiple responses per message.

The *Command/Response transfer* mode is typically required in installations that require multiple controllers sending raw/ASCII messages with expected responses, and it is desired that each controller only receive its own responses.

PEPPERL+FUCHS
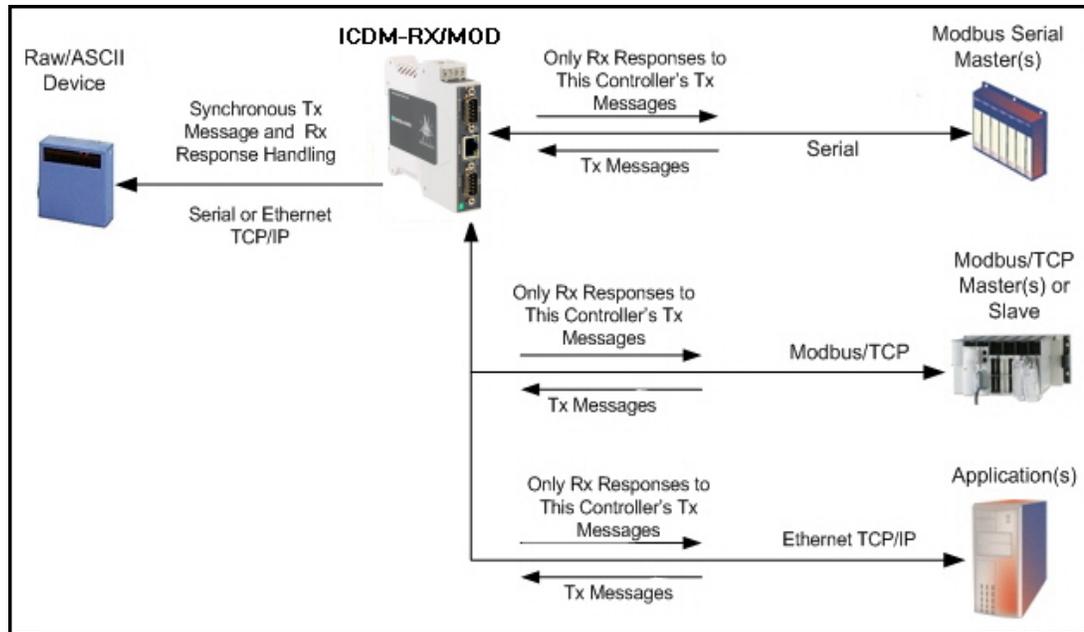
2.2.5. Receive Data Message (Raw Data)

The following topics are discussed:

- *Format on Page 18*
- *Communication Methodology (Receive Raw Data in Slave Mode) on Page 19*
- *Communication Methodology (Receive Data Master Mode) on Page 19*

2.2.5.1. Format

The *Receive Data* message for raw data contains a simple protocol including a sequence number, length and serial data fields. The Modbus standard requires a WORD format.

The following table displays the format of the *Receive Data* message.

Name	Data Type	Data Value(s)	Access Rule
Receive (ICDM-RX/MOD to PLC) message data			Read only
Structure of: Produced data sequence	WORD	0-65535 (FFFF hex)	
Data length (in bytes)	WORD	0-246 (slave Rx mode)	
Data array	Array of WORD	1-1024 (serial Rx master) 1-2048 (Ethernet Rx master)	

Receive messages have the following characteristics:

- It returns all data in WORDs.
- The ICDM-RX/MOD increments the sequence number when it returns new data.
- The message received from the PLC determines the actual length of the Modbus message returned to the PLC. (This is often greater than the length of the actual number of valid bytes in the Receive Data Message.)
- All unused bytes in a Modbus message returned to the PLC are filled with zeros.
- The default order of the bytes is Least Significant Byte First. However, you can select the *Rx MS Byte First* option in the web page to return bytes by **Most Significant Byte First**. For more information, see *Rx MS Byte First* under *Port Modbus Configuration (Raw/ASCII Only) Page* on Page 45.

11/4/20

PEPPERL+FUCHS

2.2.5.2. Communication Methodology (Receive Raw Data in Slave Mode)

Raw serial and/or EtherNet device data is returned in the response to the *Read Holding Registers* message or, optionally, the *Read/Write Multiple Register* message. The data is requested by accessing the corresponding receive data address for the desired port.
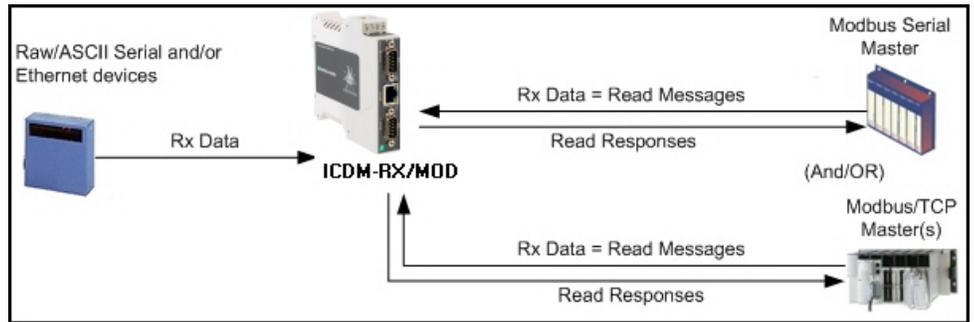
The following restrictions apply to this method:

- The *Device Index* must be 255 (FF hex) for raw/ASCII serial data and 254 (FE hex) for raw/ASCII Ethernet device data.
- The variable to receive the data on the PLC must be:
    - In the 40xxxx address range. (For Modicon type PLCs.)
    - An array of 16 bit words.
    - Of sufficient size to contain the sequence number, length, and data field associated with the received data structure. For more information, see the *Receive Data Message (Raw Data)* definition on Page 18.
- New data is indicated with an incremented sequence number.

    The same data may be returned more than once. However, the same data packet is also return the same sequence number.

- No data is indicated with a length of zero.

2.2.5.3. Communication Methodology (Receive Data Master Mode)

Raw serial and/or Ethernet device data is written to the PLC at the configured address.

The following restrictions apply to this method:

- The *Device Index* must be configured for the target PLC.
- The variable to receive the data on the PLC must be:

- In the 40xxxx. (For Modicon type PLCs.)
- An array of 16 bit words.
- Of sufficient size to contain the sequence number, length, and data field associated with the received data structure.
- New data is indicated with an incremented sequence number.

PEPPERL+FUCHS
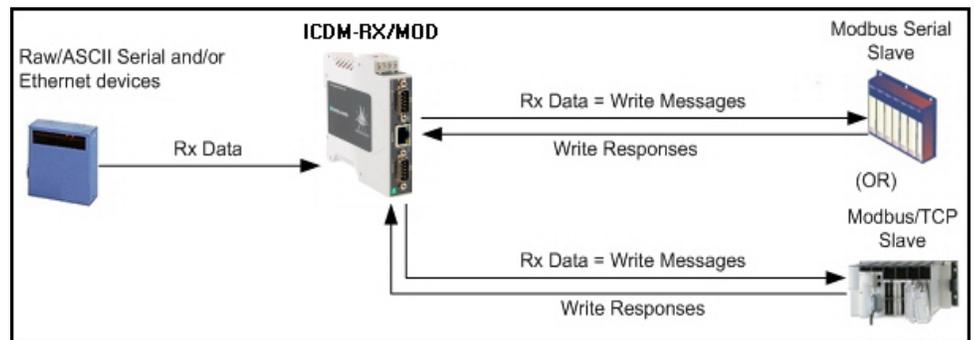
2.2.6. Transmit Data Message (Raw Data)

The following topics are discussed:

- *Format on Page 20*
- *Communication Methodology (Transmit Raw Data Slave Mode) on Page 21*
- *Communication Methodology (Transmit Data Master Mode) on Page 21*

2.2.6.1. Format

The *Transmit Data* message for raw data contains a simple protocol including a sequence number, length and serial data fields. The Modbus standard requires a WORD format.

The following table displays the format of the *Transmit Data* message.

| Name	Data Type	Data Value(s)	Access Rule
Transmit (PLC to ICDM-RX/MOD) message data			Read/Write
Structure of:			
Produced data sequence	WORD	0-65535 (FFFF hex)	
Data length (in bytes)	WORD	1-236 (Slave Tx Mode)	
		1-246 (Master Tx Mode)	
Data array	Array of WORD	0-65535	

Transmit messages have the following characteristics:

- It transfers all data in WORDs.
- If the **Disable Tx Sequence Number Check** option is not selected, the sequence number must be incremented when there is new data to transmit.
- The data length field indicates the number of valid bytes contained in the message.
- The actual length of a message received from the PLC may contain extra, unused data.
- It ignores all unused bytes in a Modbus message.
- The default order of the bytes is **Least Significant Byte First**. However, you can select the *Tx MS Byte First* option in the web page to transmit bytes by **Most Significant Byte First**. For more information, see *Tx MS Byte First* under *Port Modbus Configuration (Raw/ASCII Only) Page* on Page 45.
- A request for the *Transmit data* returns the last transmit data message.

PEPPERL+FUCHS

2.2.6.2. Communication Methodology (Transmit Raw Data Slave Mode)

Raw serial and/or EtherNet device data is sent in the *Write Multiple Registers* message or, optionally, the *Read/ Write Multiple Register* message. The data is requested by accessing the corresponding transmit data address for the desired port.
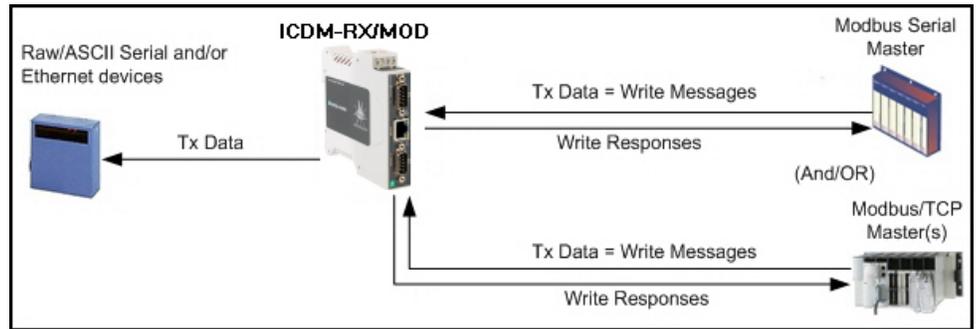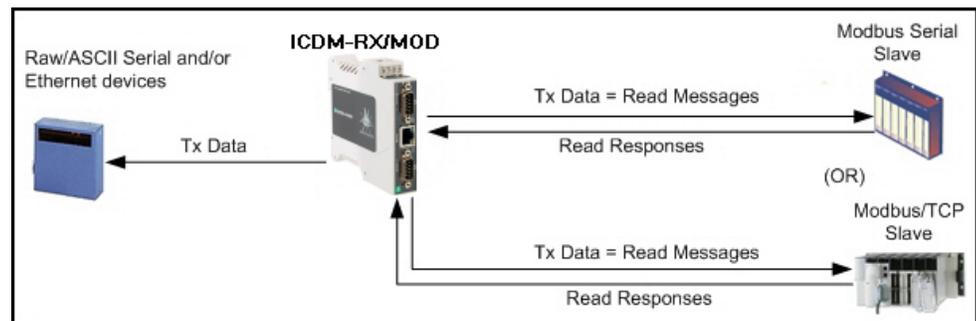

The following restrictions apply to this method:

- The *Device Index* must be 255 (FF hex) for raw/ASCII serial data and 254 (FE hex) for raw/ASCII Ethernet device data.

- The variable to transmit the data on the PLC must be:
  - In the 40xxxx address range. (For Modicon type PLCs.)
  - An array of words.
  - Of sufficient size to contain the sequence number, length, and data field associated with the transmit data structure, typically 128 words. See *Transmit Data Message (Raw Data)* on Page 20 for more information.

- If the **Disable Tx Sequence Number Check** option is not selected, the sequence number must be incremented when there is new data to transmit. The same transmit data message may be sent to the ICDM-RX/MOD more than once. However, the data packet is only transmitted when a new sequence number is received.

2.2.6.3. Communication Methodology (Transmit Data Master Mode)

Raw serial and/or Ethernet transmit data is polled from the PLC at the configured address and, when the ICDM-RX/ MOD receives a transmit message with an updated sequence number, the data is transmitted to the serial or Ethernet device.

- The following restrictions apply to this method:

- The *Device Index* must be configured for the target PLC.

- The variable to receive the data on the PLC must be:
  - In the 40xxxx. (For Modicon type PLCs.)
  - An array of 16 bit words.
  - Of sufficient size to contain the sequence number, length, and data field associated with the transmit data structure.

- The PLC indicates new data to transmit with an incremented sequence number. (The **Disable Tx Sequence Number Check** option does not apply to *transmit data master* mode.)

- The length indicates the number of bytes to transmit.

11/4/20

PEPPERL+FUCHS

- The ICDM-RX/MOD expects the length parameter and data to transmit to be updated before the transmit sequence number is incremented. Therefore, as soon as the ICDM-RX/MOD receives an incremented transmit number, it transmits the data to the serial or Ethernet device.

2.2.7. Sequence Number Messages (Raw Data)

Read Holding Registers and *Write Multiple Register* messages can read and modify both receive and transmit produced data sequence numbers. These are the same sequence numbers returned to the PLC in the *Receive Data Message* and sent to the ICDM-RX/MOD in the *Transmit Data* message. Access to these sequence numbers are provided primarily for initialization purposes at the start of the PLC program when you may want to initialize the sequence numbers on the PLC, ICDM-RX/MOD or both.

2.3. I/O Scanner (Raw Data)

The I/O Scanner is an optional PLC communications method that is implemented on some PLC programming software such as the Concept programming package for use with the Schneider Electric Modicon PLCs. The I/O Scanner provides a rather simple method that requires minimal programming effort. It automatically performs the polling and transmitting of data at set time intervals and typically utilizes the *Read/Write multiple Registers* message.


Modbus/TCP I/O Scanner Raw Data Flow
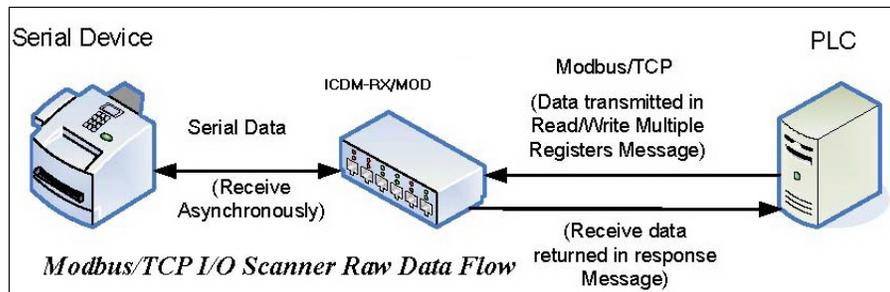
The following restrictions apply to this method:

- The *Receive* and *Transmit* mode for the serial and/or Ethernet device must both be set to *Slave* mode.
- The *Device Index* must be 255 (FF hex) for raw/ASCII serial data and 254 (FE hex) for raw/ASCII Ethernet device data.
- The variable to receive the data on the PLC must be:
    - In the 40xxxx address range. (For Modicon type PLCs.)
    - An array of words.
    - Of sufficient size to contain the sequence number, length, and data field associated with the received data structure, typically 128 words. For more information, see the *Receive Data Message (Raw Data)* definition on Page 18.
- New received data is indicated with an incremented sequence number.

    The same data may be returned more than once. However, the same data packet returns the same sequence number.
- No receive data is indicated with a length of zero.
- The variable to transmit the data on the PLC must be:
    - In the 40xxxx address range. (For Modicon PLCs.)
    - An array of words.
    - Of sufficient size to contain the sequence number, length, and data field associated with the transmit data structure, typically 128 words. See *Transmit Data Message (Raw Data)* on Page 20 for more information.

11/4/20

PEPPERL+FUCHS

- If the **Disable Tx Sequence Number Check** option is not selected, the sequence number must be incremented when there is new data to transmit.

  The same transmit data message may be sent to the ICDM-RX/MOD more than once. However, the data packet is only transmitted when a new sequence number is received.

- The ICDM-RX/MOD should be reset before starting a PLC program using the I/O Scanner due to PLC program execution scheduling. If the ICDM-RX/MOD is not reset, the sequence numbers may be out of sync. This may result in receiving outdated serial data as well as an unexpected transmission of serial data. A Transmit Unexpected Sequence Number error may also occur.

The following depicts a typical I/O Scanner screen.

PEPPERL+FUCHS
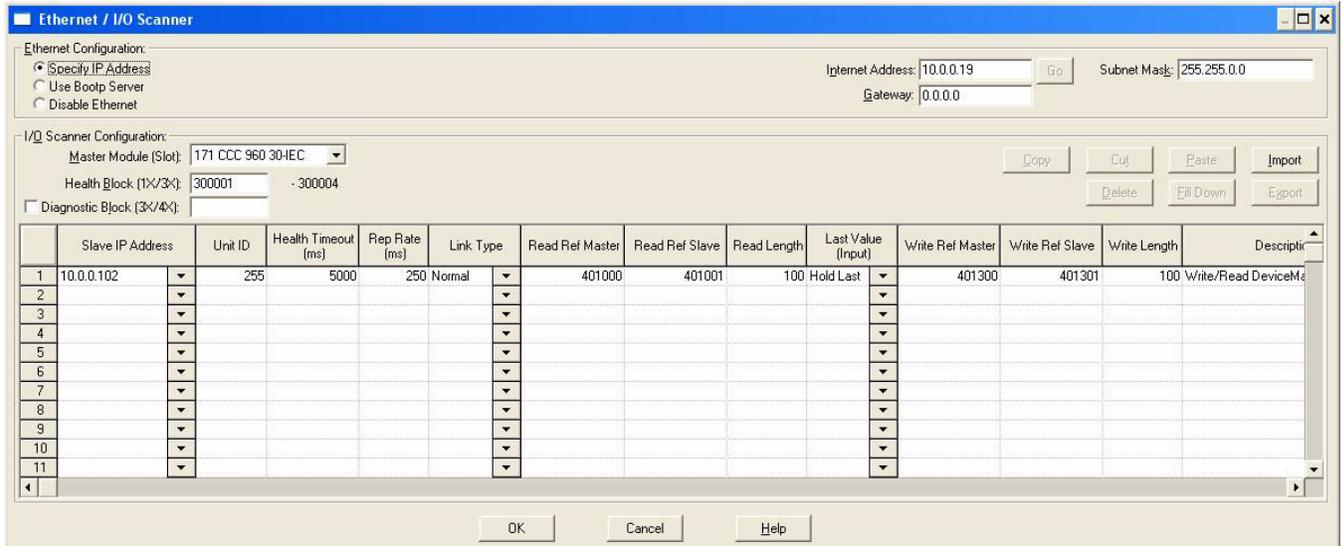
2.4. Modbus/RTU and Modbus/ASCII To-Slaves Protocol Interface

The ICDM-RX/MOD provides access to serial Modbus/RTU and Modbus/ASCII slave devices via Modbus/TCP, serial Modbus/RTU masters, and serial Modbus/ASCII masters. Modbus master messages are translated to Modbus/RTU or Modbus/ASCII messages, devices are automatically located, and appropriate Modbus responses are returned to the Modbus masters.

2.4.1. Communication Methodology

The ICDM-RX/MOD translates Modbus master messages into Modbus/RTU or Modbus/ASCII messages and forwards them to slave devices attached to the Modbus/RTU or Modbus/ASCII slave serial ports. Each Modbus message is transmitted and a response is expected. The ICDM-RX/MOD times out the Modbus/RTU or Modbus/ASCII messages if there is no response returned within the configured timeout period.

The following diagram displays the Modbus message transfer.
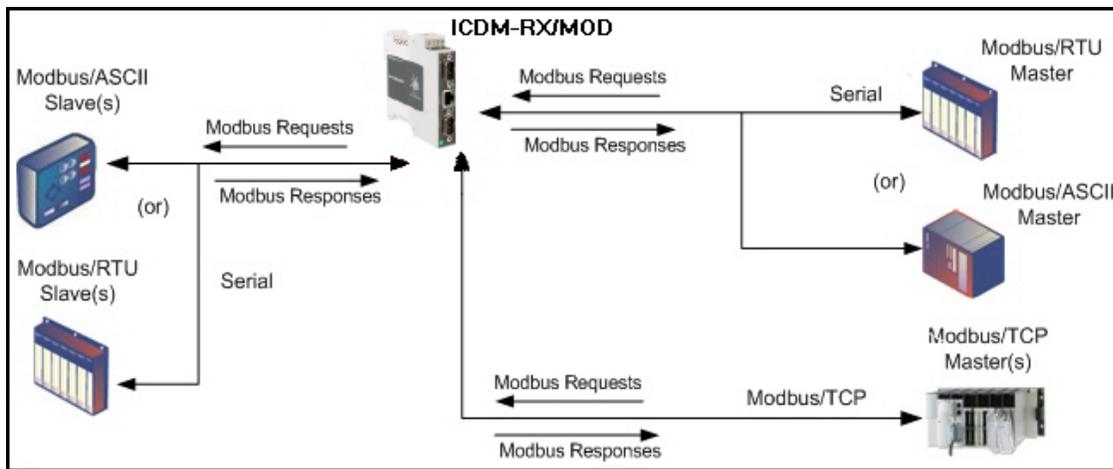
The following apply to Modbus slaves serial ports.

• All valid Modbus master messages are translated to Modbus slave messages for serial port transmission.

• Modbus slave devices are automatically located on a ICDM-RX/MOD 2-port or 4-port.

• Messages are timed out if no response is returned within the configured timeout period.

• Appropriate Modbus responses are returned to the Modbus master.

• Broadcast Modbus messages, those with a unit identifier of zero, are transmitted out all Modbus slave serial ports on the ICDM-RX/MOD.

The following restrictions apply to the Modbus slave interface:

• The ICDM-RX/MOD serves as a *slave* Modbus/TCP device, a *master* on Modbus To-Slaves serial ports, and a slave on Modbus To-master serial ports.

• All Modbus slave devices attached to a ICDM-RX/MOD gateway (1, 2, or 4-port) must have unique *Unit Identifiers*. Valid Unit Identifiers are 1 to 247 and the Broadcast Identifier is zero.

To communicate to Modbus slave device(s) through a ICDM-RX/MOD, perform the following steps.

1. Using the embedded web page, select the appropriate **Port**.

2. Under *Serial Configuration*, configure the serial port parameters such as the Mode, Baud rate, Data Bits, and so forth.

3. Under *General Protocol Settings*, set the *Select Serial Port Protocol* to **Modbus slave**.

4. Under *Modbus Slave Protocol Settings*, set the **Device Response Timeout** to the desired value.

   Note: *2- and 4-Port only: set the **Lost Device Search Enable** setting. For a discussion on this setting,*

PEPPERL+FUCHS

5. In the PLC program, address messages to the Modbus slave device using the IP Address of the ICDM-RX/MOD and the Unit Identifier of the slave device(s).

2.4.2. Modbus Slave Device Search Methodology

Locating a Modbus slave device on a ICDM-RX/MOD 1-port is relatively simple. Either the Modbus slave device is connected to the port or it is not. However, if more than one port is configured for Modbus slave on a ICDM-RX/MOD 2- or 4-port, the device must be found. The following is an explanation of how the search algorithm works on a ICDM-RX/MOD 2- or 4-port.

Locating a Modbus slave device after a reboot or port reset:

When the ICDM-RX/MOD receives a message for a Modbus slave device for the first time since reboot or port initialization, it transmits the Modbus slave message out all Modbus slave serial ports and wait for a response to be returned. Once the response is returned, the device port is known and all messages sent to the device are routed through the serial port.

Lost Devices:

Lost devices, or devices that time out, are a special case. The ICDM-RX/MOD provides two methods for handling lost devices via the **Lost Device Search Enable** option on the web page.

- Disabling this option on a Modbus slave port:
    - Prevents the ICDM-RX/MOD from searching for a lost device on other Modbus slave ports.
    - Prevents lost devices known to have been on other ports from being searched for on this port.

    Note: This is the recommended setting whenever it is desired to prevent timeout delays on other Modbus slave ports in the event that a device times out.

- Enabling this option on a Modbus slave port:
    - Allows the ICDM-RX/MOD to search for lost devices on all Modbus slave ports with the **Lost Device Search Enable** option turned on.

    Note: This can be useful for locating devices if a device has been moved onto another port by moving the serial cable or, perhaps, by moving the device onto a different Modbus slave serial loop.

    - This causes timeout delays on all Modbus slave ports with the **Device Search Enable** option turned on until the device is found.

2.5. Retrieve Statistics Message

The data returned from the *Retrieve Statistics* message contains various counters. The *Retrieve Statistics* message formats the data into 32-bit integers and returns data in an array of WORDs. The first WORD contains the most significant word and the second WORD contains the least significant word.

ICDM-RX/MOD Addressing for the Statistics Modbus/TCP Messages	Serial Port 1	Serial Port 2	Serial Port 3	Serial Port 4	Access Rule
Unit ID	255 (FF hex)	255 (FF hex)	255 (FF hex)	255 (FF hex)	N/A
Statistics Address	1600 (Base 1) 1599 (Base 0)	2600 (Base 1) 2599 (Base 0)	3600 (Base 1) 3599 (Base 0)	4600 (Base 1) 4599 (Base 0)	Read Only

Note: Some Modicon PLC programming software, such as Concept, requires one to be added to the address offset. This is because their address range begins at 40001, while the address range on the ICDM-RX/MOD begins at zero.

PEPPERL+FUCHS

The following table displays the format of the *Retrieve Statistics* message.

Index	Name	Data Type	Data Value(s)	Access Rule
1	Receive Byte Count	UDINT	0=default	Read only
2	Receive Packet Count	UDINT	0=default	Read only
3	Transmit Byte Count	UDINT	0=default	Read only
4	Transmit Packet Count	UDINT	0=default	Read only
5	Dropped Packet Count to PLC	UDINT	0=default	Read only
6	Parity Error Count	UDINT	0=default	Read only
7	Framing Error Count	UDINT	0=default	Read only
8	Overrun Error Count	UDINT	0=default	Read only
9	Unexpected Transmit Sequence Number errors	UDINT	0=default	Read only
10	Invalid Modbus/RTU Device Responses	UDINT	0=default	Read only
11	Modbus/RTU Device Timeouts	UDINT	0=default	Read only
12	Reserved	UDINT	0	Read only

The *Retrieve Statistics* messages have the following characteristics.

Retrieve Statistics Message Description	
Receive Byte Count	This attribute counts the number of bytes received on the serial port.
Receive Packet Count	This attribute counts the number of packets received on the serial port.
Transmit Byte Count	This attribute counts the number of bytes transmitted on the serial port.
Transmit Packet Count	This attribute counts the number of packets transmitted on the serial port.
Dropped Packet Count to PLC	This attribute counts the number of dropped receive packets on the serial port intended for the PLC due to: • No STX byte(s) found • No ETX byte(s) found • Time-outs • Too large of packet • Receive buffer queue overflows
Parity Error Count	This attribute counts the number of packets with parity errors received on the serial port.
Framing Error Count	This attribute counts the number of packets with framing errors received on the serial port.
Overrun Error Count	This attribute counts the number of packets with overrun type errors received on the serial port.
Unexpected Transmit Sequence Number Error Count	This attribute counts the number of Unexpected Transmit Sequence Number errors. The ICDM-RX/MOD increments this number when it receives a raw data transmit message with a sequence number that is not equal to either the previous sequence number or the previous sequence number plus one. (The ICDM-RX/MOD expects this sequence number to be incremented by one with each new transmit message.)

PEPPERL+FUCHS

Retrieve Statistics Message Description (Continued)	
Invalid Modbus/RTU Device Responses	The number of invalid messages returned from Modbus/RTU devices on this port. Such invalid responses could be the result of any or all of the following: • Invalid CRC • Invalid returned function code • Invalid Unit Identifier • Duplicate Unit Identifier
Modbus/RTU Device Timeouts	The number of messages that timed out waiting for a response from a Modbus/RTU device on this port.

PEPPERL+FUCHS

3. Configuration Overview

The latest Modbus/TCP firmware should be installed before you configure network or serial/socket port characteristics. For firmware installation and setup information, see the *ICDM-RX Hardware Installation and Configuration Manual* or the PortVision DX help system.

Note: *Modbus Router is loaded on the ICDM-RX/MOD by default. If necessary, contact Pepperl+Fuchs Technical Support for the firmware.*

Use the *ICDM-RX/MOD Quick Start* to locate detailed configuration procedures for your site. Use the following chapters as a reference if you need information about specific fields. The *ICDM-RX/MOD Modbus/TCP Quick Start* is intended to provide you with a way to quickly configure the ICDM-RX/MOD for your devices.

3.1. Home Page

If you have not configured the network information into the ICDM-RX/MOD during initial setup, you must configure the network information before configuring serial/socket port characteristics. Refer to the PortVision DX help system for help configuring the network settings.

From PortVision DX, highlight the ICDM-RX/MOD that you want to configure and click **Webpage** or enter the IP address of the ICDM-RX/MOD in the **Address** box of your web browser. The **Home** page displays.

Modbus/TCP Home Page	
Firmware	Modbus/TCP firmware version currently running on the ICDM-RX/MOD.
Device Name	You can enter a Device Name in the *Network	Configuration* page, which will display in this field.
Type Code	This is the model name, which is also displayed at the top right of every page.
Article Number	This is the part number of this ICDM-RX/MOD.

11/4/20

PEPPERL+FUCHS

Modbus/TCP Home Page (Continued)	
Serial Number	The ICDM-RX/MOD serial number.
MAC Address	This is the MAC address of this ICDM-RX/MOD, which is located on the compliance label on the ICDM-RX/MOD.
System Uptime	Displays how long this ICDM-RX/MOD has been on-line since powered on or rebooted.
IP Config	Type of IP configuration currently in use (static or DHCP).
IP Address, IP Netmask, IP Gateway	IP address, netmask, and gateway configured in the ICDM-RX/MOD.

Use the appropriate chapter as a reference for the configuration options.

- *Serial Menus* on Page 38
- *Ethernet Menus* on Page 55
- *Modbus Menus* on Page 66

You can review the configuration pages and basic procedures in the following subsections.

- *Serial Port - Configuration Overview*
- *Ethernet Device - Configuration Overview* on Page 34

PEPPERL+FUCHS

3.2. Serial Port - Configuration Overview

Use the following steps to access the serial port configuration pages.

1. Open the ICDM-RX/MOD web page using the IP address in your browser or PortVision DX.
2. Click the **Serial** menu. The *Serial Port Overview* page displays.

| COMTROL | Home | Serial | Ethernet | Modbus | Network | Diagnostics | System | ICDM-RX/MOD-2DB9/2RJ45 | Logout |

Overview   Port 1   Port 2

Serial Port Overview

This is the current configuration of the serial port(s). Select a serial port from the menu above to change the configuration.

Serial Port Settings	Port 1	Port 2
Port Name:		
Port Mode:	RS-232	RS-232
Baud Rate:	230400	9600
Parity:	none	none
Data Bits:	8	8
Stop Bits:	1	1
Flow Control:	none	none
RS-485 Terminating Resistor:	off	off
DTR Mode:	off	off
Rx Timeout Between Packets (ms):	200	200
Discard Messages With Errors:	yes	yes
General Protocol Settings		
Serial Port Protocol:	Raw-Data	Raw-Data
Modbus Slave and Raw/ASCII Data Device Settings		
Response Timeout (ms):	N/A	N/A
Modbus Slave Settings		
Lost Device Search Enable:	N/A	N/A
Raw/ASCII Data Control		
Raw/ASCII Data Message Transfer Mode:	Data-Stream	Data-Stream
Cmd/Resp Age Time, Discard Responses After (sec):	N/A	N/A
Cmd/Resp Expected Responses Per Command:	N/A	N/A
Cmd/Resp Mode Response To Modbus/TCP Based On:	N/A	N/A
Serial Packet ID Settings		
STX Rx Detect:	none	one byte
STX Rx (dec):		2
ETX Rx Detect:	none	one byte
ETX Rx (dec):		3
PLC Specific Settings		
STX Tx Append:	none	none
STX Tx (dec):		

© Pepperl+Fuchs Comtrol, Inc.

PEPPERL+FUCHS

3. Click the port number that you want to configure and the *Serial Settings | Port Configuration* page appears.

4. Change the serial port configuration properties (Page 38) as required for your site.
5. Click the **Save** button.
6. If desired, click the **Raw/ASCII Settings** menu (Page 42) and configure the settings for your site.

7. Click the **Save** button.

PEPPERL+FUCHS

8. Click **Modbus Settings** (Page 45) to configure Modbus for this ICDM-RX/MOD.

9. Make the changes for your site and click the **Save** button.

PEPPERL+FUCHS

10. If required for your site, click **Filtering** and change the filtering configuration (Page 48) to meet your needs.

11. Click the **Save** button when you have completed the changes.

12. If required for your site, click **Application Interface** and change the configuration (Page 52) to meet your needs.

13. Click the **Save** button when you have completed the changes.

PEPPERL+FUCHS

3.3. Ethernet Device - Configuration Overview

Use the following overview to configure an Ethernet device.

1. Click the **Ethernet** menu. The *Ethernet Device Overview* page displays.

2. Click the appropriate **Device** number to access the *Device Interface Configuration* page for that port.

11/4/20

PEPPERL+FUCHS

3. Change the socket configuration properties (Page 56) as required for you site.

4. Click the **Save** button after completing the changes.
5. If necessary, click **Raw/ASCII Settings**.

6. Update the settings (Page 58) to meet your site requirements.
7. Click the **Save** button after completing the changes.

11/4/20

PEPPERL+FUCHS

8. Click **Modbus/TCP Settings** to configure Modbus/TCP for this ICDM-RX/MOD.

9. Make the necessary changes (Page 61) for your environment and then click **Save**.
10. If required for your site, click the **Filtering** menu.

PEPPERL+FUCHS

11. Change the parameters (Page 64) to meet your needs.

12. Click the **Save** button.

13. If required, click the **Application Interface** option.

14. Make the necessary changes (Page 65) for your site.

15. Click the **Save** button.

16. Repeat the previous steps for each serial port and Ethernet device.

17. Go to *Programming the PLC via Concept* on Page 97 to complete the ICDM-RX/MOD installation.

11/4/20

PEPPERL+FUCHS

37

4. Serial Menus

This chapter provides detailed information about the following web serial configuration pages:

- *Serial Port Overview Page* on Page 39
- *Port Serial Configuration Page* on Page 40
- *Port Raw/ASCII Configuration Page* on Page 42
- *Port Modbus Configuration (Raw/ASCII Only) Page* on Page 45
- *Port Filtering/Data Extraction Configuration (Raw/ASCII Only) Page* on Page 48
- *Application TCP Configuration (Raw/ASCII Only) Page* on Page 52

Note: *The Diagnostics menu is discussed in Diagnostic Menus on Page 78.*

The latest Modbus/TCP firmware must be installed before you should configure serial port characteristics.

Use the *ICDM-RX/MOD Modbus/TCP Quick Start* to locate configuration procedures for your site and use this chapter as a reference if you need information about specific fields. The *Quick Start* is intended to provide you with a way to quickly configure ICDM-RX/MOD for your devices.

PEPPERL+FUCHS

4.1. Serial Port Overview Page

You can access the *Serial Port Overview* page by clicking the **Serial** menu. This page provides an overview of all of the serial settings on the serial configuration pages.

11/4/20

4.2. Port Serial Configuration Page

Access the *Port Serial Configuration* page by clicking **Serial |Port | Serial Settings** to configure serial port characteristics for the device that you plan on connecting to the port.

Port Serial Configuration Page	
Port Name	Up to 80 character ASCII string. A user definable string used to describe the serial interface. Valid characters include a-z, A-Z, 0-9, underscores, spaces and dashes. All other characters are discarded. The default name is blank.
Port Mode	Select the communications mode for the serial device that you are connecting to the port. The available modes are RS-232, RS-422, and RS-485.
Baud Rate	Select a baud rate from the list. The baud rate that you select determines how fast information is transferred through a port.
Parity	Select a method for error checking. • **None** - When the parity is set to none, there is no parity bit, and ICDM-RX/MOD does not perform parity checking. • **Odd** - Indicates that the sum of all the 1-bits in the byte plus the parity bit must be odd. When the total is odd, the parity bit is set to zero, when it is even, the parity bit is set to one. • **Even** - When the sum of all the 1-bits is even, the parity bit must be set to zero; when it is odd, the parity bit must be set to one.
Data Bits	Select the number of bits that make up the data. Choose from 5, 6, 7 or 8-bits.
Stop Bits	Select the number of bits to mark the end of data transmission.

11/4/20

PEPPERL+FUCHS

Port Serial Configuration Page (Continued)	
Flow Control	Specifies the ability to start and stop the flow of data without the loss of bytes. Select a method for controlling the flow of data from the following list: • **None** - Indicates flow control is not in affect. • **RTS/CTS** - Request To Send (RTS) tells the receiving device that the sending device has data that is ready to send and Clear To Send (CTS) indicates the device is ready to accept data. • **XON/XOFF** - When selected, applies the standard method of controlling data flow between two modems. • **Half Duplex** - Transmits data in half-duplex mode.
DTR Mode	Select the state of Data Terminal Ready (DTR). • **on** - Enables DTR. • **off** - Disables DTR. • **WhenEnabled** - Select this option when enabling the serial port through the PLC.
Rx Timeout Between Packets (ms)	Specifies the following information, once the start of a packet is received: • How long the ICDM-RX/MOD should wait (in milliseconds) before timing-out, if the **ETX Rx Detect** length is one byte or two bytes and the ETX byte(s) are not received. • The time to wait in milliseconds between serial packets if the **ETX Rx Detect** length is set to **none**.
Discard Rx Pkts with Errors	By default, this box is checked and the ICDM-RX/MOD discards serial packets with errors. Clear the check box when you need to receive a serial packet with errors to troubleshoot an issue.
Clone settings to all ports	Selecting this option before selecting **Save** applies these settings to all of the serial ports.
General Protocol Settings	
Serial Port Protocol	This is the serial port protocol. • If you select **Raw-Data**, the port receives raw/ASCII type data. • If you select **Modbus/RTU-to-Slaves**, the serial port operates in *Modbus/RTU slave(s)* mode. • If you select **Modbus/ASCII-to-Slaves**, the serial port operates in to Modbus/ASCII slave(s) mode. • If you select **Modbus/RTU-to-Master**, the serial port operates in to Modbus/RTU master mode. • If you select **Modbus/ASCII-to-Master,** the serial port operates in to Modbus/ASCII master mode.
Modbus Slave and Raw/ASCII Data Device Settings	
Response Timeout (ms)	Modbus Slave and Raw-Data Command/Response mode timeout setting. The ICDM-RX/MOD waits for the response(s) until this time has elapsed before transmitting another message. The default is 250 ms.
Modbus Slave Settings	
Lost Device Search Enable (2- and 4-port, only)	If this is set, lost devices that were on this Modbus to-slave port is searched for on other Modbus to-slave ports that also have this option set. For a discussion of this setting, see *Modbus Slave Device Search Methodology* on Page 25.

4.3. Port Raw/ASCII Configuration Page

Access the *Port Raw/ASCII Configuration* page by clicking **Serial | Port | Raw/ASCII Settings**.

Port Raw/ASCII Configuration Page	
Raw/ASCII Data Control	
Raw-Data Message Transfer Mode	If you select **Data-Stream (default)**, the serial port operates in Data-Stream mode, see *Data-Stream Transfer Mode* on Page 16. If you select **Command/Response**, the serial port operates in Command/Response mode, see *Command/Response Mode* on Page 17.
Cmd/Resp Age Time, Discard Responses After (seconds)	The Age Time, or elapsed time, when responses destined for Modbus interface(s) are discarded. This typically occurs when a Modbus controller stops communicating and leaves a response in the response queues. The default is 10 seconds.
Cmd/Resp Expected Responses Per Command	The number of responses expected for each transmitted command. The default is 1.
Cmd/Resp Mode Response To Modbus/TCP Based On	Method or returning responses to Modbus/TCP interfaces. • If you select **IP-Address** (default), all responses are returned to the IP-Address of the original command was received from. This may the same or different TCP connection. • If you select **TCP-Connection**, all responses are returned to the same TCP connection the original command was received from. ***Note:*** **TCP-Connection** *is typically required when multiple controllers are communicating from the same PLC or computer.*

11/4/20

PEPPERL+FUCHS

Port Raw/ASCII Configuration Page (Continued)
Serial Packet Identification

STX (Start of Transmission) Rx Detect	When enabled, the ICDM-RX/MOD detects an STX (start of transmission) byte sequence which is configured as **one byte** or **two bytes** when it receives a serial packet. The length indicates the number of STX bytes, valid values for length are: • **none** - Disables this function and the ICDM-RX/MOD accepts the first byte received after the last ETX byte(s) as the start of the next data packet. • **one byte** - Scans serial data for one STX byte and when the ICDM-RX/MOD finds an STX byte it collects the data. If the first byte is not the STX byte, it discards the byte. The ICDM-RX/MOD continues to discard the bytes until it finds an STX byte. • **two bytes** - Scans serial data for two STX bytes and when the ICDM-RX/MOD finds two STX bytes it collects the data. If the STX bytes cannot be found, it discards the bytes. The ICDM-RX/MOD continues to discard the bytes until it finds the two STX bytes. **Byte 1** - Specifies the character that represents the first STX byte. The ICDM-RX/MOD looks for this character in the first STX byte, if the length is **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the character that represents the second STX byte. The ICDM-RX/MOD looks for this character in the second STX byte, only if the length is **two bytes**. You can specify a value between 0 and 255 in decimal format.
ETX (End of Transmission) Rx Detect	When enabled, the ICDM-RX/MOD detects an ETX (end of transmission) byte sequence that is configured as **one byte** or **two bytes** marking the end of the serial packet. The length indicates the number of ETX bytes, valid values for length are: • **none** - Disables this function and the ICDM-RX/MOD uses the *Rx Timeout Between Packets* to indicate the end of data packet. • **one byte** - Scans serial data for one ETX byte and when the ICDM-RX/MOD finds the ETX byte, it identifies the data as a serial packet. • **two bytes** - Scans serial data for two ETX bytes and when the ICDM-RX/MOD finds the ETX bytes, it identifies the data as a serial packet. **Byte 1** - Specifies the character to scan for in the first ETX byte, if the length is **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the character to scan for in the second ETX byte, if the length is **two bytes**. You can specify a value between 0 and 255 in decimal format.
Discard Rx Pkts with Errors	By default, this box is checked and the ICDM-RX/MOD discards serial packets with errors. Clear the check box when you need to receive a serial packet with errors to troubleshoot an issue.

PEPPERL+FUCHS

Port Raw/ASCII Configuration Page (Continued)	
PLC Specific Settings and Application Specific Settings Options	
STX Tx Append	When enabled, the ICDM-RX/MOD appends an STX (start of transmission) byte sequence which is configured as **one byte** or **two bytes** to the beginning of the serial packet before it is sent. The length indicates the number of STX bytes, values for length are: • **none** - Disables this function. • **one byte** - Inserts one STX byte before the data. • **two bytes** - Inserts two STX bytes before the data. **Byte 1** - Specifies the transmit character associated with the first STX byte, if the length is **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the transmit character associated with the second STX byte, if the length is **two bytes**. You can specify a value between 0 and 255 in decimal format.
ETX Tx Append	When enabled, the ICDM-RX/MOD appends an ETX (end of transmission) byte sequence which is configured as **one byte** or **two bytes** to the end of the serial packet before it is sent. The length indicates the number of ETX bytes, valid values for length are: • **none** - Disables this function. • **one byte** - Inserts one ETX byte at the end of the data. • **two bytes** - Inserts two ETX bytes at the end of the data. **Byte 1** - Specifies the transmit character associated with the first ETX byte, if the length is set to **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the transmit character associated with the second ETX byte, if the length is **two bytes**. You can specify a value between 0 and 255 in decimal format.
Strip Rx STX/ETX	When you select this check box, the ICDM-RX/MOD strips STX/ETX characters from received serial packets. Clear the check box when you do not want the ICDM-RX/MOD to strip STX/ETX characters from received serial packets. Serial Packets sent from the PLC or application to the ICDM-RX/MOD (over Ethernet), and then sent out the serial port, are not checked for STX/ETX. No STX/ETX character stripping occurs in these serial packets, and framing/parity/overrun error checking does not apply.

PEPPERL+FUCHS

4.4. Port Modbus Configuration (Raw/ASCII Only) Page

Access the *Port Modbus Configuration (Raw/ASCII Only)* page by clicking **Serial | Port | Modbus Settings**.

The following table also provides supporting information for the *Device Modbus Configuration* page that is found under **Ethernet | Device | Modbus/TCP Settings**.

Note: *(Raw-Data Only) displays on the serial version of the Modbus configuration pages.*

Port Modbus Configuration (Raw/ASCII Only) Page	
Serial Modbus Master and Modbus/TCP Interface Settings	
Rx (To PLC) Transfer Mode	Specifies the Modbus/TCP raw-data receive data transfer mode to the PLC. There are three possible settings. • **Slave (PLC Polls)** – The PLC polls the ICDM-RX/MOD for received data by sending read requests on a continual basis. • **Master (Write to PLC)** – The ICDM-RX/MOD writes received data into the specified PLC address using write messages. • **Off** – Received data is not sent to the PLC. The default is Slave Mode. *Note: Slave (PLC Polls) must be selected to interface to serial Modbus masters.*

Port Modbus Configuration (Raw/ASCII Only) Page (Continued)	
Tx (From PLC) Transfer Mode	Specifies the Modbus/TCP raw-data transmit data transfer mode to the PLC. There are three possible settings. • **Slave (PLC Writes)** – The PLC sends write messages to transmit data. A message received with an incremented sequence number indicates new data to send, unless the **Disable Tx Sequence Number Check** option is selected. • **Master (Poll the PLC)** – The ICDM-RX/MOD sends read messages to poll the PLC at the specified address, rate, and message length. A message received with an incremented sequence number indicates new data to send, unless the **Disable Tx Sequence Number Check** option is selected. • **Off** – Transmit data is not accepted from the PLC. The default is Slave Mode. *Note: Slave (PLC Writes) must be selected to interface to serial Modbus masters.*
Maximum Rx Data Packet Size	Specifies the maximum acceptable size of a received serial or Ethernet packet. Default is 246 bytes. Maximums, *Slave Receive* mode = 246, *Serial receive master* mode=1518, *Ethernet receive master* mode = 2048.
Oversized Rx Packet Handling	Specifies how to process oversized received packets. • **Truncate** – truncate the packet to the *Maximum Rx Data Packet Size*. • **Drop** – drop the packet. Default = Truncate
Rx MS Byte First	When you select this check box, the ICDM-RX/MOD receives the Most Significant (MS) byte of a WORD first. This check box is clear by default. Clear the check box when you need to receive the Least Significant (LS) byte of a WORD first.
Tx MS Byte First	This check box is clear by default. When you select this check box, ICDM-RX/MOD transmits the Most Significant (MS) byte of a WORD first. Clear the check box when you need to transmit the Least Significant (LS) byte of a WORD first.
Disable Non-Filtered To PLC Rx Queue	If filtering is disabled, only the last message received is sent to the PLC. This box is clear by default.
Disable Tx Sequence Number Check	Controls the transmit sequence number checking. • If selected, the transmit sequence number checking is disabled. All transmit messages are transmitted if the sequence number has been incremented or not. • If not selected, the sequence number is checked and the message only is transmitted if the sequence number has been updated. Default is *Not selected*.
Modbus/TCP Master Rx/Tx Settings	
PLC IP Address	Specifies the PLC IP Address of the PLC for operating in either *Receive* or *Transmit Master* mode.
PLC Device ID	Specifies the PLC Device ID of the PLC for operating in either *Receive* or *Transmit Master* mode. The default is 1.
Disable Sequence Number/Length Header	The data is transmitted without the 4 byte header (sequence number and data length).

PEPPERL+FUCHS

Port Modbus Configuration (Raw/ASCII Only) Page (Continued)	
Master Rx Mode Only	
PLC Rx Data Address	Specifies the PLC address to write the received data at while operating in the *Master Receive Transfer* mode. The data area must be comprised of 16 bit words and large enough to contain the largest possible received message plus two 16 bit words for the sequence and length parameters. The address is base 1. Therefore, if your address scheme starts at zero, you need to add one to your address.
Maximum PLC Update Rate	The maximum rate (or minimum time interval) in milliseconds that messages are sent to the PLC in the *Master Receive Transfer* mode. This setting configures the ICDM-RX/MOD to space the messages to the PLC in order to prevent overrunning of data before the PLC can process it.
Use Maximum Sized Modbus Messages	Controls the size of Modbus/TCP messages used to write to the Modbus/TCP slave device. • If selected, maximum sized Modbus messages of 242 bytes per message. • If not selected, Modbus messages of no more than 200 bytes are used. The default is *Not selected*. *Note:* *This option only takes affect when large messages are received. Selecting this option may decrease the number of messages sent to the Modbus/TCP slave, thereby reducing network traffic and latency. However, not all Modbus/TCP slaves support maximum sized Modbus messages. So, this setting must be tested with the Modbus/TCP slave to ensure operability*
Master Tx Mode Only	
PLC Tx Data Address	Specifies the PLC address to request transmit data messages while operating in the *Master Transmit Transfer* mode. The data area must be comprised of 16 bit words and contain a sequence number, length, and data to transmit. An updated sequence number indicates new data to transmit. Therefore, the length and data must be written to the transmit data area before updating the sequence number. The address is base 1. Therefore, if your address scheme starts at zero, you need to add one to your address.
PLC Tx Poll Rate	Specifies the rate, in milliseconds, that the ICDM-RX/MOD polls the PLC for transmit data. The default is 100ms
PLC TX Poll Message Length (bytes)	Specifies the length, in bytes, of the transmit message the ICDM-RX/MOD requests from the PLC. This must be large enough to contain the largest size data packet plus four bytes for the sequence number and length fields. Any extra bytes received from the PLC are ignored. The default is 250 bytes
Tx Sequence Number Syncing Enable	If the **Tx (From PLC) Transfer Mode** is operating in *Master (Poll the PLC)*; this specifies whether or not to enable synchronizing transmit data messages between the PLC and the ICDM-RX/MOD. This setting is clear by default.
PLC Tx Consumed Sequence Number Address	Specifies the PLC memory address at which the ICDM-RX/MOD writes the transmit consumed sequence number. This memory address must point to a 16-bit word and, like the other address definitions, is base 1. When the Tx Produced Sequence Number (at the **PLC Tx Data Address**) and this consumed sequence number are equal, the ICDM-RX/MOD has transmitted the last message and is ready for the next transmit message.

11/4/20

PEPPERL+FUCHS

4.5. Port Filtering/Data Extraction Configuration (Raw/ASCII Only) Page

Access the *Port Filtering/Data Extraction Configuration (Raw/ASCII Only)* page by clicking **Serial | Port | Filter or Ethernet | Device | Filter**.

The options on this page are the same as the *Device Filtering/Data Extraction Configuration* page that is accessed by clicking **Ethernet | Device | Filtering**.

Port Filtering/Data Extraction Configuration (Raw/ASCII Only)	
To PLC Filter Mode	Defines the filter/data extraction mode to be employed on data to be sent to the PLC. • **Off** • **String (128 char max)** - Raw/ASCII data is filtered up to 128 characters (or bytes) in length. • **RFID (EPCglobal formats)** - RFID data in any of the EPCglobal formats is filtered, the associated parameters are extracted, and the extracted data and RFID tag are sent to the PLC in a specified format. • **Barcode (UPC/EAN formats)** - Barcode data in specified UPC/EAN formats is filtered, the associated parameters are extracted, and the extracted data and barcode are sent to the PLC in a specified format.

11/4/20

PEPPERL+FUCHS

Port Filtering/Data Extraction Configuration (Raw/ASCII Only) (Continued)	
To PLC Filtering Options (RFID Only)	Defines the RFID filtering criteria to the PLC. If an option is enabled, it is used to decide when an RFID tag can be filtered or sent to the PLC. • **Antenna** - Include the antenna number in the filtering criteria. This is data from the RFID reader and not from the RFID tag itself. • **Filter Value** - Include the filter value in the filtering criteria, which is part of the RFID tag data. • **Serial Number** - Include the serial number in the filtering criteria, which is part of the RFID tag data.
To PLC Filtering Options (RFID/Barcode)	Defines the RFID filtering criteria and the barcode filtering criteria to the application. If an option is enabled, it is used to decide when a valid RFID tag or barcode can be filtered or sent to the PLC. • **Company** - Include the company code in the filtering criteria, which is part of the RFID tag or barcode data. • **Product/Location** - Include the product/location code in the filtering criteria, which is part of the RFID tag or barcode data. • **Encoding/Numbering** - Include the encoding/numbering code in the filtering criteria, which is part of the RFID tag or barcode data.
To Application Filter Mode	The filter/data extraction mode to be employed on data to be sent to the application. • **Off** • **String (128 char max)** - Raw/ASCII data is filtered up to 128 characters (or bytes) in length. • **RFID (EPCglobal formats)** - RFID data in any of the EPCglobal formats are filtered, the associated parameters are extracted, and the extracted data and RFID tag are sent to the application in a specified format. • **Barcode (UPC/EAN formats)** - Barcode data in specified UPC/EAN formats is filtered, the associated parameters are extracted, and the extracted data and barcode are sent to the application in a specified format. *Note: The application filter mode can be set independently of the PLC filtering mode. The only exceptions are:* • *If the PLC filter mode is set to **RFID**, the application filter mode cannot be set to **Barcode**.* • *If the PLC filter mode is set to **Barcode**, the application filter mode cannot be set to **RFID**.*
To Application Filtering Options (RFID Only)	Defines the RFID filtering criteria to the application. If an option is enabled, it is used to decide when an RFID tag can be filtered or sent to the PLC. • **Antenna** - Include the antenna number in the filtering criteria. This is data from the RFID reader and not part of the RFID tag. • **Filter Value** - Include the filter value in the filtering criteria, which is part of the RFID tag data. • **Serial Number** - Include the serial number in the filtering criteria, which is part of the RFID tag data.

PEPPERL+FUCHS

Port Filtering/Data Extraction Configuration (Raw/ASCII Only) (Continued)	
To Application Filtering Options (RFID/Barcode)	Defines the barcode filtering criteria and part of the RFID filtering criteria to the application. If an option is enabled, it is used to decide when a valid RFID tag or barcode can be filtered or sent to the application. • **Company** - Include the company code in the filtering criteria, which is part of the RFID tag or barcode data. • **Product/Location** - Include the product/location code in the filtering criteria, which is part of the RFID tag or barcode data. • **Encoding/Numbering** - Include the encoding/numbering code in the filtering criteria, which is part of the RFID tag or barcode data.
RFID Antenna Grouping	This setting is applicable only to RFID filtering and only if the antenna filtering option is enabled. It allows the ICDM-RX/MOD to filter RFID tags based on antenna groupings. The possible groupings are: <table><tr><td>Setting</td><td>Group 1 Antennas</td><td>Group 2 Antennas</td><td>Group 3 Antennas</td><td>Group N Antennas</td></tr><tr><td>None</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>Groups of Twos</td><td>1,2</td><td>3,4</td><td>5,6</td><td>Etc.</td></tr><tr><td>Groups of Threes</td><td>1,2,3</td><td>4,5,6</td><td>7,8,9</td><td>Etc.</td></tr><tr><td>Groups of Fours</td><td>1,2,3,4</td><td>5,6,7,8</td><td>9,10,11,12</td><td>Etc.</td></tr><tr><td>First Two Only</td><td>1,2</td><td>3</td><td>4</td><td>N+1</td></tr><tr><td>First Three Only</td><td>1,2,3</td><td>4</td><td>5</td><td>N+2</td></tr></table>
RFID Reader Interface Type	Defines the expected RFID data format to be used while operating in the RFID filtering mode. Each Reader Interface Type is unique and pertains to the RFID reader manufacturer. If a different RFID reader is to be used and it provides a similar format to any of the RFID readers listed below, it can also be used in the RFID filtering mode. • **Unspecified** - The ICDM-RX/MOD assumes a HEX ASCII format and attempts to locate the antenna number. • **Alien (Text Mode)** - Specifies the Alien RFID reader **Text Mode**. • **Alien (Terse Mode)** - Specifies the Alien RFID reader **Terse Mode**. • **Intermec (Hex ASCII Mode)** - Specifies the Intermec reader returning data in the **Hex ASCII Mode**. See the *ICDM-RX/EN and ICDM-RX/MOD with Modbus/TCP Filtering and Data Extraction Reference Manual* (https://www.pepperl-fuchs.com) for further details.

11/4/20

PEPPERL+FUCHS

Port Filtering/Data Extraction Configuration (Raw/ASCII Only) (Continued)	
Barcode Formats: UPC/EAN Standard 12-14 Digit UPD/EAN Eight Digit	Defines barcode format to be used for both standard and eight digit UPC labels. The term *standard* refers to UPC-A, EAN-13, JAN, and EAN-14 barcodes which all have ten company/product digits. The standard and eight digit formats are selected independently and each operates independently. Barcode filtering/data extraction does not function if no format is selected.

Barcode Formats table:

Format	Numbering Digits	Company Digits	Product Digits	Check Digit
Standard Formats				
None	N/A	N/A	N/A	N/A
Company-5/ Product-5	1-3	5	5	1
Company-6/ Product-4	1-3	6	4	1
Company-7/ Product-3	1-3	7	3	1
Company-8/ Product-2	1-3	8	2	1
Company-9/ Product-1	1-3	9	1	1
Eight Digit Formats				
EAN-8 Number-2/Product 5	2	0	5	1
EAN-8 Number-3/Product 4	3	0	4	1
UPC-E				

See the *ICDM-RX/EN and ICDM-RX/MOD with Modbus/TCP Filtering and Data Extraction Reference Manual* (https://www.pepperl-fuchs.com) for further details.

Filter Age Time (Time filtered after last read)	Defines the time a filter string, RFID tag, or barcode continues to be filtered after the last time it was received. If an entry is received before the **Filter Age Time** has passed, the entry is filtered and the data is not sent to the PLC and/or application. However, if the **Filter Age Time** has passed, it passes filtering and be sent to the PLC and/or application.
Discard Unrecognized Data (RFID/Barcode)	Specifies what to do with unrecognized RFID or barcode data. • **Off** - Sends unrecognized data to the PLC and/or application. • **To-PLC** - Discards unrecognized data to the PLC. Allows sending of unrecognized data to the application. • **To-Application** - Discards unrecognized data to the application. Allows sending of unrecognized data to the PLC. • **To-PLC/Application** - Discards unrecognized data to both the PLC and application.

4.6. Application TCP Configuration (Raw/ASCII Only) Page

Access the *Port Application TCP Configuration (Raw/ASCII Only)* page by clicking **Serial | Port | Application Interface**.

The options on the *Device Application TCP Configuration* page that is accessed by clicking **Ethernet | Device | Application Interface** are the same.

Note: *The following image shows both Serial device and Ethernet device connections to an application.*

Port Application TCP Configuration (Raw/ASCII Only) Page	
Enable	Enables/disables the *Application Socket Interface*. Enabling this function allows an application to be connected to the device serial/socket port. If both the PLC and application are connected to the device serial/socket port, both can transmit to and receive data from the serial/socket port. However, the PLC and application cannot communicate directly to each other.
Listen	Enabling this setting allows the application to connect to the ICDM-RX/MOD via an Ethernet TCP/IP socket. • **Not selected** - Disables listening and the ICDM-RX/MOD does not accept connection attempts. • **Selected** - Enables listening and the ICDM-RX/MOD accepts connection attempts from the specified **Listen Port**.

PEPPERL+FUCHS

Port Application TCP Configuration (Raw/ASCII Only) Page (Continued)	
Listen Port	The socket port number on the ICDM-RX/MOD the application connects to if the **Application Listen Enable** is selected.
Connect To Mode	Specifies if and how the ICDM-RX/MOD attempts to connect to the application at the specified **Connect IP Address** and **Connect Port**. • **Never** - The ICDM-RX/MOD does not attempt to connect to the application. • **Connect-Always** - The ICDM-RX/MOD attempts to connect to the application until a connection is made. • **Connect-On-Data** – The ICDM-RX/MOD does not attempt to connect to the application until there is data to send to the application. Once data is received from the serial/socket device, the ICDM-RX/MOD attempts to connect to the application until a connection is made.
Connect Port	The application socket port number the ICDM-RX/MOD connects to if the **Application Connect To Mode** is set to either **Connect-Always** or **Connect-On-Data**.
Connect IP Address	The application IP address the ICDM-RX/MOD connects to if the **Application Connect To Mode** is set to either **Connect-Always** or **Connect-On-Data**.
Disconnect Mode	Controls if and how the ICDM-RX/MOD disconnects from an application. • **Never** – The ICDM-RX/MOD does not disconnect from the application. • **Idle** - The ICDM-RX/MOD disconnects when there has been no transmit or received data between the serial/socket device and application for a specified **Idle Timer** period.
Idle Timer	The idle timeout period in milliseconds that is used if the application **Disconnect Mode** is set to **Idle**.
Application Rx Packet ID Settings	
Rx Timeout Between Packets	Specifies the following information, once the start of the packet is received: • How long the ICDM-RX/MOD should wait (in milliseconds) before timing-out, if the ETX Rx Detect length is one byte or two bytes and the ETX byte(s) are not received. • The time to wait in milliseconds between Ethernet packets if the ETX Rx Detect length is set to none.
STX (Start of Transmission) Rx Detect	When enabled, the ICDM-RX/MOD detects an STX (start of transmission) byte sequence which is configured as **one byte** or **two bytes** when it receives an Ethernet packet. The length indicates the number of STX bytes, valid values for length are: • **none** - Disables this function and the ICDM-RX/MOD accepts the first byte received after the last ETX byte(s) as the start of the next Ethernet packet. • **one byte** - Scans Ethernet data for one STX byte and when the ICDM-RX/MOD finds an STX byte it collects the data. If the first byte is not the STX byte, it discards the byte. The ICDM-RX/MOD continues to discard the bytes until it finds an STX byte. • **two bytes** - Scans Ethernet data for two STX bytes and when the ICDM-RX/MOD finds two STX bytes it collects the data. If the STX bytes cannot be found, it discards the bytes. The ICDM-RX/MOD continues to discard the bytes until it finds the two STX bytes. **Byte 1** - Specifies the character that represents the first STX byte. The ICDM-RX/MOD looks for this character in the first STX byte, if the length is **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the character that represents the second STX byte. The ICDM-RX/MOD looks for this character in the second STX byte, only if the length is two bytes. You can specify a value between 0 and 255 in decimal format.

Port Application TCP Configuration (Raw/ASCII Only) Page (Continued)	
ETX (End of Transmission) Rx Detect	When enabled, the ICDM-RX/MOD detects an ETX (end of transmission) byte sequence that is configured as one byte or two bytes marking the end of the Ethernet packet. The length indicates the number of ETX bytes, valid values for length are: • **none** - Disables this function and the ICDM-RX/MOD uses the Rx Timeout Between Packets to indicate the end of data packet. • **one byte** - Scans Ethernet data for one ETX byte and when the ICDM-RX/MOD finds the ETX byte, it identifies the data as an Ethernet packet. • **two bytes** - Scans Ethernet data for two ETX bytes and when the ICDM-RX/MOD finds the ETX bytes, it identifies the data as an Ethernet packet. **Byte 1** - Specifies the character to scan for in the first ETX byte, if the length is one byte or two bytes. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the character to scan for in the second ETX byte, if the length is two bytes. You can specify a value between 0 and 255 in decimal format.

11/4/20

PEPPERL+FUCHS

5. Ethernet Menus

The following Ethernet menus are discussed in this chapter.

- *Ethernet Device Overview Page* on Page 55
- *Device Interface Configuration Page* on Page 56
- *Device Raw/ASCII Configuration Page* on Page 58
- *Device Modbus Configuration Page* on Page 61
- *Device Filtering/Data Extraction Configuration Page* on Page 64
- *Application TCP Configuration* on Page 65

5.1. Ethernet Device Overview Page

When you select the Ethernet menu, the *Ethernet Device Overview* page displays, which displays the configured Ethernet settings.

PEPPERL+FUCHS

5.2. Device Interface Configuration Page

Access the *Device Interface Configuration* page by clicking **Ethernet | Device | Socket Connection**.

The following table provides information about the options on this page.

Device Interface Configuration Page	
Device Name	A user definable string used to describe the serial interface. Valid characters include a-z, A-Z, 0-9, underscores, spaces and dashes. All other characters are discarded. Up to 80 character ASCII string. The default is blank.
Enable	This setting enables/disables the *Device Ethernet Device*. Enabling this function allows an Ethernet TCP/IP device to be connected to a PLC and/or application. If both the PLC and application are connected to the device, both can transmit to and receive data from the device socket port. However, the PLC and application cannot communicate directly to each other.

PEPPERL+FUCHS

Device Interface Configuration Page (Continued)	
Listen	Enabling this setting allows the device to connect to the ICDM-RX/MOD via an Ethernet TCP/IP socket. • **Not selected** - Disables listening; the ICDM-RX/MOD does not accept connection attempts. • **Selected** - Enables listening; the ICDM-RX/MOD accepts connection attempts from the specified **Listen Port**.
Listen Port	This is the socket port number on the ICDM-RX/MOD the application connects to if the **Device Listen Enable** is selected.
Connect To Mode	This setting specifies if and how the ICDM-RX/MOD attempts to connect to the device at the specified **Connect IP Address** and **Connect Port**. • **Never** - The ICDM-RX/MOD does not attempt to connect to the device. • **Connect-Always** - The ICDM-RX/MOD attempts to connect to the device until a connection is made. • **Connect-On-Data** - The ICDM-RX/MOD does not attempt to connect to the device until there is data to send to the device. Once data is received for the device, the ICDM-RX/MOD attempts to connect to the device until a connection is made.
Connect Port	The device socket port number the ICDM-RX/MOD connects to if the **Device Connect To Mode** is set to either **Connect-Always** or **Connect-On-Data**.
Connect IP Address	The device IP address the ICDM-RX/MOD connects to if the **Device Connect To Mode** is set to either **Connect-Always** or **Connect-On-Data**.
Disconnect Mode	This setting specifies if and how the ICDM-RX/MOD disconnects from the device. • **Never** - The ICDM-RX/MOD does not disconnect from the device. • **Idle** - The ICDM-RX/MOD disconnects when there has been no transmit or received data between the device and PLC/application for a specified Idle Timer period.
Idle Timer	The idle timeout period in milliseconds that is used if the **Device Disconnect Mode** is set to **Idle**.
Rx Timeout Between Packets	Specifies the following information, once the start of a packet is received: • How long the ICDM-RX/MOD should wait (in milliseconds) before timing-out, if the ETX Rx Detect length is one byte or two bytes and the ETX byte(s) are not received. • The time to wait in milliseconds between Ethernet packets if the ETX Rx Detect length is set to none.

PEPPERL+FUCHS

5.3. Device Raw/ASCII Configuration Page

Access the *Device Raw/ASCII Configuration* page by clicking **Ethernet | Device | Raw ASCII Settings**.
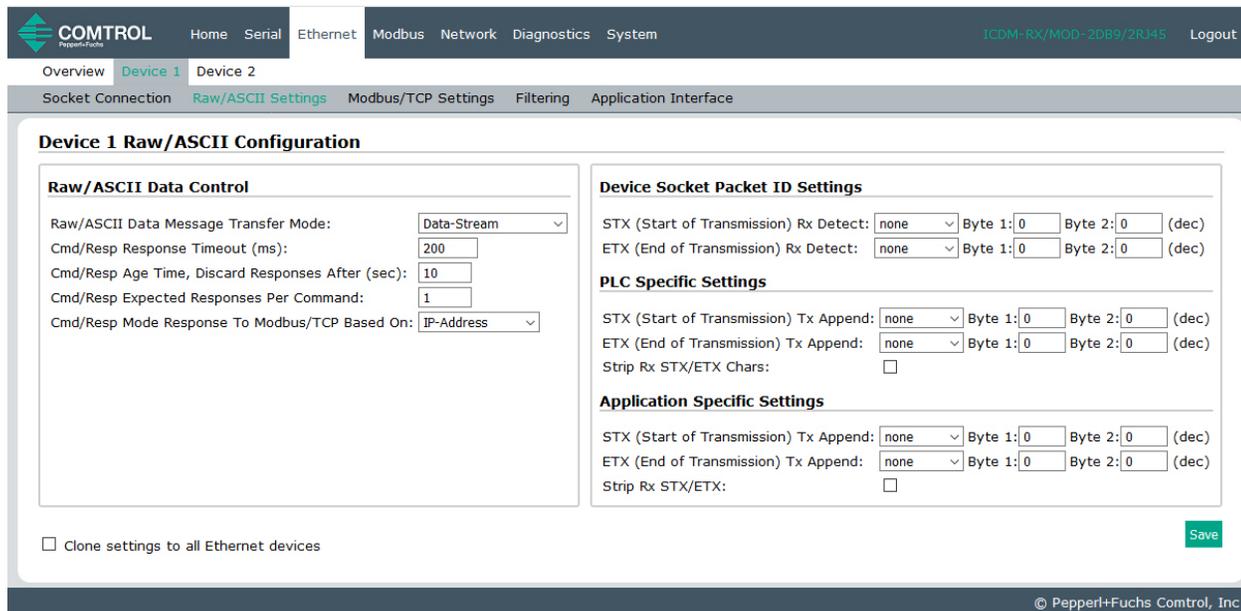
The following table discusses options on this page.

Device Raw/ASCII Configuration Page	
Raw/ASCII Data Control	
Raw-Data Message Transfer Mode	If you select **Data-Stream** (default), the serial port operates in Data-Stream mode, see *Data-Stream Transfer Mode* on Page 16. If you select **Command/Response**, the serial port operates in Command/Response mode, see *Command/Response Mode* on Page 17.
Cmd/Resp Response Timeout	The **Command/Response** mode response timeout setting. The ICDM-RX/MOD waits for the response(s) until this time has elapsed before transmitting another message. The default is 200 msec.
Cmd/Resp Age Time, Discard Responses After (seconds)	The Age Time, or elapsed time, when responses destined for Modbus interface(s) are discarded. This typically occurs when a Modbus controller stops communicating and leaves a response in the response queues. The default is 10 seconds.
Cmd/Resp Expected Responses Per Command	The number of responses expected for each transmitted command. The default is 1.

PEPPERL+FUCHS

Device Raw/ASCII Configuration Page (Continued)	
Cmd/Resp Response To Modbus/TCP Based On	Method for returning responses to Modbus/TCP interfaces. • If you select **IP-Address** (default), all responses are returned to the IP-Address the original command was received from. This may the same or different TCP connection. • If you select **TCP-Connection**, all responses are returned to the same TCP connection the original command was received from. *Note:* *TCP-Connection is typically required when multiple controllers are communicating from the same PLC or computer.*
Device Socket Packet ID Settings	
STX (Start of Transmission) Rx Detect	When enabled, the ICDM-RX/MOD detects an STX (start of transmission) byte sequence which is configured as **one byte** or **two bytes** when it receives an Ethernet packet. The length indicates the number of STX bytes, valid values for length are: • **none** - Disables this function and the ICDM-RX/MOD accepts the first byte received after the last ETX byte(s) as the start of the next Ethernet packet. • **one byte** - Scans Ethernet data for one STX byte and when the ICDM-RX/MOD finds an STX byte it collects the data. If the first byte is not the STX byte, it discards the byte. The ICDM-RX/MOD continues to discard the bytes until it finds an STX byte. • **two bytes** - Scans Ethernet data for two STX bytes and when the ICDM-RX/MOD finds two STX bytes it collects the data. If the STX bytes cannot be found, it discards the bytes. The ICDM-RX/MOD continues to discard the bytes until it finds the two STX bytes. **Byte 1** - Specifies the character that represents the first STX byte. The ICDM-RX/MOD looks for this character in the first STX byte, if the length is **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the character that represents the second STX byte. The ICDM-RX/MOD looks for this character in the second STX byte, only if the length is two bytes. You can specify a value between 0 and 255 in decimal format.
ETX (End of Transmission) Rx Detect	When enabled, the ICDM-RX/MOD detects an ETX (end of transmission) byte sequence that is configured as one byte or two bytes marking the end of the Ethernet packet. The length indicates the number of ETX bytes, valid values for length are: • **none** - Disables this function and the ICDM-RX/MOD uses the Rx Timeout Between Packets to indicate the end of data packet. • **one byte** - Scans Ethernet data for one ETX byte and when the ICDM-RX/MOD finds the ETX byte, it identifies the data as an Ethernet packet. • **two bytes** - Scans Ethernet data for two ETX bytes and when the ICDM-RX/MOD finds the ETX bytes, it identifies the data as an Ethernet packet. **Byte 1** - Specifies the character to scan for in the first ETX byte, if the length is one byte or two bytes. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the character to scan for in the second ETX byte, if the length is two bytes. You can specify a value between 0 and 255 in decimal format.

Device Raw/ASCII Configuration Page (Continued)
PLC Specific Settings and Application Specific Settings

STX Tx Append	When enabled, the ICDM-RX/MOD appends an STX (start of transmission) byte sequence which is configured as one byte or two bytes to the beginning of the Ethernet packet before it is sent. The length indicates the number of STX bytes, values for length are: • **none** - Disables this function. • **one byte** - Inserts one STX byte before the data. • **two bytes** - Inserts two STX bytes before the data. **Byte 1** - Specifies the transmit character associated with the first STX byte, if the length is **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the transmit character associated with the second STX byte, if the length is **two bytes**. You can specify a value between 0 and 255 in decimal format.
ETX Tx Append	When enabled, the ICDM-RX/MOD appends an ETX (end of transmission) byte sequence which is configured as **one byte** or **two bytes** to the end of the Ethernet packet before it is sent. The length indicates the number of ETX bytes, valid values for length are: • **none** - Disables this function. • **one byte** - Inserts one ETX byte at the end of the data. • **two bytes** - Inserts two ETX bytes at the end of the data. **Byte 1** - Specifies the transmit character associated with the first ETX byte, if the length is set to **one byte** or **two bytes**. You can specify a value between 0 and 255 in decimal format. **Byte 2** - Specifies the transmit character associated with the second ETX byte, if the length is **two bytes**. You can specify a value between 0 and 255 in decimal format.
Strip Rx STX/ETX	When you select this check box, the ICDM-RX/MOD strips STX/ETX characters from received Ethernet packets. Clear the check box when you do not want the ICDM-RX/MOD to strip STX/ETX characters from received Ethernet packets. Packets sent from the PLC to the ICDM-RX/MOD (over Ethernet), and then sent out the Ethernet port, are not checked for STX/ETX. No STX/ETX character stripping occurs in these Ethernet packets.

11/4/20

PEPPERL+FUCHS

5.4. Device Modbus Configuration Page

Access the *Device Modbus Configuration* page by clicking **Ethernet | Device | Modbus Settings**.

The following table provides information about the options on this page.

Device Modbus Configuration Page	
Serial Modbus Master and Modbus/TCP Interface Settings	
Rx (To PLC) Transfer Mode	Specifies the Modbus/TCP raw-data receive data transfer mode to the PLC. There are three possible settings. • **Slave (PLC Polls)** – The PLC polls the ICDM-RX/MOD for received data by sending read requests on a continual basis. • **Master (Write to PLC)** – The ICDM-RX/MOD writes received data into the specified PLC address using write messages. • **Off** – Received data is not sent to the PLC. The default is Slave Mode. *Note: Slave (PLC Polls) must be selected to interface to serial Modbus masters.*

PEPPERL+FUCHS

Device Modbus Configuration Page (Continued)	
Tx (From PLC) Transfer Mode	Specifies the Modbus/TCP raw-data transmit data transfer mode to the PLC. There are three possible settings. • **Slave (PLC Writes)** – The PLC sends write messages to transmit data. A message received with an incremented sequence number indicates new data to send, unless the **Disable Tx Sequence Number Check** option is selected. • **Master (Poll the PLC)** – The ICDM-RX/MOD sends read messages to poll the PLC at the specified address, rate, and message length. A message received with an incremented sequence number indicates new data to send, unless the **Disable Tx Sequence Number Check** option is selected. • **Off** – Transmit data is not accepted from the PLC. The default is Slave Mode. *Note:* *Slave (PLC Writes) must be selected to interface to serial Modbus masters.*
Maximum Rx Data Packet Size	Specifies the maximum acceptable size of a received serial or Ethernet packet. Default is 246 bytes. Maximums, *Slave Receive* mode = 246, *Serial receive master* mode=1518, *Ethernet receive master* mode = 2048.
Oversize Rx Packet Handling	Specifies how to process oversized received packets. • **Truncate** – truncate the packet to the *Maximum Rx Data Packet Size*. • **Drop** – drop the packet. Default = Truncate
Rx MS Byte First	When you select this check box, the ICDM-RX/MOD receives the Most Significant (MS) byte of a WORD first. This check box is clear by default. Clear the check box when you need to receive the Least Significant (LS) byte of a WORD first.
Tx MS Byte First	This check box is clear by default. When you select this check box, ICDM-RX/MOD transmits the Most Significant (MS) byte of a WORD first. Clear the check box when you need to transmit the Least Significant (LS) byte of a WORD first.
Disable Non-Filtered To PLC Rx Queue	If filtering is disabled, only the last message received is sent to the PLC. This box is clear by default.
Disable Tx Sequence Number Check	Controls the transmit sequence number checking. • If selected, the transmit sequence number checking is disabled. All transmit messages are transmitted if the sequence number has been incremented or not. • If not selected, the sequence number is checked and the message only is transmitted if the sequence number has been updated. Default is *Not selected*.
Modbus/TCP Master Rx/Tx Settings	
PLC IP Address	Specifies the PLC IP Address of the PLC for operating in either *Receive* or *Transmit Master* mode.
PLC Device ID	Specifies the PLC Device ID of the PLC for operating in either *Receive* or *Transmit Master* mode. The default is 1.
Disable Sequence Number/Length Header	The data is transmitted without the 4 byte header (sequence number and data length).

11/4/20

PEPPERL+FUCHS

Device Modbus Configuration Page (Continued)	
Master Tx Mode Only	
PLC Rx Data Address	Specifies the PLC address to write the received data at while operating in the *Master Receive Transfer* mode. The data area must be comprised of 16 bit words and large enough to contain the largest possible received message plus two 16 bit words for the sequence and length parameters. The address is base 1. Therefore, if your address scheme starts at zero, you need to add one to your address.
Maximum PLC Update Rate	The maximum rate (or minimum time interval) in milliseconds that messages are sent to the PLC in the *Master Receive Transfer* mode. This setting configures the ICDM-RX/MOD to space the messages to the PLC in order to prevent overrunning of data before the PLC can process it.
Use Maximum Sized Modbus Messages	Controls the size of Modbus/TCP messages used to write to the Modbus/TCP slave device. • If selected, maximum sized Modbus messages of 242 bytes per message. • If not selected, Modbus messages of no more than 200 bytes are used. The default is *Not selected*. *Note:* *This option only takes affect when large messages are received. Selecting this option may decrease the number of messages sent to the Modbus/TCP slave, thereby reducing network traffic and latency. However, not all Modbus/TCP slaves support maximum sized Modbus messages. So, this setting must be tested with the Modbus/TCP slave to ensure operability*
Master Tx Mode Only	
PLC Tx Data Address (Base 1)	Specifies the PLC address to request transmit data messages while operating in the *Master Transmit Transfer* mode. The data area must be comprised of 16 bit words and contain a sequence number, length, and data to transmit. An updated sequence number indicates new data to transmit. Therefore, the length and data must be written to the transmit data area before updating the sequence number. The address is base 1. Therefore, if your address scheme starts at zero, you need to add one to your address.
PLC Tx Poll Rate (msec)	Specifies the rate, in milliseconds, that the ICDM-RX/MOD polls the PLC for transmit data. The default is 100ms
PLC TX Poll Message Length (bytes)	Specifies the length, in bytes, of the transmit message the ICDM-RX/MOD requests from the PLC. This must be large enough to contain the largest size data packet plus four bytes for the sequence number and length fields. Any extra bytes received from the PLC are ignored. The default is 250 bytes
Tx Sequence Number Syncing Enable	If the **Tx (From PLC) Transfer Mode** is operating in *Master (Poll the PLC)*; this specifies whether or not to enable synchronizing transmit data messages between the PLC and the ICDM-RX/MOD. This setting is clear by default.
PLC Tx Consumed Seq Nbr Address (Base 1)	Specifies the PLC memory address at which the ICDM-RX/MOD writes the transmit consumed sequence number. This memory address must point to a 16-bit word and, like the other address definitions, is base 1. When the Tx Produced Sequence Number (at the **PLC Tx Data Address**) and this consumed sequence number are equal, the ICDM-RX/MOD has transmitted the last message and is ready for the next transmit message.

11/4/20

PEPPERL+FUCHS

5.5. Device Filtering/Data Extraction Configuration Page

Access the *Device Filtering/Data Extraction Configuration* page by clicking **Ethernet | Device | Filtering**.

The options on this page have the same meaning as the *Port Filtering/Data Extraction Configuration* page accessed by clicking **Serial | Port | Filtering**.

See *Port Filtering/Data Extraction Configuration (Raw/ASCII Only) Page* on Page 48 for information about these options.

PEPPERL+FUCHS

11/4/20

5.6. Application TCP Configuration

Access the *Device Application TCP Configuration* page by clicking **Ethernet | Device | Application Interface**.

The options on this page have the same meaning as the *Port Application TCP Configuration* page accessed by clicking **Serial | Port | Application Interface**.

See *Application TCP Configuration (Raw/ASCII Only) Page* on Page 52 for information about these options.

6. Modbus Menus

This chapter discusses the following Modbus web pages:

- *Known Modbus Slave Devices List Page*
- *Alias Device ID Functionality Overview* on Page 67
- *Modbus Alias Statistics Page* on Page 69
- *Modbus Alias Configuration Page* on Page 70

6.1. Known Modbus Slave Devices List Page

You can access the *Known Modbus Slave Device List* page by clicking **Modbus | Modbus Devices**.

Use the *Known Modbus Slave Device List* page to monitor any Modbus slave devices connected to the ICDM-RX/MOD. This page is updated every 20 seconds.

Known Modbus Slave Device List

Reset Statistics

Port 1 Raw Data:
N/A

Port 2 Modbus/ASCII Slave(s):

Device Id	Active?	Tx Requests	Rx Responses	Timeouts	Invalid Responses	Last Rsp Time	Avg Rsp Time	Min Rsp Time	Max Rsp Time
1	yes	438	438	0	0	0.04 sec	0.04 sec	0.03 sec	0.07 sec
2	yes	429	429	0	0	0.04 sec	0.04 sec	0.03 sec	0.05 sec
3	yes	429	429	0	0	0.03 sec	0.04 sec	0.03 sec	0.08 sec
21	yes	429	429	0	0	0.04 sec	0.04 sec	0.03 sec	0.05 sec
37	yes	429	429	0	0	0.04 sec	0.04 sec	0.03 sec	0.05 sec
43	yes	429	429	0	0	0.03 sec	0.04 sec	0.03 sec	0.05 sec

Port 3 Modbus/RTU Slave(s):

Device Id	Active?	Tx Requests	Rx Responses	Timeouts	Invalid Responses	Last Rsp Time	Avg Rsp Time	Min Rsp Time	Max Rsp Time
16	yes	397	397	0	0	0.04 sec	0.03 sec	0.03 sec	0.04 sec

Port 4 Modbus/RTU Master:
N/A

© Pepperl+Fuchs Comtrol, Inc.

PEPPERL+FUCHS

Known Modbus Slave Device List Page	
Device Id	Unit identifier associated with this device.
Active?	Status of device. • **Yes** means that the last request did not time out. • **No** means that the last request timed out.
Tx Requests	Number of Modbus requests transmitted to this device.
Rx Responses	Number of Modbus responses received from this device.
Timeouts	Number of response timeouts associated with this device.
Last Rsp Time	The last response time from Modbus slave device.
Avg Rsp Time	The average response time from Modbus slave device.
Min Rsp Time	The minimum response time from Modbus slave device.
Max Rsp Time	The maximum response time from Modbus slave device.
Tx Broadcasts	Number of broadcast messages transmitted to this device.
Invalid Responses	Number of invalid responses received from this device.

6.2. Alias Device ID Functionality Overview

One of the most common challenges people face when setting up Modbus systems are problems caused by the limited device ID range. The *Alias Device ID* functionality has been developed to help solve those problems.

The Modbus specification has the following limitations:

• Requires all devices attached to the gateway to be addressed by a device ID.

• Allows only 256 device IDs with a range of 0 to 255.

• Not all device IDs can be used for addressing devices.

  - Device ID 0 is reserved for broadcast messages

  - 1-247 are for device addressing

  - 248 to 255 are reserved for such things as gateway functions. The Modbus/TCP firmware reserves device ID 254 for Ethernet TCP/IP raw/ASCII devices and 255 for serial raw/ASCII devices.

The following are common problems that can occur as a result of the device ID limitations:

• A gateway must route Modbus messages based on the device ID. Therefore, it cannot route to multiple Modbus devices with the same device ID.

• It not always possible or practical to change the device ID of serial Modbus slave devices.

• It is not always possible or practical to modify the device IDs on existing Modbus master programs. This is often true when adding a SCADA system to an existing PLC controlled system.

• A Modbus master with one connection, such as serial PLC, requires connectivity to multiple Modbus slave devices with the same device ID and one or more of the slave devices are connected remotely to different gateways.

Note: *The Modbus Router firmware has been designed to provide network-wide Modbus connectivity for serial Modbus masters.*

The Alias Modbus Device ID functionality allows modification of device IDs only when messages are received from Modbus masters. When configured, a Modbus message from a master with the specified device ID is converted to the alias device ID, the message is then routed internally using the alias device ID. All responses are returned to the master with the original received message device ID.

11/4/20

PEPPERL+FUCHS

The following table demonstrates several device ID aliasing examples:

Received Device ID	Alias Device ID	Routed Message Device ID	Description
1	10	10	Convert messages with received device ID 1 to 10. Route message with device ID 10.
50	5	5	Convert messages with received device ID 50 to 5. Route message with device ID 5.
100	254	254	Convert messages with received device ID 100 to 254. Route message with device ID 254.
10	10	10	Invalid configuration attempt. No change to device ID is performed.

The functionality is described in the following diagram:

ICDM-RX/MOD with Modbus/TCP

Alias Device ID Functionality
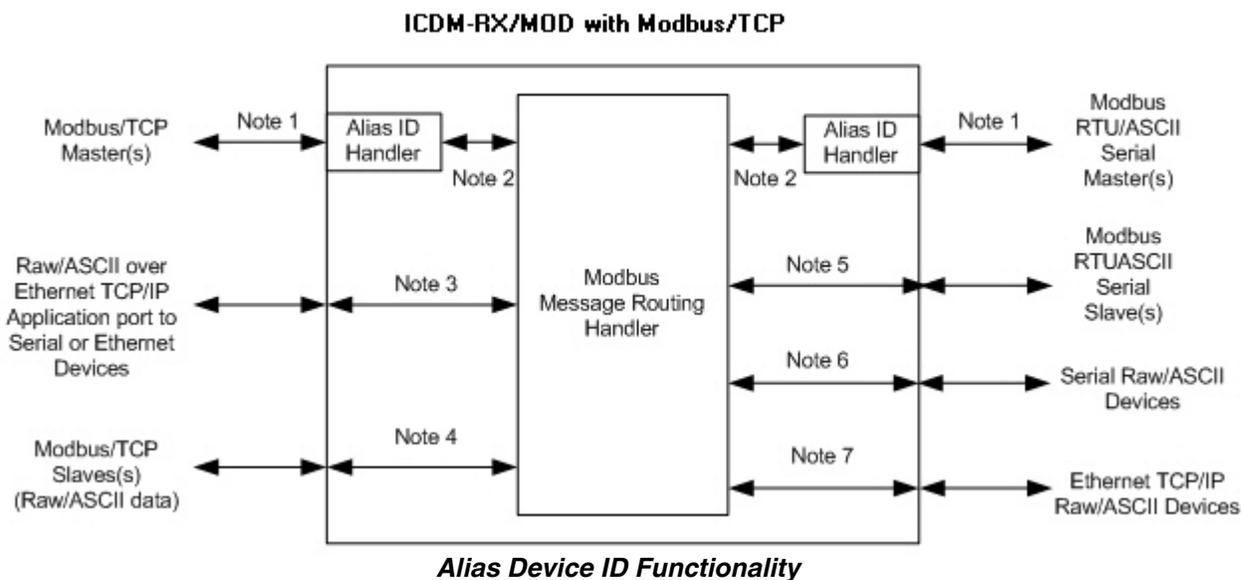
Note 1: These are the originally received Modbus messages. All responses are returned with the original device ID.

Note 2: Modbus messages sent to and responses received from Modbus Message Routing Handler. Depending on the Alias Device ID configuration, these messages may contain the originally received device ID or the alias device ID.

Note 3: The Alias Device ID functionality does not apply to raw/ASCII data received from Ethernet TCP/IP application connections.

Note 4: The Alias Device ID functionality does not apply to Modbus/TCP slaves when a raw/ASCII serial or Ethernet TCP/IP device is set to Master Rx and/or Tx mode. This is when the ICDM-RX/MOD is writing or reading raw/ASCII data from the Modbus/TCP slave device's memory.

Note 5: Modbus messages received from the Modbus Message Routing Handler and sent to Modbus slaves. Depending on the Alias ID configuration, these messages may contain the originally received device ID from the Modbus master or the alias device ID. All responses contain the device ID as received from the Modbus Message Routing Handler.

Note 6: Serial raw/ASCII devices must be addressed with a device ID of 255. The device ID of 255 may be either in the original message or derived from the alias device ID configuration.

Note 7: Ethernet TCP/IP raw/ASCII devices must be addressed with a device ID of 254. The device ID of 254 may be either in the original message or derived from the alias device ID configuration.

PEPPERL+FUCHS

6.3. Modbus Alias Statistics Page

Access the *Modbus Alias Statistics* page by clicking **Modbus | Alias Diagnostics**.

The following table explains the fields in the *Alias Modbus Device ID Configuration/Status* page.

Modbus Alias Statistics Page	
Rx Device ID	The device ID of the received message from a master.
Alias Device ID	The received device ID is converted to the Alias device ID.
Modbus/TCP Master	Alias applied to Modbus/TCP master messages if set to Yes.
Modbus Serial Master	Alias applied to serial Modbus master messages if set to Yes.
Modbus/TCP Count	The number of Alias conversions performed for this configuration to messages received from Modbus/TCP masters.
Modbus Serial Count	The number of Alias conversions performed for this configuration to messages received from serial Modbus masters.

PEPPERL+FUCHS

6.4. Modbus Alias Configuration Page

Access the *Modbus Alias Configuration* page by clicking **Modbus** | **Alias Configuration**.

The following configuration options apply:

Modbus Alias Configuration Page	
Rx Device ID	The device ID (also often called the unit ID) of the received message from a master. The range is from 1 to 247.
Alias Device ID	The alias device ID to convert the received device ID to. These values are permitted: 1-247,254,255.
Modbus/TCP Master	If selected, applies the alias device ID configuration to messages received from Modbus/TCP masters.
Modbus RTU/ASCII Serial Master	If selected, applies the alias device ID configuration to messages received from serial Modbus masters.

11/4/20

PEPPERL+FUCHS

PEPPERL+FUCHS

7. Network Menus

This section discusses the pages under the *Network* menu, which includes:
- *Network Configuration Page* on Page 72
- *Password Page* on Page 74
- *Security Page* on Page 75
- *Key and Certificate Management Page* on Page 76

7.1. Network Configuration Page

You can use the *Network Configuration* page to change the ICDM-RX/MOD network configuration after using PortVision DX for initial network configuration.

Click the **Network tab** to access this page.

11/4/20

PEPPERL+FUCHS

The following table provides information about the **Network Configuration** fields.

Network Configuration Page	
General	
Device Name	You can enter a 16-character **Device Name** to identify this ICDM-RX/MOD on the *Home* page.
TCP Keepalive Default = 60	The TCP protocol has a keepalive feature where the two network stacks periodically ping each other to make sure the connection is still up.
	Upon loss of a TCP/IP connection, the network stack starts a timer. If the TCP/IP connection is still lost after the number of seconds set by the TCP Keepalive value, then the ICDM-RX/MOD closes the connection and frees all the ports associated with the connection.
	If the ICDM-RX/MOD was the originator of the first connection, it will then try to re-connect the TCP/IP connection. This allows the ICDM-RX/MOD to be connected and ready to send/receive data even after a network disturbance.
	For most networks the default value should not need to be changed.
Boot Timeout Default = 15	Allows you to change the bootloader time-out value before the default application, Modbus Router loads.
	You may need to increase this time-out value to 45 for compatibility with spanning tree devices (normally switches). If you change the time-out value to 0, this prevents Modbus Router from loading.
IP Configuration (IPv4)	
Use DHCP	Configures the ICDM-RX/MOD to use DHCPv4 mode.
	If you select **Use DHCP**, the IPv4 Address field below is disabled and set to 0.0.0.0.
	See your System Administrator to acquire a unique reserved IPv4 address if you are using DHCP. They will need the MAC address of the unit to provide you with an IPv4 address.
Use static configuration below	Configures the ICDM-RX/MOD with the static IPv4 address information that you provide in the IPv4 Address, IPv4 Netmask, and IPv4 Gateway fields below.
	The ICDM-RX/MOD is shipped from the factory with the following default IPv4 network settings:
	• IPv4 address = 192.168.250.250
	• IPv4 Netmask = 255.255.0.0
	• IPv4 Gateway address = 192.168.250.1

PEPPERL+FUCHS

7.2. Password Page

You can easily set up a password to secure the ICDM-RX/MOD.

There is no password set from the factory.

Use the following information to configure a password for this ICDM-RX/MOD.

1. If necessary, click **Network | Password**.
2. If changing an existing password, enter that password in the **Old Password** field.
3. Enter a new password.
4. Enter the password in the **Confirm New Password** field.
5. Click the **Save** button.

To log into the ICDM-RX/MOD, you must enter the following:

*   **admin** for the username
*   The configured password for the password

11/4/20

PEPPERL+FUCHS

7.3. Security Page

This table discusses **Security Settings** options.

Security Settings Page	
Enable Secure Config Mode	If **Secure Config** mode is enabled, unencrypted access to administrative and diagnostic functions is disabled. **Secure Config** mode changes ICDM-RX/MOD behavior as follows: • Telnet access to administrative and diagnostic functions is disabled. SSH access is still allowed. • Unencrypted access to the web server via port 80 (http:// URLs) is disabled. Encrypted access to the web server via port 443 (https:// URLs) is still allowed. • Administrative commands that change configuration or operating state which are received using the Pepperl+Fuchs proprietary TCP driver protocol on TCP port 4606 are ignored. • Administrative commands that change configuration or operating state that are received using the Pepperl+Fuchs MAC mode proprietary Ethernet protocol number 0x11FE are ignored.
Enable Telnet/ssh	This option enables or disables the telnet security feature after you click Save and the ICDM-RX/MOD has been rebooted. This option is enabled by default.
Enable SNMP	This option enables or disables the SNMP security feature after you click **Save** and the ICDM-RX/MOD has been rebooted. This option is enabled by default.
Minimum Allowed SSL/TLS Version	You can select the appropriate version for your environment. • SSLv3.0 • TLSv1.0 (default) • TLSv1.1 • TLSv1.2

Use the following steps to change security settings in the ICDM-RX/MOD.

1. Click the **Network | Security**.

2. Click the appropriate check boxes in the *Security Settings* page to enable or disable security accordingly.

3. After making changes, you must click **Save**.

4. You may need to configure security keys or certificates depending on your choices using the next subsection.

11/4/20

PEPPERL+FUCHS

7.4. Key and Certificate Management Page

For secure operation, the ICDM-RX/MOD uses a set of three keys and certificates. These keys and certificates are user configurable.

Note: *All* ICDM-RX/MOD *units are shipped from the factory with identical configurations. They all have the identical self-signed, Pepperl+Fuchs Server RSA Certificates, Server RSA Keys, and Server DH Keys.*

For maximum data and access security, you should configure all ICDM-RX/MOD units with custom certificates and keys.

Key and Certificate Management Page	
	This is a private/public key pair that is used for two purposes:
	• It is used by some cipher suites to encrypt the SSL/TLS handshaking messages. Possession of the private portion of this key pair allows an eavesdropper to both decrypt traffic on SSL/TLS connections that use RSA encryption during handshaking.
RSA Key pair used by SSL and SSH servers	• It is used to sign the Server RSA Certificate in order to verify that the ICDM-RX/MOD is authorized to use the server RSA identity certificate.
	Note: *Possession of the private portion of this key pair allows somebody to pose as the ICDM-RX/MOD.*
	If the Server RSA Key is to be replaced, a corresponding RSA identity certificate must also be generated and uploaded or clients are not able to verify the identity certificate.

11/4/20

PEPPERL+FUCHS

Key and Certificate Management Page (Continued)	
RSA Server Certificate used by SSL servers	This is the RSA identity certificate that the ICDM-RX/MOD uses during SSL/TLS handshaking to identify itself. It is used most frequently by SSL server code in the ICDM-RX/MOD when clients open connections to the ICDM-RX/MOD's secure web server or other secure TCP ports.
	If a ICDM-RX/MOD serial port configuration is set up to open (as a client) a TCP connection to another server device, the ICDM-RX/MOD also uses this certificate to identify itself as an SSL client if requested by the server.
	In order to function properly, this certificate must be signed using the Server RSA Key. This means that the server RSA certificate and server RSA key must be replaced as a pair.
DH Key pair used by SSL servers	This is a private/public key pair that is used by some cipher suites to encrypt the SSL/TLS handshaking messages.
	Note: *Possession of the private portion of the key pair allows an eavesdropper to decrypt traffic on SSL/TLS connections that use DH encryption during handshaking.*

Use the following steps to update security keys and certificates in the ICDM-RX/MOD.

1. Click the **Network | Keys/Cert**.

2. Click **Browse** to locate the key or certificate file, highlight the file, and click **Open**.

3. Click **Upload** when you return to the Key and Certificate Management page.

   The key or certificate notation changes from **factory** or **none** to **User** when the ICDM-RX/MOD is secure.

Note: *You do not need to click **Save**, but changes will not take effect until the ICDM-RX/MOD is rebooted.*

PEPPERL+FUCHS

8. Diagnostic Menus

You can access the following diagnostic and statistics pages from the *Diagnostics* menu:

This section discusses the following pages:

- *Serial Communication Statistics Page* on Page 79
- *Ethernet Device Statistics Page* on Page 82
- *Modbus/TCP and Serial Modbus Master Statistics* on Page 84
- *Serial Interface Logs* on Page 86
- *Ethernet Device Logs* on Page 88
- *System Log* on Page 89

PEPPERL+FUCHS

8.1. Serial Communication Statistics Page

The default *Diagnostics* menu page is the *Serial Communication Statistics* page.

Note: This screen shot is not displaying Application Connection Statistics, which only display if it is configured.

Serial Communications Statistics Page	
Reset Statistics	This button clears the serial port statistics, which sets the value to 0 for all ports.
Serial Interface	
Tx Byte Count	Displays the number of bytes sent out of the serial port.
Tx Pkt Count	Displays the number of serial packets sent out of the serial port.
Rx Byte Count	Displays the number of bytes received over the serial port.
Rx Pkt Count	Displays the number of packets received over the serial port.
Parity Error Count	Displays the number of received serial packets dropped due to parity errors.
Framing Error Count	Displays the number of received serial packets dropped due to framing errors.

PEPPERL+FUCHS

Serial Communications Statistics Page (Continued)	
Overrun Error Count	Displays the number of received serial packets dropped due to overrun error incidents.
To PLC Dropped Packet Count	Displays the number of received serial packets intended for the PLC dropped: • No STX byte(s) found • No ETX byte(s) found • Time-outs • Packet to large • Receive buffer queue overflows
To PLC Truncated Packet Count	Displays the number of received packets that were truncated before being sent to the PLC.
Tx Unexpected Seq Errors	Displays the number of *Unexpected Transmit Sequence Number* errors. The ICDM-RX/MOD increments this number when the ICDM-RX/MOD receives a transmit message with a sequence number that is not equal to either the previous transmit sequence number or the previous transmit sequence number plus one. (If the **Disable Tx Sequence Number Check** option is not selected, the ICDM-RX/MOD expects this sequence number to be incremented by one with each new transmit message.)
Invalid Modbus Message/ Response Count	Displays the number of invalid Modbus To-Master messages or Modbus To-Slaves responses that were received on this port.
Device Timeouts	The number of Command/Response or Modbus To-Slaves messages that timed out waiting for a response.
Cmd/Resp Mode Response Discards	Displays the number of raw-data Command/Response mode responses that were discarded as a result of either: • The connection to the controller was closed. • The response timed out after the Age Time had been reached.
Filtering Statistics	*Note:* *Filtering Statistics are only displayed if filtering is enabled on or more serial ports.* **Filtering Statistics** Valid Data Items Sent to PLC Interface: 0 0 0 0 Valid Data Items Filtered From PLC: 0 0 0 0 Invalid Data Items Discarded From PLC: 0 0 0 0 Valid Data Items Sent to App Interface: 0 0 0 0 Valid Data Items Filtered From App: 0 0 0 0 Invalid Data Items Discarded From App: 0 0 0 0 RFID Tags With Unknown Formats: 0 0 0 0
Valid Data Items Sent To PLC Interface	Displays the number of valid string, RFID, or barcode data sent to the PLC. Applies when filtering is enabled.
Valid Data Items Filtered From PLC	Displays the number of valid string, RFID, or barcode data filtered from (not sent) to the PLC. Applies when filtering is enabled.
Invalid Data Items Discarded From PLC	Displays the number of invalid RFID or barcode data not sent to the PLC. Applies when RFID or barcode filtering is enabled.
Valid Data Items Sent To App Interface	Displays the number of valid string, RFID, or barcode data sent to the application. Applies when filtering is enabled.
Valid Data Items Filtered From App	Displays the number of valid string, RFID, or barcode data filtered from (not sent) to the application. Applies when filtering is enabled.

11/4/20

PEPPERL+FUCHS

Serial Communications Statistics Page (Continued)	
Invalid Data Items Discarded From App	Displays the number of invalid RFID or barcode data not sent to the PLC. Applies when RFID or barcode filtering is enabled.
RFID Tags With Unknown Formats	Data received that was in the general form of 64 or 96 bit RFID tags, but was not in any of the EPCglobal formats. Applies only when RFID filtering is enabled.
Application Connection Statistics	***Note:*** *Application Connection Statistics are only displayed if an application connection is enable on one or more serial ports.* **Application Connection Statistics** TX Byte Count: 0 0 0 0 TX Pkt Count: 0 0 0 0 To Application Dropped Packet Count: 0 0 0 0 RX Byte Count: 0 0 0 0 RX Pkt Count: 0 0 0 0 To Device Dropped Packet Count: 0 0 0 0
TX Byte Count	Displays the number of bytes sent out the application socket port.
TX Pkt Count	Displays the number of packets sent out the application socket port.
To Application Dropped Packet Count	Displays the number of received serial or Ethernet device packets intended for the application dropped: • No STX byte(s) found • No ETX byte(s) found • Time-outs • Packet to large • Receive buffer queue overflows • Application connection is off line
To PLC Truncated Packet Count	Displays the number of received packets that were truncated before being sent to the PLC.
RX Byte Count	Displays the number of bytes received over the application socket port.
RX Pkt Count	Displays the number of packets received over the application socket port.
To Device Dropped Packet Count	Displays the number of dropped packets that were intended for the device.

PEPPERL+FUCHS

8.2. Ethernet Device Statistics Page

Access the *Ethernet Device Statistics* page by clicking **Diagnostics | Ethernet Statistics**.

Ethernet Device Statistics Page	
Reset Statistics	This button clears the socket port statistics, which sets the value to 0 for all ports.
Ethernet Interface	
Remote Connection Status	Displays the connected Ethernet device IP address and port.
Tx Byte Count	Displays the number of bytes sent out the device socket port.
Tx Pkt Count	Displays the number of packets sent out the device socket port.
Rx Byte Count	Displays the number of bytes received over the device socket port.
Rx Pkt Count	Displays the number of packets received over the device socket port.
To PLC Dropped Packet Count	Displays the number of dropped packets that were intended for the PLC.
To PLC Truncated Packet Count	Displays the number of received packets that were truncated before being sent to the PLC.
Tx Unexpected Sequence Errors	Same as the serial port statistics (Page 80).
Cmd/Resp Mode Device Timeouts	The number of Command/Response messages that timed out waiting for a response.

11/4/20

PEPPERL+FUCHS

Ethernet Device Statistics Page (Continued)	
Cmd/Resp Mode Response Discards	Displays the number of Command/Response mode responses that were discarded as a result of either: The connection to the controller was closed. The response timed out after the Age Time had been reached.
Filtering Statistics	***Note:** Filtering Statistics are only displayed if filtering is enabled on one or more Ethernet device interface.*
Valid Data Items Sent To PLC Interface	Displays the number of valid string, RFID, or barcode data sent to the PLC. Applies when filtering is enabled.
Valid Data Items Filtered From PLC	Displays the number of valid string, RFID, or barcode data filtered from (not sent) to the PLC. Applies when filtering is enabled.
Invalid Data Items Discarded From PLC	Displays the number of invalid RFID or barcode data not sent to the PLC. Applies when RFID or barcode filtering is enabled.
Valid Data Items Sent To App Interface	Displays the number of valid string, RFID, or barcode data sent to the application. Applies when filtering is enabled.
Valid Data Items Filtered From App	Displays the number of valid string, RFID, or barcode data filtered from (not sent) to the application. Applies when filtering is enabled.
Invalid Data Items Discarded From Application	Displays the number of invalid RFID or barcode data not sent to the PLC. Applies when RFID or barcode filtering is enabled.
RFID Tags With Unknown Formats	Data received that was in the general form of 64 or 96 bit RFID tags, but was not in any of the EPCglobal formats. Applies only when RFID filtering is enabled.
Application Connection Statistics	***Note:** Application Connection Statistics are only displayed if an Application connection is enabled on or more Ethernet device interfaces.*
TX Byte Count	Displays the number of bytes sent out the application socket port.
TX Pkt Count	Displays the number of packets sent out the application socket port.
To Application Dropped Packet Count	Displays the number of received serial or Ethernet device packets intended for the application dropped: • No STX byte(s) found • No ETX byte(s) found • Time-outs • Packet to large • Receive buffer queue overflows • Application connection is offline
To PLC Truncated Packet Count	Displays the number of received packets that were truncated before being sent to the PLC.
RX Byte Count	Displays the number of bytes received over the application socket port.
RX Pkt Count	Displays the number of packets received over the application socket port.
To Device Dropped Packet Count	Displays the number of dropped packets that were intended for the device.

PEPPERL+FUCHS

8.3. Modbus/TCP and Serial Modbus Master Statistics

Access the *Modbus/TCP and Serial Modbus Master Statistics* page by clicking **Diagnostics | PLC Diagnostics**. It is intended to help with debugging PLC programs, monitoring the PLC interface, and solving configuration problems.

Modbus/TCP and Serial Modbus Master Statistics Page	
Reset Statistics	Selecting this button clears the values displayed on this page.
Modbus/TCP Slave Mode Specific Statistics	
Messages Received From PLC	Displays the total number of raw data and Modbus RTU/ASCII messages received from the PLC.
Responses Sent to PLC	Displays the total number of raw data and Modbus RTU/ASCII responses sent to the PLC.
Raw Serial Port Data Messages Received From PLC	Displays the number of serial port raw data messages received.
Raw Socket Port Data Messages Received From PLC	Displays the number of socket port raw data messages received.

11/4/20

PEPPERL+FUCHS

Modbus/TCP and Serial Modbus Master Statistics Page (Continued)	
Modbus RTU/ASCII Messages Received From PLC	Displays the number of Modbus RTU/ASCII device specific messages received.
Modbus RTU/ASCII Broadcasts Received From PLC	Displays the number of Modbus RTU/ASCII broadcast messages received.
Invalid Command Lengths	Displays the number of messages received with invalid command lengths.
Invalid Message Data Errors	Displays the number of messages received with invalid message data errors. These errors occur when the ICDM-RX/MOD receives a message that cannot be processed due to improper message data.
Unknown Request Destination IDs	Displays the number of messages received with unknown request destination IDs.
Invalid Request Protocol Types	Displays the number of messages received with invalid protocol errors. This occurs when a message is received with a protocol other than the Modbus protocol value of zero.
Unsupported Modbus Function Codes	Displays the number of messages received with unsupported function codes.
Modbus/TCP Master Mode Specific Statistics	
Messages Sent To PLC	Displays the total number of raw data messages sent to the PLC.
Responses Received From PLC	Displays the total number of raw data responses received from the PLC.
Invalid Response Data Errors	Displays the number of response data errors to polling requests returned from the PLC. Possible causes include: • Incorrectly formed transmit data memory format. (Possibly missing the sequence number and/or length fields.) • More data to transmit indicated via the length field than was returned in the message. • Transmit sequence number error. Sequence number increased by more than one. This could indicate an unsent transmit message. • Insufficient polling length configuration. (ICDM-RX/MOD polling length is to small.) • Attempting to transmit to large of message for a single Modbus/TCP message. (More than 246 bytes of data.)
Error responses	Displays the number of responses received from the PLC with errors indicated. This occurs when the PLC returns a response with an error indication. This may be caused by such things as: • Invalid PLC address configuration • Improper PLC configuration
Unexpected Response Function Codes	Displays the number of unexpected response function codes from either a *Master Receive* or *Master Transmit* mode message. This occurs when a response was received without an expected function code.
Unknown Response Destination Ids	Displays the number of responses with unknown destination IDs. This occurs when the PLC returns a response with an unknown destination ID.
Invalid Response Protocol Types	Displays the number of responses with invalid protocol errors. This occurs when a response is returned with a protocol other than the Modbus protocol value of zero.

11/4/20

![PEPPERL+FUCHS]

Modbus/TCP and Serial Modbus Master Statistics Page (Continued)	
Failed Modbus/TCP Connection Attempts	Displays the number of failed Modbus/TCP connection attempts to the specified PLC IP address.
Modbus/TCP Connection Problems	Displays the number of Modbus/TCP connection attempt problems. This occurs when the device responds and the connection is made, but there are problems setting up the connection options. Possible problems include: • Setting the TCP connection to *TCP_NODELAY*. • Setting the socket connection to *SO_OOBINLINE*. • Setting the socket connection to *SO_KEEPALIVE*
No Available Modbus/TCP Connection Errors	Displays the number of connections aborted when there are no available Modbus/TCP connections. This error occurs when the maximum number of 32 Modbus/TCP connections has been reached and the ICDM-RX/MOD is attempting to form another Modbus/TCP connection.
Non-Mode Specific Statistics/Diagnostics	
Oversized Received Data Packet Errors	Displays the number of received serial or Ethernet data packets that were larger than the configured maximum receive data packet.
Improper Configuration Errors	Displays the number of improper configuration errors. These errors occur when the ICDM-RX/MOD receives a message that cannot be performed due to an invalid configuration.
System Resource Errors	Displays the number of system resource errors. These errors indicate a system error on the ICDM-RX/MOD such as an inoperable serial port or a full transmit queue. These errors typically occur when the PLC(s) are sending data to the ICDM-RX/MOD faster than the ICDM-RX/MOD can process it.
Writes to Offline Ethernet Device on Socket N	Displays the number of write attempts by a PLC to the Ethernet device when the device was offline.
First Error Description	Text description of the first error that occurred.
Last Error Description	Text description of the last or most recent error that occurred.

8.4. Serial Interface Logs

Access the *Serial Interface Logs* page by clicking **Diagnostics | Serial Logs**.

The *Serial Interface Logs* page provides a log of received and transmitted serial port messages. Up to 128 bytes per message and up to 128 messages are logged. It is intended to help with debugging serial connectivity problems, determining the proper start and end of transmission bytes, and diagnosing device problems.

The format is as follows:

> **Pkt(n): ddd:hh:mm:ss.ms Tx/Rx:<Data>**

Where:

ddd – days since last system restart

hh – hours since last system restart

ms – minutes since last system restart

ss – seconds since last system restart

mm – milliseconds since last system restart

11/4/20

PEPPERL+FUCHS

<Data> - Data packet received.

- For Modbus slave data, all data bytes shown in hex (xxh) format.
- For Raw/ASCII and Modbus/ASCII data
- ASCII characters displayed as characters
- Non-ASCII displayed in hex (xxh) format

8.5. Ethernet Device Logs

The *Ethernet Device Interface Logs* page is accessed using the Display Ethernet Device Logs option, which provides a log of received and transmitted Ethernet device messages. Up to 128 bytes per message and up to 128 messages are logged. It is intended to help with debugging Ethernet connectivity problems, determining the proper start and end of transmission bytes, and diagnosing device problems.

The format is as follows: **Pkt(n)**: ddd:hh:mm:ss.ms Tx/Rx:<Data>

Where:

ddd – days since last system restart

hh – hours since last system restart

mm – minutes since last system restart

ss – seconds since last system restart

ms – milliseconds since last system restart

<Data> - Data packet received.

- ASCII characters displayed as characters
- Non-ASCII displayed in hex (xxh) format

8.6. System Log

The *System Log* page provides system level information, which is updated every 10 seconds.

- Click the **Refresh** button to view the latest system log information.
- Click the **Clear** button to restart a new system log file.
- Click the **Save Logfile** button to save the system log file, if requested by Technical Support.

9. System Menus

This section discusses the web pages under the **System** menu:

- *Update Firmware* on Page 90
- *Configuration File Page* on Page 91
- *System Snapshot Page* on Page 93
- *Restore Defaults Page* on Page 94
- *Reboot* on Page 94

9.1. Update Firmware

You can upload firmware (Modbus/TCP or Bootloader) using the **System | Update Firmware** page. You must first unpackage the firmware from the **.zip** file.

Note: *Optionally, you can use PortVision DX to upload firmware after unpackaging the **.zip** file.*

Use the following procedure to upload the latest firmware onto the ICDM-RX/MOD.

1. If necessary, download the firmware from https://www.pepperl-fuchs.com.
2. Unzip the file.
3. Open your web browser and enter the IP address of the ICDM-RX/MOD.
4. Click the **System** menu, which opens the *Update Firmware* page.
5. Click the **Browse** button, navigate to the file, select it and click the **Open** button.
6. Click the **Update** button.
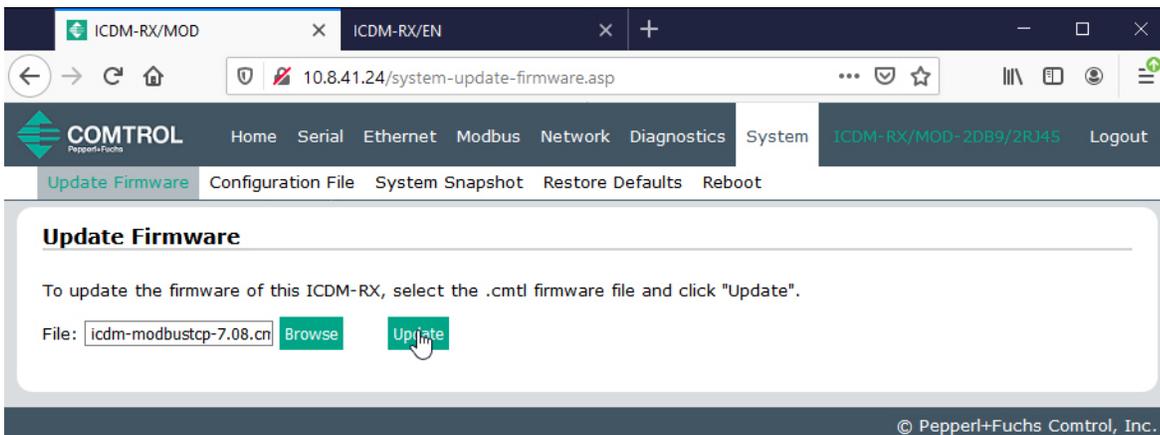
An *Update In Progress* pop up notifies you with the upload duration, not to reset or disconnect the device or to close the page.

11/4/20

PEPPERL+FUCHS

9.2. Configuration File Page

You can use the **Save Configuration** option to save a ICDM-RX/MOD configuration file for recovery purposes or to quickly configure other ICDM-RX/MODs that require the same configuration using the **Load Configuration** option.

Note: *Optionally, you can use PortVision DX to save and load configuration files.*

9.2.1. Saving a Configuration File

You can use this procedure to save a ICDM-RX/MOD configuration file.

1. Click **System** | **Configuration File**.
2. Click the **Save Configuration** button.
3. Save the configuration file following your browser prompts.
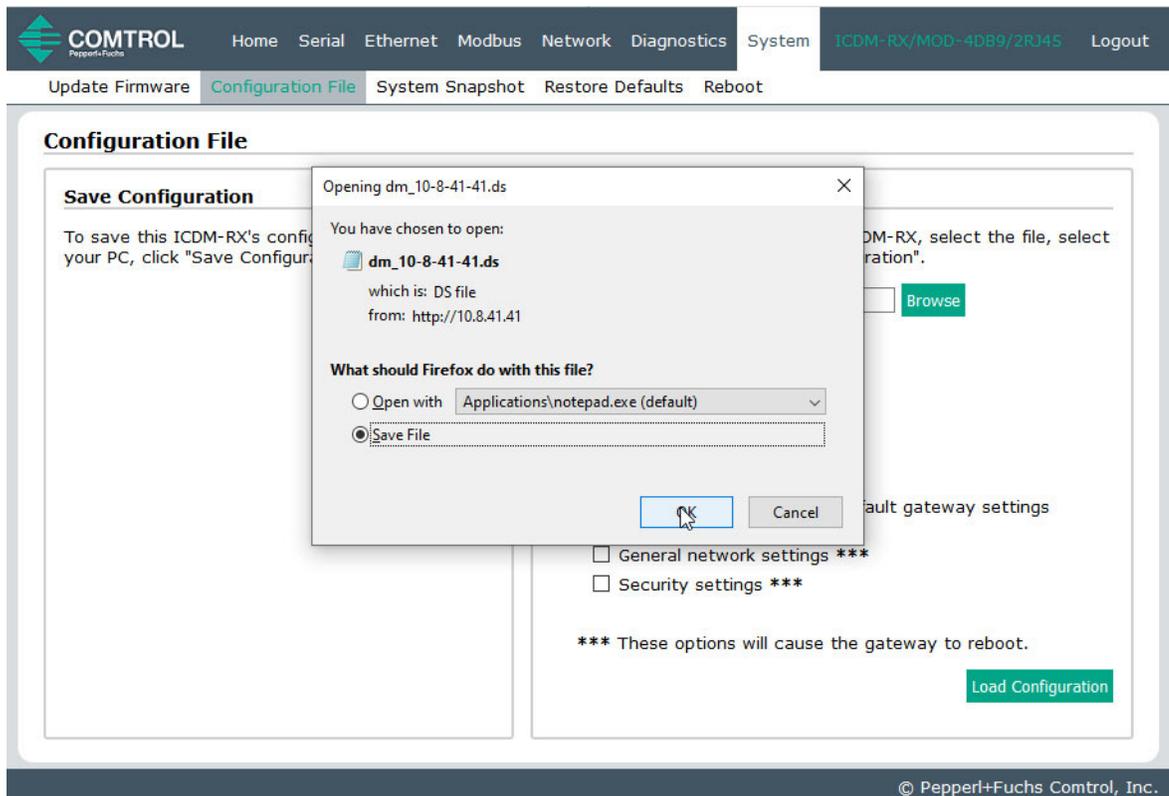
9.2.2. Loading a Configuration File

You can use this procedure to load a previously saved ICDM-RX/MOD configuration file.

1. Click **System** | **Configuration File**.
2. Click the **Browse** button, highlight the configuration file you want to load, and click the **Open** button.
3. Click the **Load Configuration** button.

PEPPERL+FUCHS

9.3. System Snapshot Page

You can use the *System Snapshot* page to download a snapshot of the device status, log, and configuration. You may find the information can help you diagnose a problem with the ICDM-RX/MOD. In addition, this information may be requested by technical support if you have called for assistance.

1. Open your browser and enter the IP address of the ICDM-RX/MOD.
2. Click **System** | **System Snapshot**.
3. Click the **Device Snapshot** button.
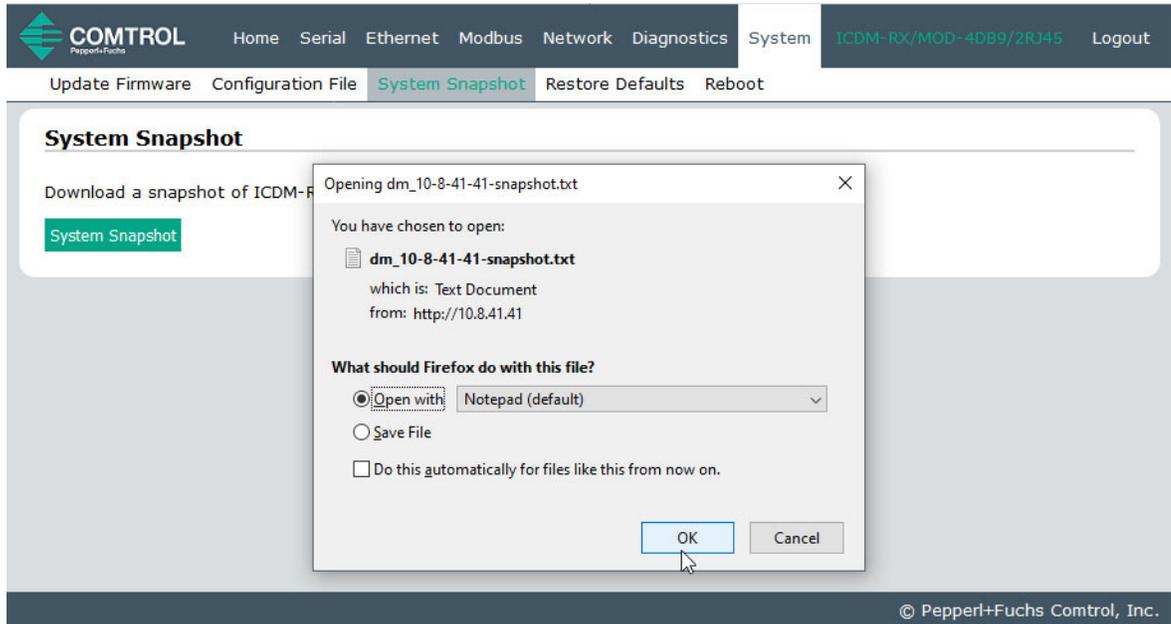
4. Save the file using the method for your browser.

PEPPERL+FUCHS

9.4. Restore Defaults Page

You can easily some or all of your settings to factory defaults by using the procedure below.

1. Open your browser and enter the IP address of the ICDM-RX/MOD.
2. Click **System | Restore Defaults**.
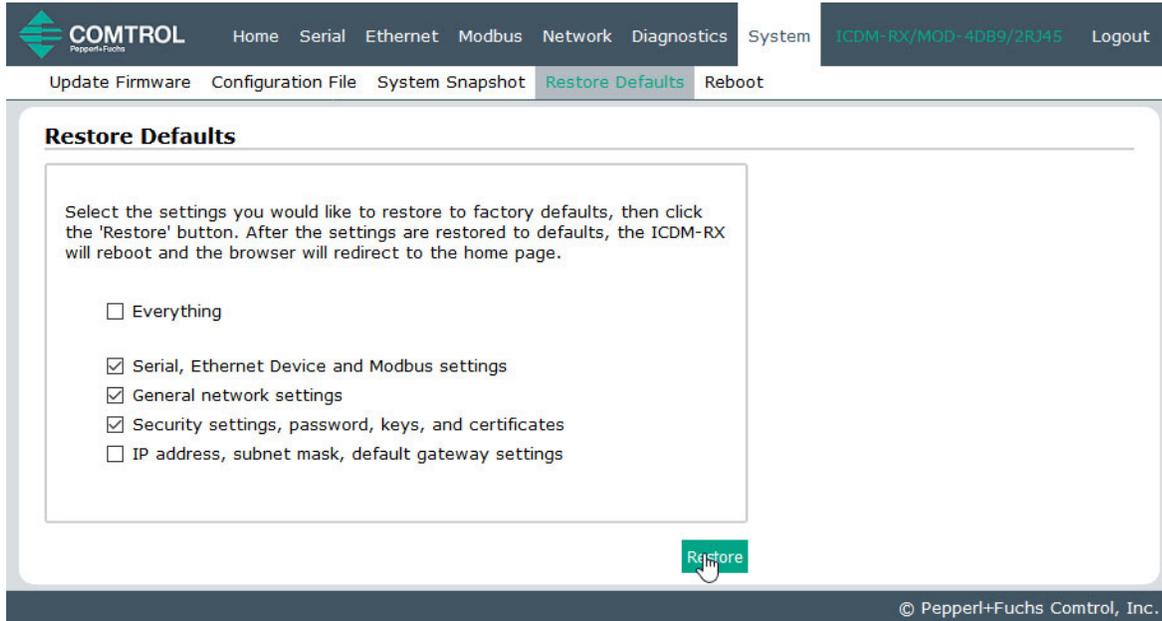
3. Select **Everything** or the specific setting or settings that you want to restore.
4. Click the **Restore** button.
5. The ICDM-RX/MOD reboots and re-opens the web interface.

9.5. Reboot

You can reboot the ICDM-RX/MOD remotely using the *Reboot* web page.

1. Click **System | Reboot**.
2. Click the **Reboot Now** button or wait the 10 seconds for it automatically reboot.

PEPPERL+FUCHS
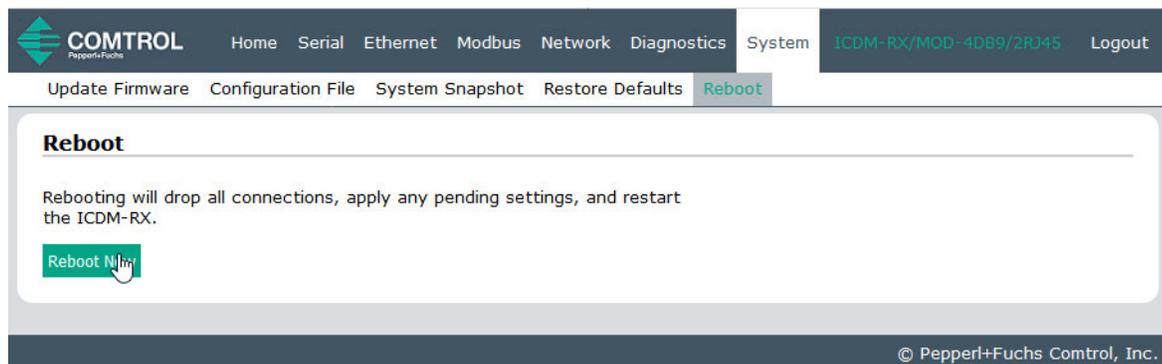
11/4/20

10. Troubleshooting and Technical Support

You should review the *Troubleshooting* chapter in the *ICDM-RX Hardware Installation and Configuration Manual* before calling Technical Support because they will request that you perform many of the procedures or verifications before they can help you diagnose a problem.

- *Troubleshooting Checklist* on Page 95
- *General Troubleshooting* on Page 96

If you cannot diagnose the problem, you can contact *Technical Support* on Page 96.

10.1. Troubleshooting Checklist

The following checklist may help you diagnose your problem:

- Verify that you are using the correct types of cables on the correct connectors and that all cables are connected securely.

    Note: *Most customer problems reported to Pepperl+Fuchs Technical Support are eventually traced to cabling or network problems.*

- Isolate the ICDM-RX/MOD from the network by connecting the device directly to a NIC in a host system.
- Verify that the Ethernet hub and any other network devices between the system and the ICDM-RX/MOD are powered up and operating.
- Reset the power on the ICDM-RX/MOD and watch the **PWR** or **Status** light activity.

PWR or Status LED	Description
5 sec. off, 3 flashes, 5 sec. off, 3 flashes...	RedBoot^{TM} checksum failure.
5 sec. off, 4 flashes, 5 sec. off, 4 flashes...	SREC load failure.

- If the device has a power switch, turn the device's power switch off and on, while watching the LED diagnostics.
- If the ICDM-RX/MOD does not have a power switch, disconnect and reconnect the power cord.
- Verify that the network IP address, subnet mask, and gateway is correct and appropriate for the network. If IP addressing is being used, the system should be able to ping the ICDM-RX/MOD.
- Verify that the IP address programmed into the ICDM-RX/MOD matches the unique reserved IP configured address assigned by the system administrator.
- If using DHCP, the host system needs to provide the subnet mask and gateway.
- Reboot the system and the ICDM-RX/MOD.
- If you have a spare ICDM-RX/MOD, try replacing the device.

11/4/20

PEPPERL+FUCHS

10.2. General Troubleshooting

This table illustrates some general troubleshooting tips.

Note: *Make sure that you have reviewed the Troubleshooting Checklist on Page 95.*

General Condition	Explanation/Action
PWR or **Status** LED flashing	Indicates that boot program has not downloaded to the unit. 1. Reboot the system. 2. Make sure that you have downloaded the most current firmware for your protocol: https://www.pepperl-fuchs.com. If the PWR or Status LED is still flashing, contact Technical Support.
PWR or **Status** LED not lit	Indicates that power has not been applied or there is a hardware failure. Contact Technical Support.
Cannot ping the device through Ethernet hub	Isolate the ICDM-RX/MOD from the network. Connect the device directly to the NIC in the host system (see Page 95).
Cannot ping or connect to the ICDM-RX/MOD	The default IP address is often not accessible due to the subnet masking from another network unless **192.168** is used in the network. In most cases, it is necessary to program in an address that conforms to your network.
ICDM-RX/MOD continuously reboots when connected to some Ethernet switches or routers	Invalid IP information may also cause the switch or router to check for a gateway address. Lack of a gateway address is a common cause.

10.3. Technical Support

The *ICDM-RX Hardware Installation and Configuration Manual* contains troubleshooting procedures that you should perform before contacting Technical Support since they will request that you perform, some or all of the procedures before they will be able to help you diagnose your problem.

Worldwide Headquarters

Pepperl+Fuchs
68307 Mannheim, Germany
+49 621 766-0
info@de.pepperl-fuchs.com

USA Headquarters

Pepperl+Fuchs, Inc.
Twinsburg, Ohio 44087 - USA
+1 330 425 3555
sales@us.pepperl-fuchs.com

Asia Pacific Headquarters

Pepperl+Fuchs Pte Ltd.
Company Registration No. 199003130E
Singapore 139942
+65 67799091
sales@sg.pepperl-fuchs.com

11/4/20

PEPPERL+FUCHS

Appendix A. Programming the PLC via Concept

A.1. Overview

After reviewing *Programming Interface* on Page 12, you can use information in the *Concept Program Screens* subsection on Page 98 to help set up your PLC and program the various messages.

A.1.1. What is Concept?

Concept is the PLC programming software package designed to support the Schneider Electric Momentum, Quantum, and Compact PLCs. It does not support any other Schneider Electric PLC or any other manufacturers PLC. However, the example PLC program and installation process may be helpful when working with other PLCs

A.1.2. Requirements

The following requirements must be met to run the example programs.

- The Modbus/TCP firmware must be installed on the ICDM-RX/MOD and configured as described in the *ICDM-RX Hardware Installation and Configuration Manual*.
- The ICDM-RX/MOD must be installed on the same Ethernet network segment as the PLC.
- Concept must be installed on your computer.
- The instructions in this guide require that you have some familiarity with this programming application.
- A loopback plug is required for the first port on the ICDM-RX/MOD when running an example PLC program. See the *ICDM-RX Hardware Installation and Configuration Manual* for more information if you need to build loopback plugs.

The PLC program examples (*LPBKCNCP* on Page 105 and *SCANCNCP* on Page 105) are optional. You can download the latest program examples from the Internet.

A.1.3. Example Program Considerations (Raw Data)

The example programs are for raw data only.

- While the receive and transmit sequence numbers are cleared on the ICDM-RX/MOD at the start of the program, the only requirement is that the sequence numbers be in sync between the PLC and ICDM-RX/MOD.
- The ICDM-RX/MOD should be reset before starting SCANCNCP example program due to PLC program execution scheduling. If the ICDM-RX/MOD is not reset, the sequence numbers may be out of sync. This may result in receiving outdated serial data as well as an unexpected transmission of serial data. A *Transmit Unexpected Sequence Number* error may also occur.

Statistics retrieval is not included in the example programs, but can be easily added by inserting a request statistics message.

A.2. Concept Program Screens

The following screens are intended to aid the PLC programmer in setting up their PLC and programming the various messages.

A.2.1. Processor and Ethernet Setup

The Processor and Ethernet communications port needs to be set up properly in order for Modbus/TCP to function. It is highly recommended to read and follow your PLC manufacturer's documentation.

The following documents are recommended for the Concept programming software.

- *Concept User Manual* – 840 USE 503 00
- *Concept IEC Block Library Part: Comm* – 840 USE 504 00

In addition to that information, it is recommended that the following settings be made to allow Modbus/TCP to function properly on a Schneider Electric Momentum, Quantum, or Compact PLC.

1. Verify that the correct processor type has been selected.
2. Verify the proper memory is configured to interface to the ICDM-RX/MOD. At least 256 registers must be available.
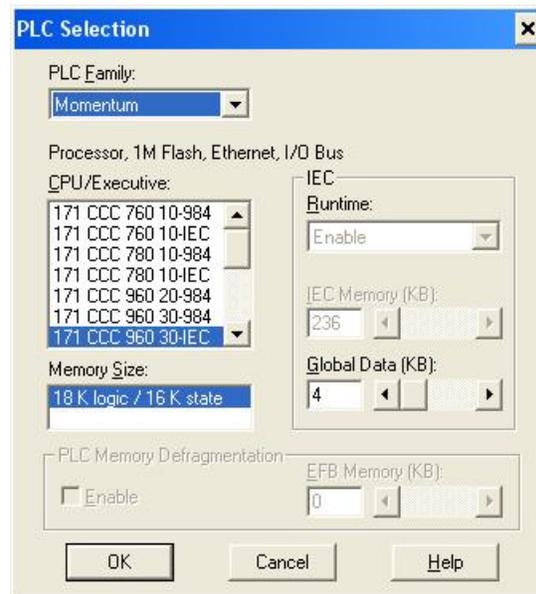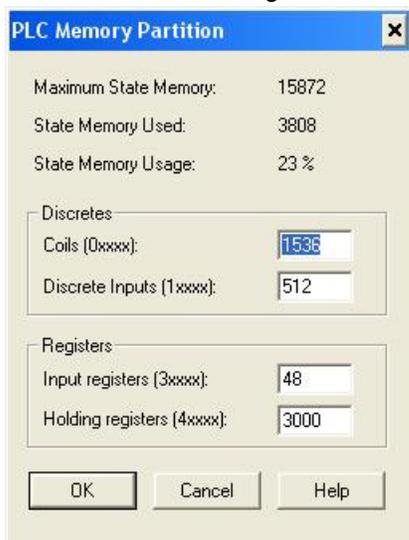
3. Select the proper extension for Modbus/TCP Ethernet. This is generally 1 for Momentum.

PEPPERL+FUCHS
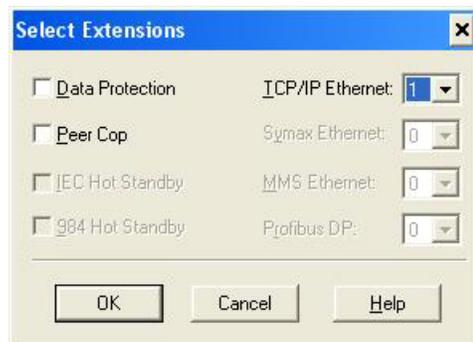
4. Set the Network Configuration. The following is recommended for this screen:
   - Select the **Specify IP Address** option.
   - Set the **Internet Address**.
   - Set the **Subnet Mask**.
   - Set the **Gateway**.
   - Optionally set a diagnostic block.
- In this screen, the I/O Scanner can be configured to directly access the serial port communications on the ICDM-RX/MOD. For more information, see *I/O Scanner (Raw Data)* on Page 22.

A.2.2. Message Screens

The following message screens are discussed in the upcoming subsections.
- *Read Serial Data via Read Holding Registers Message on Page 100*
- *Transmit Serial Data via Write Multiple Registers Message on Page 101*
- *Set Receive Sequence Number via Write Multiple Registers Message on Page 101*
- *Set Transmit Sequence Number via Write Multiple Registers Message on Page 102*
- *Read Serial Port Statistics via Read Holding Registers Message on Page 103*
- *Modbus/TCP Slot/Index and ICDM-RX/MOD IP Address Definition on Page 104*

11/4/20

PEPPERL+FUCHS

A.2.2.1. Read Serial Data via Read Holding Registers Message

The following screen depicts a Read Holding Registers message used to receive raw serial data in ladder logic.
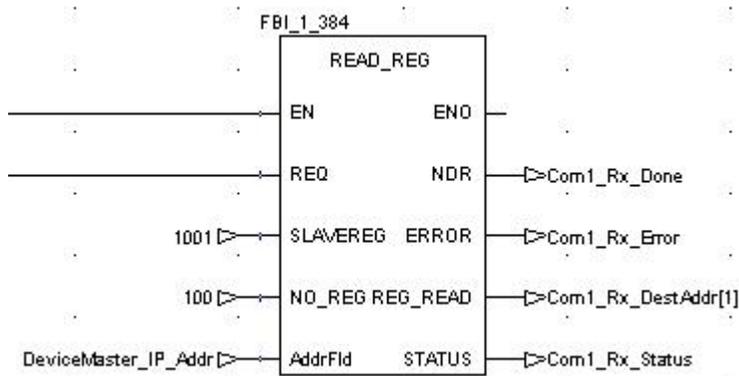
Where:	
SLAVEREG	Refers to the port receive data address (+1 for use with Concept).
NO_REG	The maximum receive message size in 16 bit words. The maximum size of 100 includes two words for the sequence number and length and then up to 196 bytes of serial data.
NDR	The **done** flag.
ERROR	The **error** flag.
REG_READ	The destination address to place the received data.
STATUS	The message status word.
AddrFld	This contains the PLC Modbus/TCP slot/index and IP address of the ICDM-RX/MOD.

11/4/20

PEPPERL+FUCHS

A.2.2.2. Transmit Serial Data via Write Multiple Registers Message

The following screen depicts a *Write Multiple Registers* message used to transmit raw serial data in ladder logic.

```
                                      FBI 1 424
                                    WRITE_REG

                            EN              ENO

                            REQ            DONE ──▷Com1_Tx_Done

                  1301▷──── SLAVEREG     ERROR ──▷Com1_Tx_Error

                   100 ▷──── NO_REG

     Com1_Tx_Reg_Write[1]▷── REG_WRIT

      DeviceMaster_IP_Addr▷── AddrFld     STATUS ──▷Com1_Tx_Status
```
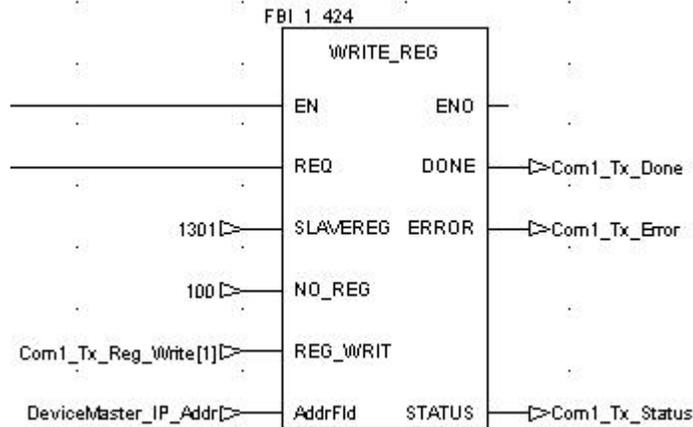
Where:	
SLAVEREG	Refers to the port transmit data address (+1 for use with Concept).
NO_REG	The maximum receive message size in 16 bit words. The maximum size of 100 includes two words for the sequence number and length and then up to 196 bytes of serial data.
REG_WRIT	The memory location where the data message to transmit resides on the PLC. (This includes the sequence number, length in bytes, and serial data to transmit.)
DONE	The **done** flag.
ERROR	The **error** flag.
STATUS	The message status word.
AddrFld	This contains the PLC Modbus/TCP slot/index and IP address of the ICDM-RX/MOD.

A.2.2.3. Set Receive Sequence Number via Write Multiple Registers Message

The following screen depicts a *Write Multiple Registers* message used to initialize the receive data sequence number in ladder logic.

```
                                     FBI_1_252
                                    WRITE_REG

                            EN              ENO

                            REQ            DONE ──▷Com1_SetRxSeqMsgDone

                  1257▷──── SLAVEREG     ERROR ──▷Com1_SetRxSeqMsgError

                    1 ▷──── NO_REG

      Com1_Rx_DestAddr[1]▷── REG_WRIT

      DeviceMaster_IP_Addr▷── AddrFld     STATUS ──▷Com1_SetRxSeqMsgStatus
```
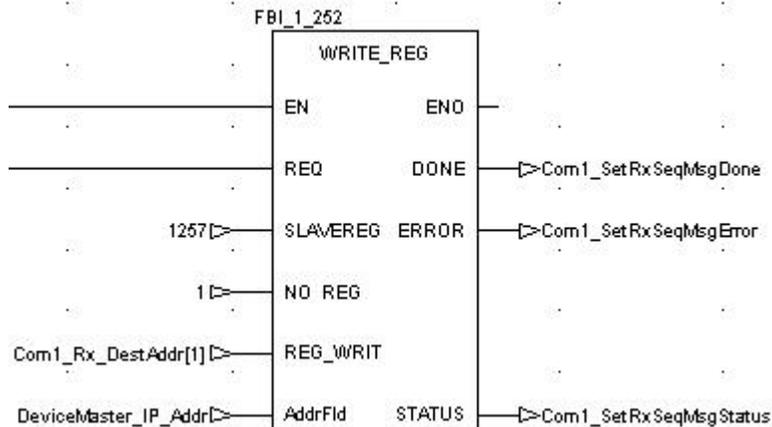
11/4/20

Where:	
SLAVEREG	Refers to the port receive sequence number address (+1 for use with Concept).
NO_REG	Set to 1.
REG_WRIT	The memory location where the receive sequence number resides on the PLC.
DONE	The **done** flag.
ERROR	The **error** flag.
STATUS	The message status word.
AddrFld	This contains the PLC Modbus/TCP slot/index and IP address of the ICDM-RX/MOD.

A.2.2.4. Set Transmit Sequence Number via Write Multiple Registers Message

The following screen depicts a *Write Multiple Registers* message used to initialize the transmit data sequence number in ladder logic.
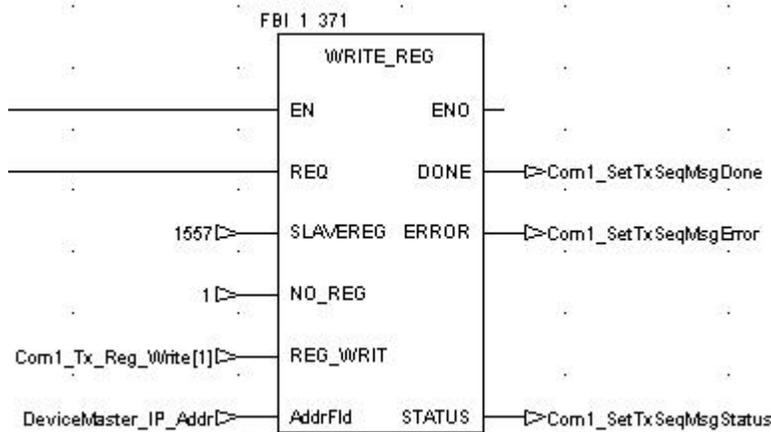
Where:	
SLAVEREG	Refers to the port transmit sequence number address (+1 for use with Concept).
NO_REG	Set to 1.
REG_WRIT	The memory location where the transmit sequence number resides on the PLC.
DONE	The **done** flag.
ERROR	The **error** flag.
STATUS	The message status word.
AddrFld	This contains the PLC Modbus/TCP slot/index and IP address of the ICDM-RX/MOD.

PEPPERL+FUCHS

A.2.2.5. Read Serial Port Statistics via Read Holding Registers Message

The following screen depicts a *Read Holding Registers* message used to retrieve the serial port statistics in ladder logic.
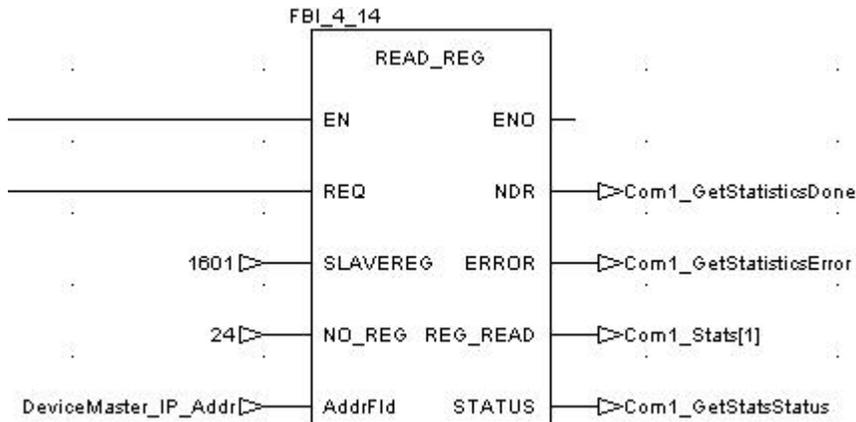
Where:	
SLAVEREG	Refers to the serial port statistics address on the ICDM-RX/MOD (+1 for use with Concept).
NO_REG	Set to 24 (size of statistics data in words).
NDR	The **done** flag.
ERROR	The **error** flag.
REG_READ	The destination address on the PLC to place the statistics data. (Must be at least 24 words in length.)
STATUS	The message status word.
AddrFld	This contains the PLC Modbus/TCP slot/index and IP address of the ICDM-RX/MOD.

PEPPERL+FUCHS

A.2.2.6. Modbus/TCP Slot/Index and ICDM-RX/MOD IP Address Definition

The following screen displays the AddrFld used in all Concept Modbus/TCP messages.
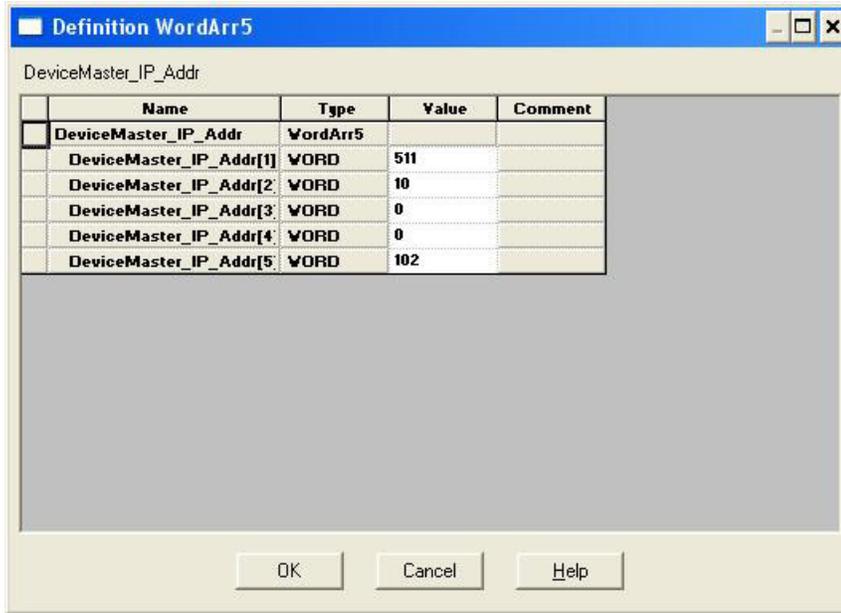
Where:	The first entry (511 or 1FF hex) denotes the following:
Byte 1	• LS byte must be 255 (FF hex) to indicate to the ICDM-RX/MOD that this is raw/ASCII serial data. • MS byte: Momentum PLC = 1 • Quantum PLC = Slot number of Ethernet card • Compact PLC = Slot number of Ethernet card
Byte 2	MS byte of IP Address.
Byte 3	Second byte of IP Address.
Byte 4	Third byte of IP Address.
Byte 5	LS byte of IP Address.

A.2.3. Concept Example Programs

⚠️ **Caution**

Disclaimer: **Comtrol supplies example PLC programs for demonstration purposes only. They are intended for the sole purpose of an example loop-back demonstration in a controlled lab environment. They are not intended for use in a production environment and may not function correctly on all PLCs. Comtrol does not warrant these example programs or any part thereof. The user assumes all liability for any modification to and use of a modified example program.**

The following PLC programs have been included with the released binary. They are designed to interface to a ICDM-RX/MOD 1-port or port one of a ICDM-RX/MOD 2- or 4-port. Additional programming is required to interface to additional ports on a ICDM-RX/MOD 2- or 4-port.

Note: *The following example programs were developed with version 2.6 of Concept and a Schneider Electric Momentum PLC.*

11/4/20

PEPPERL+FUCHS

A.2.3.1. LPBKCNCP

This example program demonstrates a loop-back PLC program using *Read Holding Registers* and *Write Multiple Registers* messages in a standard *polling* type receive method. This program initializes receive and transmit data sequence numbers at startup and then loops raw data via a loop-back plug on the serial port. The data is transmitted and received and the sequence numbers are incremented.

The following files apply:

- **LPBKCNCP.SEC** – Ladder logic and variable definitions in Concept programming format.
- **LPBKCNCP.CCF** – Configuration file.
- **LPBKEXPL.RDF** – Reference data template file.

See *LPBKCNCP Example Program* on Page 114 for more information.

A.2.3.2. SCANCNCP

This example program demonstrates a loop-back PLC program using the I/O Scanner utility on the Concept PLC programming software. This program uses the I/O Scanner to send and receive serial data at a predefined rate. The data is transmitted and received and the sequence numbers are incremented.

The following files apply:

- **SCANCNCP.SEC** – Ladder logic and variable definitions in Concept programming format
- **SCANCNCP.CCF** – Configuration file
- **LPBKEXPL.RDF** – Reference data template file.

Note: *The ICDM-RX/MOD should be reset before starting a PLC program using the I/O Scanner due to PLC program execution scheduling. If the ICDM-RX/MOD is not reset, the sequence numbers may be out of sync. This may result in receiving outdated raw serial data as well as an unexpected transmission of serial data. A Transmit Unexpected Sequence Number error may also occur.*

See *SCANCNCP Example Program* on Page 120 for more information.

A.2.3.3. Setting up and Running the Concept Example Programs

The following steps are required to set up and run the Concept example programs.

If you have not done so, configure the ICDM-RX/MOD by setting the IP address, mask, and gateway for your network and load the Modbus/TCP binary file.

1. Attach the loop-back plug on the serial port.
   a. Attach the PLC and ICDM-RX/MOD to the same Ethernet subnet.
2. Open the *Serial Configuration* web page by entering the ICDM-RX/MOD IP address in your web browser to configure the ICDM-RX/MOD serial port.
3. Set the following *Serial Port Settings*:
   - **Mode**: RS-232
   - **Baud**: 57600
   - **Parity**: none
   - **Data Bits**: 8
   - **Stop Bits**: 1
   - **Flow Control**: none
   - **DTR**: off
   - **Rx Timeout Between Packets**: 200
4. Set the following *General Protocol Settings*:
   - **Serial Port Protocol**: Raw-Data

11/4/20

- • **Discard Rx Packets With Errors**: Enable

5. Set the following *Raw-Data Settings*:
   - • **STX Rx Detect**: one byte, Byte 1 = 2
   - • **ETX Rx Detect**: one byte, Byte 1 = 3
   - • **STX Tx Append**: one byte, Byte 1 = 2
   - • **ETX Tx Append**: one byte, Byte 1 = 3
   - • **Strip Rx Stx/ETX**: Enable
   - • **Rx MS Byte First**: *Optional*
   - • **Tx MS Byte First**: *Optional*

6. Select **Enable** for the **Reset Port** option.

7. Select the **Save** option.

8. Choose either the loop-back (**LPBKCNCP**) or I/O Scanner (**SCANCNCP**) example programs.

9. Load the **.SEC**, **.CCF,** and **.RDF** files into the desired directory.

10. Open Concept.

11. Open a new project by selecting **File->New Project**.
    a. In the *Create a new project database* pane, enter a file name under *File name*.
    b. Navigate to the directory where you wish to place the new project.
    c. Select **OK**.

12. Import the configuration by selecting **File->Import…**
    a. In the *Select Source File Format* pane, select **Configuration.**
    b. Select **OK**.
    c. In the *Configuration Import* pane, select the **.CCF** file.
    d. Select **OK**.
    e. Select **OK** in the *Configuration Import completed* pane.
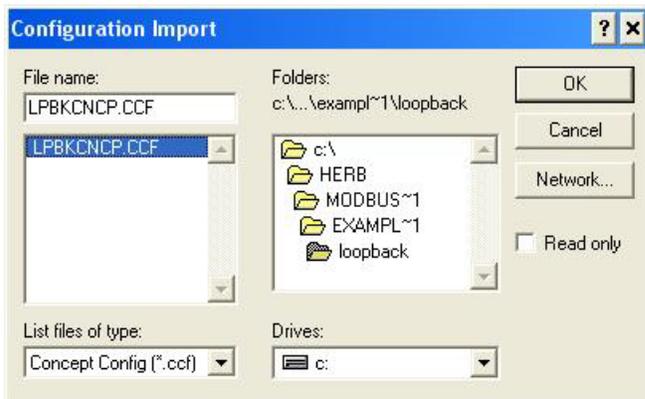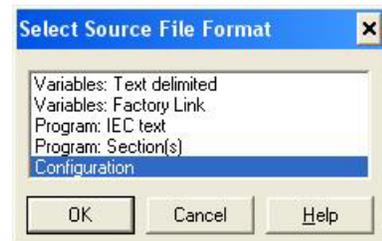
13. Import the program sections by selecting **File->Import…**
    a. In the *Select Source File Format* pane, select **Program Section(s).**
    b. Select **OK**.
    c. In the *select section_file* pane, select the **.SEC** file.

11/4/20

PEPPERL+FUCHS
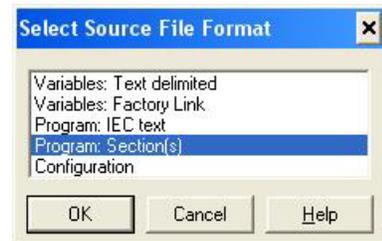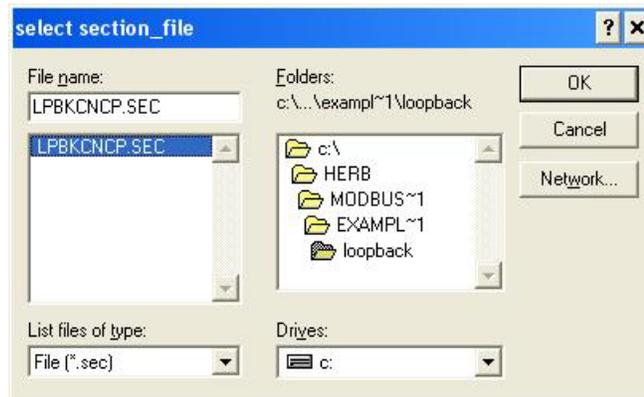
d.   Select **OK**.

e.   Select **YES** when asked whether to *save project file first*.

f.   In the *Replacement* pane, select **OK**.
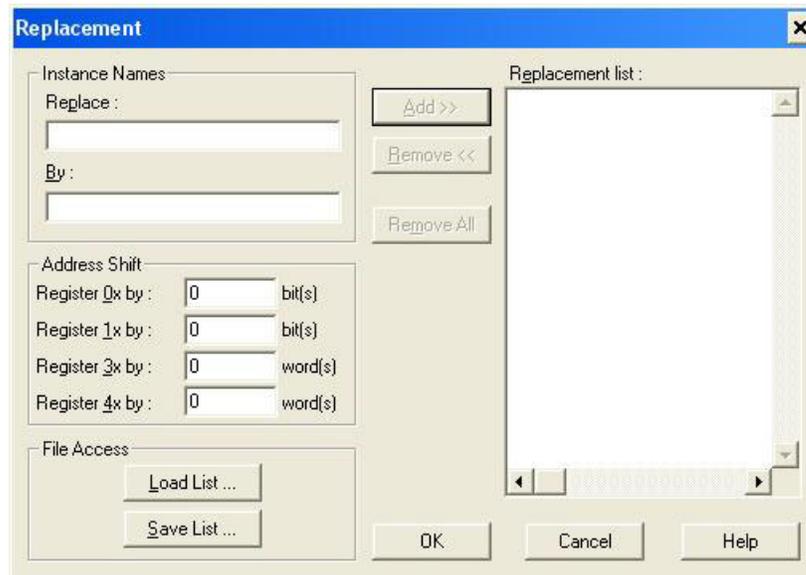
g.   Select **OK** in the status window.

PEPPERL+FUCHS

14. Modify the configuration for your PLC.

    a. Select **Project->Configurator**.
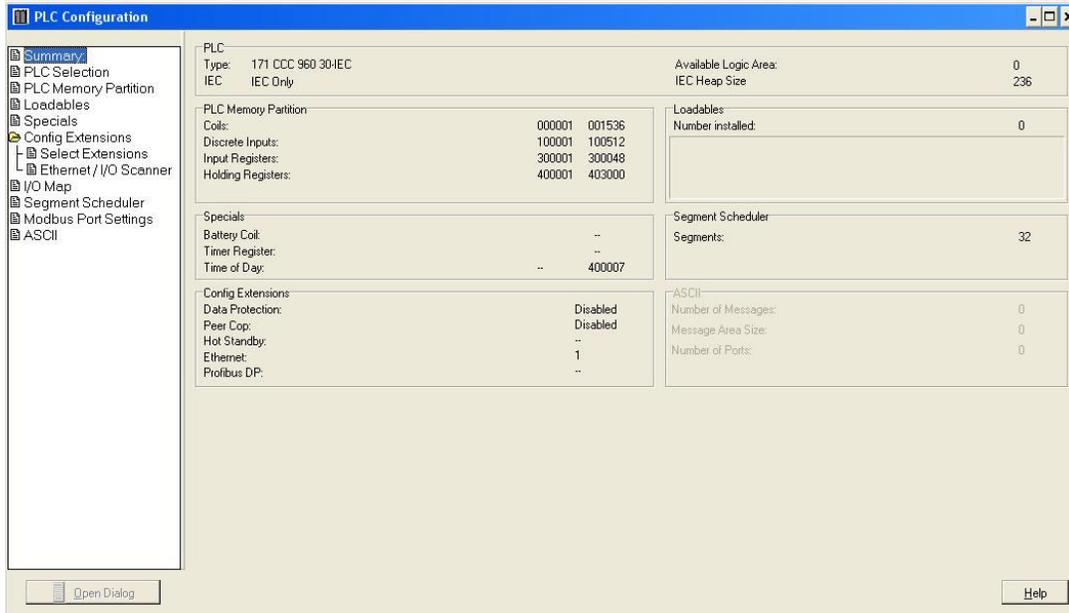
In the *PLC Selection* pane:

    b. Select your **PLC Family**.

    c. Select your **CPU/Executive**.

    d. Select your **Memory Size**.

    e. Select **OK**.

11/4/20

PEPPERL+FUCHS
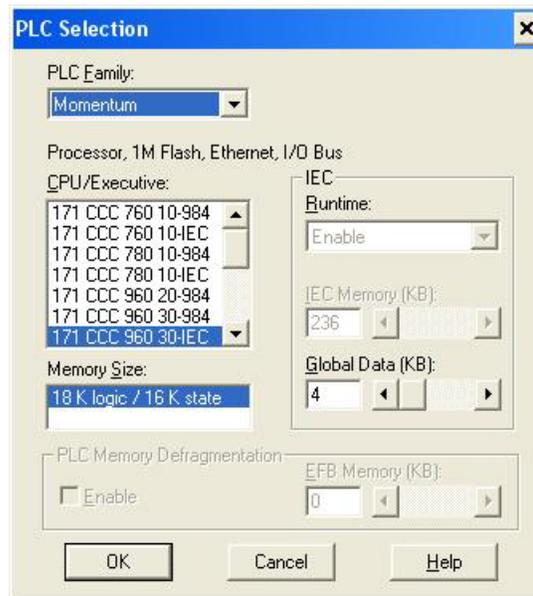
15. In the *PLC Memory Partition* pane:
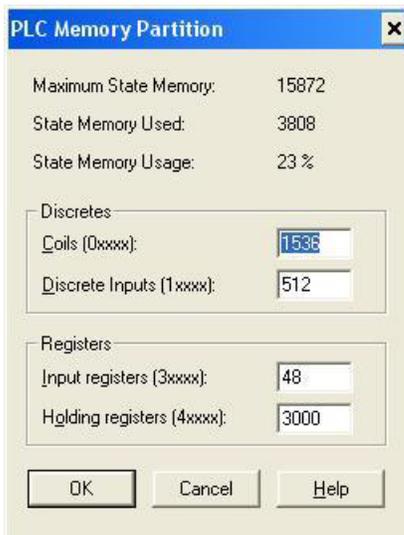
a.   Verify that your PLC has sufficient memory. The example program requires 3000 holding registers.

b.   Select **OK**.

16. In the *Config Extensions->Ethernet/ IO Scanner* pane:

a.   Select your method of specifying an IP address.

If you are specifying an IP Address:

- Enter the PLC IP Address under **Internet Address**.
- Enter the Gateway address.
- Enter the Subnet Mask.

b.   Verify the **Master Module** is configured properly.

c.   Select **OK**.

PEPPERL+FUCHS
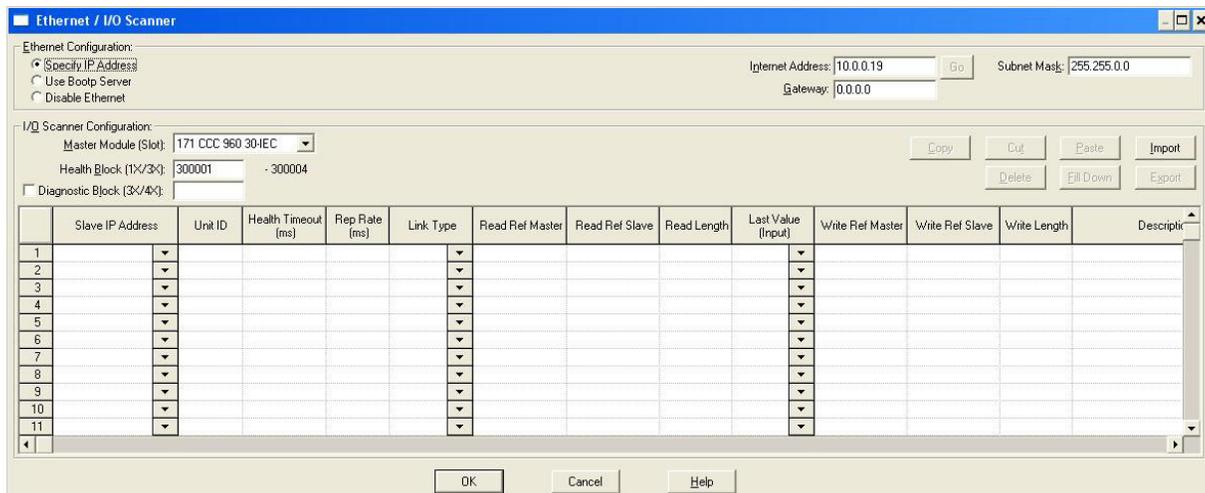
17. Modify the ICDM-RX/MOD IP Address in the *DeviceMaster_IP_Addr* array:
    a. Select **Project->Variable Declarations…**
    b. Click the **DeviceMaster_IP_Addr->Set…** variable, which is the slot/device index number.

       For the Schneider Electric Momentum PLC, the first word does not change.

       For the Schneider Electric Quantum or Compact PLC, set the upper eight bits to the slot number of the Ethernet module.
    c. WORDS two through five contain the IP address in the standard 255.255.255.255 mask format.
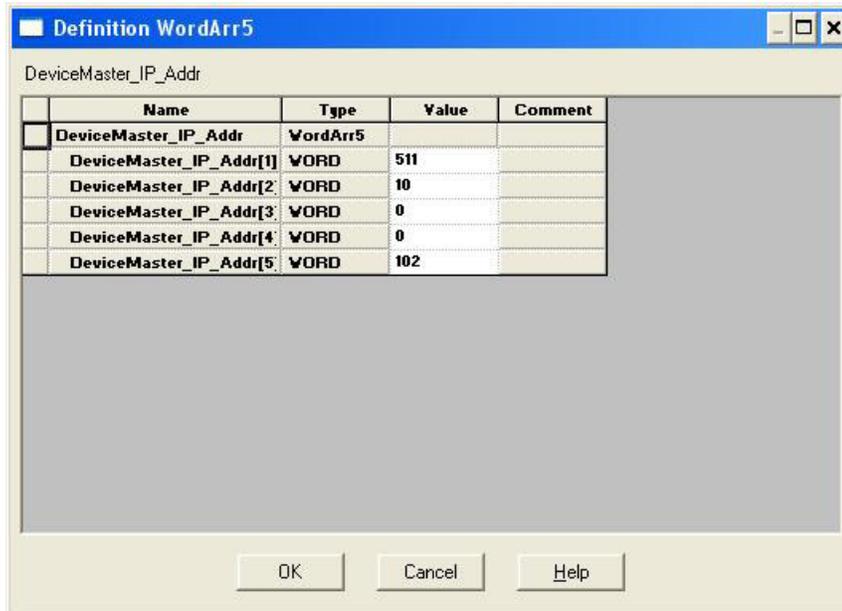
18. *For the SCANCNCP example program only*, modify the ICDM-RX/MOD IP address in the I/O scanner window. Change the default 10.0.0.102 IP address to that of your ICDM-RX/MOD.
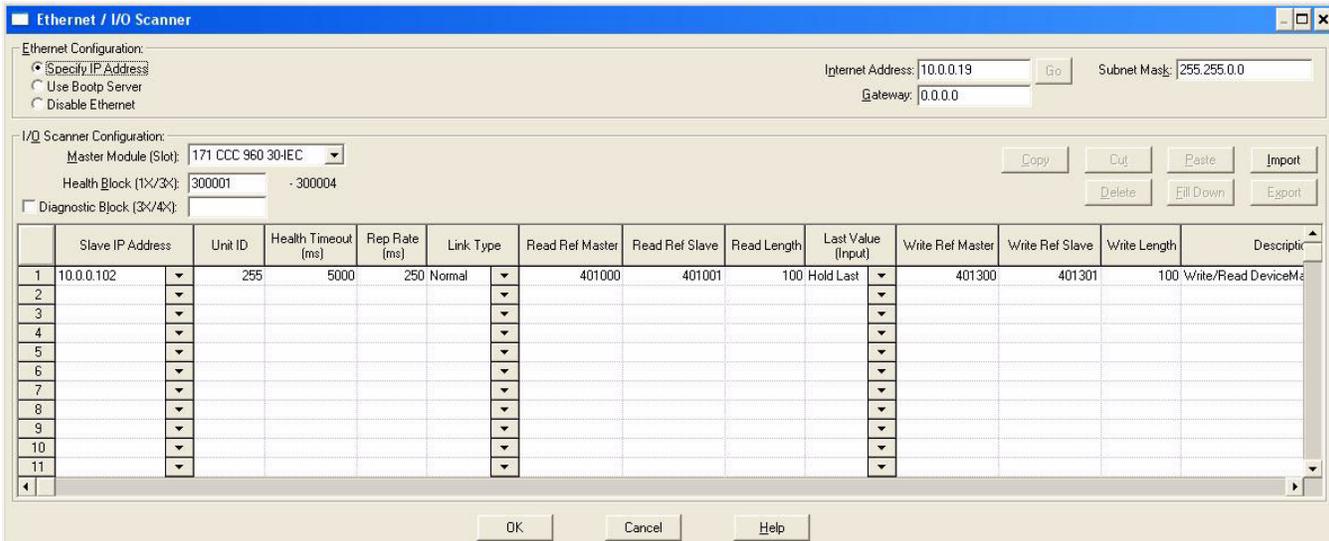
19. Analyze the program by selecting **Project->Analyze Program**.
    a. There should be no errors.
    b. There may be a few *Multi-assignment* warnings, but these can be ignored.

11/4/20

PEPPERL+FUCHS

20. Connect to the PLC by selecting **Online->Connect….**

   Configure the *Connect to PLC* pane.

   a. Under *Protocol Type*, select **TCP/IP**.

   b. Under *IP Address or DNS host name*, enter the PLC IP address.

   c. Select **Change Configuration**.

   d. Select **OK**.
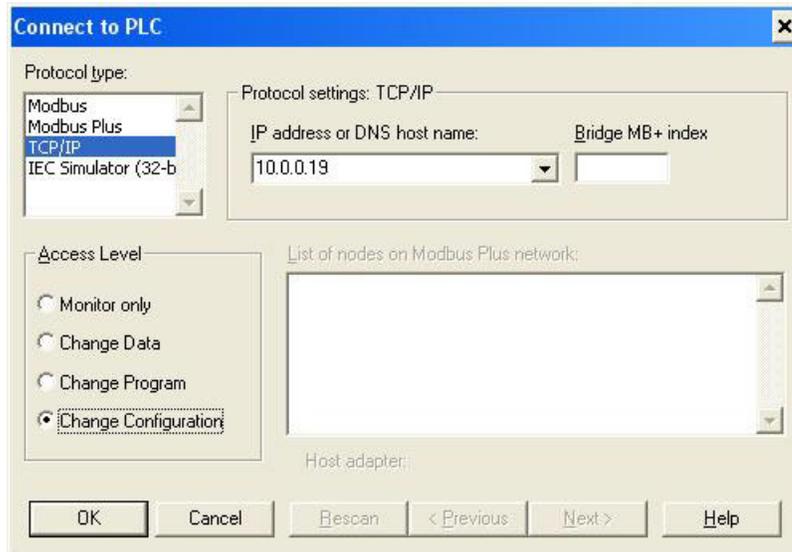
21. Download the PLC program by selecting **Online->Download…**

   In the *Download Controller* pane:

   a. Select **ALL**.

   b. Select **Download…**

22. Set up the program monitoring:

   a. Select the **Open Reference Data Template** button on the menu.

      In the *Open Reference Data Template* pane:

      • Locate and select the **LPBKEXPL.RDF** template file supplied with the example files.

      • Select **OK**.

      Note: *There may be warnings when loading the reference template file and running the* **SCANCNCP** *example program. These warnings can be ignored.*

PEPPERL+FUCHS

b. Animate the reference template by clicking once on the template and then selecting **Online->Animate**.

c. Click once on the **RxTxDataSection** section. Select **Online->Animate Booleans**.

23. Start the processor by selecting **Online->Online Control Panel…**
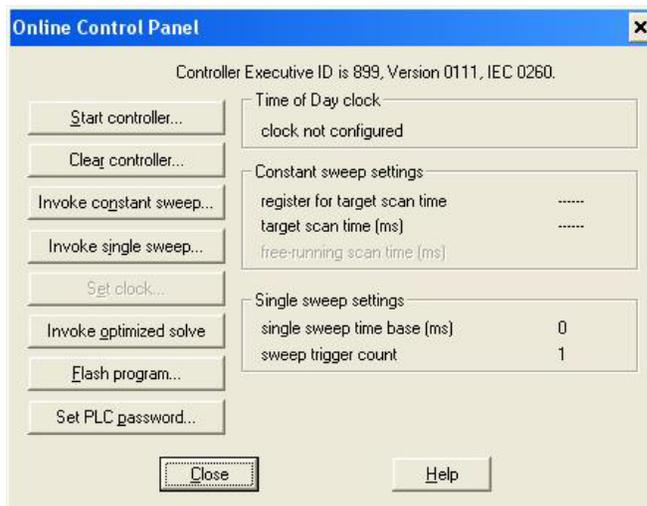
In the *Online Control Panel*, select **Invoke Constant Sweep…**

a. Verify the 4x register is 2999. (This is an unused register.)

b. Verify the sweep rate is 10 msec.

c. Select **OK**.

d. Select **Start Controller**.

e. Select **Close**.

24. Observe the data being transmitted and received.

11/4/20

PEPPERL+FUCHS

PEPPERL+FUCHS
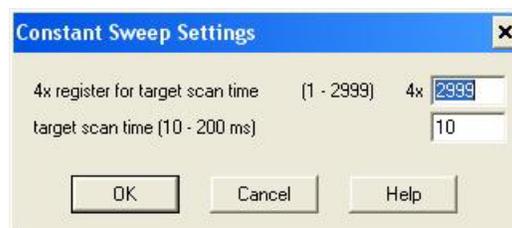
Appendix B. LPBKCNCP Example Program

The following is the ladder logic for the LPBKCNCP example program:

DISCLAIMER
Comtrol suppiles example PLC programs for demonstration purposes only.
This PLC program is intended for the sole purpose of an example loop-back demonstration
in a controlled lab environment. This example PLC program is not intended to be used in a
production enviroment and may not function correctly on all PLCs. Comtrol does not warrant
this example program or any part thereof. The user assumes all liability for any modification
to and use of a modified example program.

This is the loop-back timer. It controls how fast data is transmitted and received.

```
                    FBI_1_22
                    ┌──────────┐
                    │   TON    │
   Run   ClockEnable│          │
   ──┤├──────┤├─────┤IN      Q ├──▷ClockTickTrue
                    │          │
   SystemTimerIn ▷──┤PT     ET ├──▷SystemTimerOut
                    └──────────┘
```

Timer handling: Re-enable clock after resetting to restart.

```
   Run   ClockEnable   ClockEnable
   ──┤├──────┤/├─────────(S)
```

Initialization: Clear the Rx produced sequence number on the DeviceMaster UP.

```
                                                                 FBI_1_252
                                                            ┌─────────────────┐
   Run                                                      │   WRITE_REG     │
   ──┤├──────                                               │EN           ENO │
   Run   ClockTickTrue  Com1_SetRxSeqNum  MsgActive MsgActive│                 │
   ──┤├──────┤├─────────────┤├──────────────┤/├─────(S)─────┤REQ         DONE ├──▷Com1_SetRxSeqMsgDone
                                                            │                 │
                                           1257 ▷───────────┤SLAVEREG   ERROR ├──▷Com1_SetRxSeqMsgError
                                                            │                 │
                                              1 ▷───────────┤NO_REG           │
                                                            │                 │
                              Com1_Rx_DestAddr[1] ▷─────────┤REG_WRIT         │
                                                            │                 │
                            DeviceMaster_IP_Addr ▷──────────┤AddrFld   STATUS ├──▷Com1_SetRxSeqMsgStatus
                                                            └─────────────────┘
```

Initialization: Clear the Tx produced sequence number on the DeviceMaster UP.

```
                                                                 FBI_1_371
                                                            ┌─────────────────┐
   Run                                                      │   WRITE_REG     │
   ──┤├──────                                               │EN           ENO │
   Run   ClockTickTrue  Com1_SetTxSeqNum  MsgActive MsgActive│                 │
   ──┤├──────┤├─────────────┤├──────────────┤/├─────(S)─────┤REQ         DONE ├──▷Com1_SetTxSeqMsgDone
                                                            │                 │
                                           1557 ▷───────────┤SLAVEREG   ERROR ├──▷Com1_SetTxSeqMsgError
                                                            │                 │
                                              1 ▷───────────┤NO_REG           │
                                                            │                 │
                              Com1_Tx_Reg_Write[1] ▷────────┤REG_WRIT         │
                                                            │                 │
                            DeviceMaster_IP_Addr ▷──────────┤AddrFld   STATUS ├──▷Com1_SetTxSeqMsgStatus
                                                            └─────────────────┘
```

Operational loop: Read the latest data from port 1 on the DeviceMaster UP.

```
                                                              FBI_1_384
                                                           READ_REG
 Run                                                    EN          ENO
 ─┤ ├─
 Run    ClockTickTrue  Com1 ReadData  MsgActive  MsgActive
 ─┤ ├──────┤ ├───────────┤ ├──────────┤/├────────(S)─────REQ        NDR ──▷Com1_Rx_Done
                                                  1001▷─SLAVEREG  ERROR ──▷Com1_Rx_Error
                                                   100▷─NO_REG  REG_READ ──▷Com1_Rx_DestAddr[1]
                                   DeviceMaster_IP_Addr▷─AddrFld   STATUS ──▷Com1_Rx_Status
```

Operational loop: Check for new data received here. If new data is received, the sequence number (first 16 bt word) will have been incremented.

```
                    .1.431                              .1.432
                 WORD_TO_UINT                          NE_UINT
 Run            EN        ENO                          EN  ENO
 ─┤ ├───────────                                       
 Com1_Rx_DestAddr[1]▷─                 ─▷Com1_RxProdSeqNum   Com1_RxProdSeqNum▷─        ─▷Com1_RxNewData
                                                            Com1_RxConSeqNum▷─
```

Operational loop: If the new data flag is set, process new data here (Just bumping a total message counter as example)

```
                                             .1.433                            .1.403
                                            MOVE                             ADD_UDINT
 Run    Com1 RxNewData   Com1_RxNewData    EN  ENO                          EN     ENO
 ─┤ ├─────────┤ ├───────────(R)──────────                                  
                          Com1_RxProdSeqNum▷─      ─▷Com1_RxConSeqNum   Com1_RxTotalMsgs▷─   ─▷Com1_RxTotalMsgs
                                                                                       1▷─
```

Operational loop: If there is new data to transmit, load the data into the message (starting at the third word), and then increment the Transmit Sequence number.

```
                                             .1.451
                                            ADD_UDINT
 Run    ClockTickTrue  Com1 TransmitData  MsgActive  EN    ENO                .1.452
 ─┤ ├──────┤ ├───────────┤ ├──────────────┤/├───────                        UDINT_TO_WORD
                          Com1_TxProdSeqNum▷─      ─▷Com1_TxProdSeqNum      EN        ENO
                                             1▷─                Com1_TxProdSeqNum▷─    ─▷Com1_Tx_Reg_Write[1]
```

PEPPERL+FUCHS

Operational loop:Handle rollover of 16 bit integer transmit sequence number.

```
                                                                    .1.454
  Run      ClockTickTrue  Com1_TransmitData        MsgActive     WORD_TO_UDINT
  ─┤├────────┤├──────────────┤├──────────────────────┤/├────────EN        ENO
                                                                
                          Com1_Tx_Reg_Write[1]▷                    ▷Com1_TxProdSeqNum
```

Operational loop:Transmit data if transmit dat flag is set.

```
                                                                    FBI_1_424
  Run                                                             WRITE_REG
  ─┤├──────────────────────────────────────────────────────────EN        ENO
  Run      ClockTickTrue  Com1_TransmitData   MsgActive   MsgActive
  ─┤├────────┤├──────────────┤├─────────────────┤/├────────(S)───REQ       DONE──▷Com1_Tx_Done
                                                         1301▷    SLAVEREG ERROR──▷Com1_Tx_Error
                                                          100▷    NO REG
                                       Com1_Tx_Reg_Write[1]▷      REG_WRIT
                                       DeviceMaster_IP_Addr▷      AddrFld  STATUS─▷Com1_Tx_Status
```

Operational loop: Wait for active messages to complete here.

```
  Run      MsgActive    Com1_SetRxSeqNum        Com1_SetRxSeqMsgDone    Com1_SetRxSeqNum    Com1_SetTxSeqNum    MsgActive
  ─┤├────────┤├────────────┤├───────┬────────────────┤├──────────────────(R)──────────────────(S)────────────────(R)
                                    │           Com1_SetRxSeqMsgError   Com1_SetRxSeqNum    Com1_SetTxSeqNum    MsgActive
                                    └────────────────┤├──────────────────(R)──────────────────(S)────────────────(R)

  Run      MsgActive    Com1_SetTxSeqNum        Com1_SetTxSeqMsgDone    Com1_SetTxSeqNum    Com1_TransmitData   MsgActive
  ─┤├────────┤├────────────┤├───────┬────────────────┤├──────────────────(R)──────────────────(S)────────────────(R)
                                    │           Com1_SetTxSeqMsgError   Com1_SetTxSeqNum    Com1_TransmitData   MsgActive
                                    └────────────────┤├──────────────────(R)──────────────────(S)────────────────(R)
```

```
  Run      MsgActive    Com1_ReadData       Com1_Rx_Done  Com1_ReadData  Com1_TransmitData    MsgActive    .1.306
  ─┤├────────┤├────────────┤├───────┬──────────┤├──────────(R)────────────(S)───────────────────(R)     ADD_DINT
                                    │     Com1_Rx_Error  Com1_ReadData  Com1_TransmitData    MsgActive  EN       ENO
                                    └──────────┤├──────────(R)────────────(S)───────────────────(R)
                                                                              Com1_Rx_Error_Cnt▷            ▷Com1_Rx_Error_Cnt
                                                                                              1▷
```

```
  Run      MsgActive    Com1_TransmitData   Com1_Tx_Done  Com1_TransmitData  Com1_ReadData  MsgActive    .1.283
  ─┤├────────┤├────────────┤├───────┬──────────┤├──────────(R)────────────(S)───────────────(R)       ADD_DINT
                                    │     Com1_Tx_Error  Com1_TransmitData  Com1_ReadData  MsgActive  EN       ENO
                                    └──────────┤├──────────(R)────────────(S)───────────────(R)
                                                                              Com1_Tx_Error_Cnt▷            ▷Com1_Tx_Error_Cnt
                                                                                              1▷
```

Reset timer to keep operational loop running.

```
  Run      ClockTickTrue  ClockEnable
  ─┤├────────┤├─────────────(R)
```

11/4/20

PEPPERL+FUCHS

The following screens depict the variables for the **LPBKCNCP** example program.

Definition WordArr128

Com1_Rx_DestAddr

Name	Type	Value	Cc
Com1_Rx_DestAddr	**WordArr128**		
Com1_Rx_DestAddr[1]	WORD	0	
Com1_Rx_DestAddr[2]	WORD	0	
Com1_Rx_DestAddr[3]	WORD	0	
Com1_Rx_DestAddr[4]	WORD	0	
Com1_Rx_DestAddr[5]	WORD	0	
Com1_Rx_DestAddr[6]	WORD	0	
Com1_Rx_DestAddr[7]	WORD	0	
Com1_Rx_DestAddr[8]	WORD	0	
Com1_Rx_DestAddr[9]	WORD	0	
Com1_Rx_DestAddr[10]	WORD	0	
Com1_Rx_DestAddr[11]	WORD	0	
Com1_Rx_DestAddr[12]	WORD	0	
Com1_Rx_DestAddr[13]	WORD	0	
Com1_Rx_DestAddr[14]	WORD	0	
Com1_Rx_DestAddr[15]	WORD	0	
Com1_Rx_DestAddr[16]	WORD	0	
Com1_Rx_DestAddr[17]	WORD	0	
Com1_Rx_DestAddr[18]	WORD	0	
Com1_Rx_DestAddr[19]	WORD	0	
Com1_Rx_DestAddr[20]	WORD	0	
Com1_Rx_DestAddr[21]	WORD	0	
Com1_Rx_DestAddr[22]	WORD	0	
Com1_Rx_DestAddr[23]	WORD	0	

[OK]   [Cancel]   [Help]

Note: *There is no valid data in this variable array until received data is requested from the ICDM-RX/MOD.*

Definition WordArr128

Com1_Tx_Reg_Write

Name	Type	Value	Cc
Com1_Tx_Reg_Write	**WordArr128**		
Com1_Tx_Reg_Write[1]	WORD	0	
Com1_Tx_Reg_Write[2]	WORD	196	
Com1_Tx_Reg_Write[3]	WORD	4370	
Com1_Tx_Reg_Write[4]	WORD	4884	
Com1_Tx_Reg_Write[5]	WORD	5398	
Com1_Tx_Reg_Write[6]	WORD	5912	
Com1_Tx_Reg_Write[7]	WORD	6426	
Com1_Tx_Reg_Write[8]	WORD	6940	
Com1_Tx_Reg_Write[9]	WORD	7454	
Com1_Tx_Reg_Write[10]	WORD	7968	
Com1_Tx_Reg_Write[11]	WORD	8482	
Com1_Tx_Reg_Write[12]	WORD	8996	
Com1_Tx_Reg_Write[13]	WORD	9510	
Com1_Tx_Reg_Write[14]	WORD	10024	
Com1_Tx_Reg_Write[15]	WORD	10538	
Com1_Tx_Reg_Write[16]	WORD	43	
Com1_Tx_Reg_Write[17]	WORD	0	
Com1_Tx_Reg_Write[18]	WORD	0	
Com1_Tx_Reg_Write[19]	WORD	0	
Com1_Tx_Reg_Write[20]	WORD	0	
Com1_Tx_Reg_Write[21]	WORD	0	
Com1_Tx_Reg_Write[22]	WORD	0	
Com1_Tx_Reg_Write[23]	WORD	0	

[OK]   [Cancel]   [Help]

Where:

- The first word contains the sequence number starting at zero.
- The second word contains the length that is set to the maximum number of bytes transmitted by the Concept software package. (100 words = 200 bytes. 196 bytes maximum transmit data size.)
- Words three to 100 contain transmit data.

11/4/20

PEPPERL+FUCHS
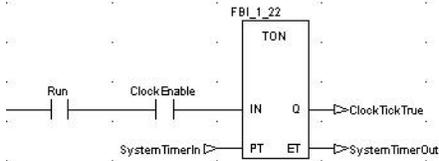
PEPPERL+FUCHS

Appendix C. SCANCNCP Example Program

The following is the ladder logic for the SCANCNCP example program.
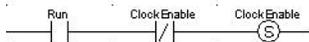
DISCLAIMER
Comtrol suppiles example PLC programs for demonstration purposes only.
This PLC program is intended for the sole purpose of an example loop-back demonstration
in a controlled lab environment. This example PLC program is not intended to be used in a
production enviroment and may not function correctly on all PLCs. Comtrol does not warrant
this example program or any part thereof. The user assumes all liability for any modification
to and use of a modified example program.

NOTE: The DeviceMaster UP must be reset before starting this example program or any other
PLC program that utilizes the PLC I/O scanner utility to communicate to the DeviceMaster UP.
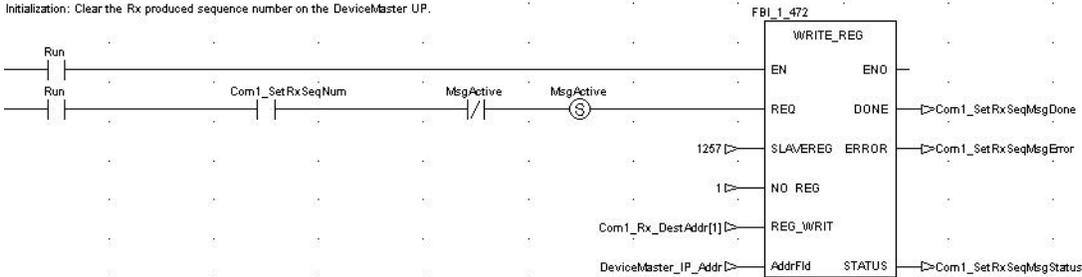
This is the loop-back timer. It controls how fast the transmit data message will be updated
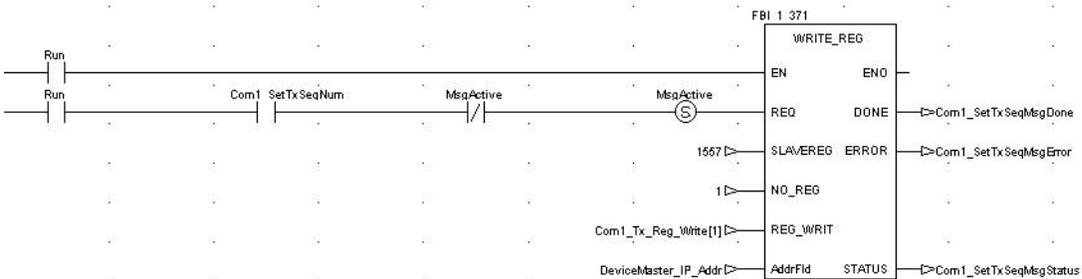so that new serial data is transmitted and received.

FBI_1_22
TON

Run   Clock Enable
                        IN   Q   ▷Clock Tick True

SystemTimerIn ▷   PT   ET   ▷System Timer Out

Re-enable clock after resetting to restart.

Run   Clock Enable   Clock Enable
                            (S)

Initialization: Clear the Rx produced sequence number on the DeviceMaster UP.

FBI_1_472
WRITE_REG

Run
                                                        EN   ENO
Run   Com1_SetRxSeqNum   MsgActive   MsgActive
                                    (S)   REQ   DONE   ▷Com1_SetRxSeqMsgDone

                              1257 ▷   SLAVEREG   ERROR   ▷Com1_SetRxSeqMsgError

                                 1 ▷   NO_REG

            Com1_Rx_DestAddr[1] ▷   REG_WRIT

            DeviceMaster_IP_Addr ▷   AddrFld   STATUS ▷Com1_SetRxSeqMsgStatus

Initialization: Clear the Tx produced sequence number on the DeviceMaster UP.

FBI_1_371
WRITE_REG

Run
                                                        EN   ENO
Run   Com1_SetTxSeqNum   MsgActive   MsgActive
                                    (S)   REQ   DONE   ▷Com1_SetTxSeqMsgDone

                              1557 ▷   SLAVEREG   ERROR   ▷Com1_SetTxSeqMsgError

                                 1 ▷   NO_REG

            Com1_Tx_Reg_Write[1] ▷   REG_WRIT

   DeviceMaster_IP_Addr ▷   AddrFld   STATUS ▷Com1_SetTxSeqMsgStatus

Operational loop: Check for new data received here. If new data is received, the sequence number (first 16 bit word) will have been incremented.
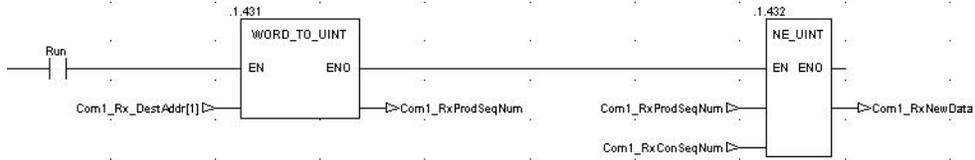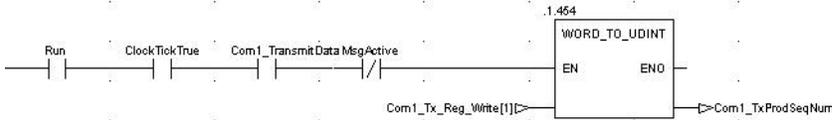

Operational loop: If the new received data flag is set, process new received data here (Just bumping a total message counter as example)
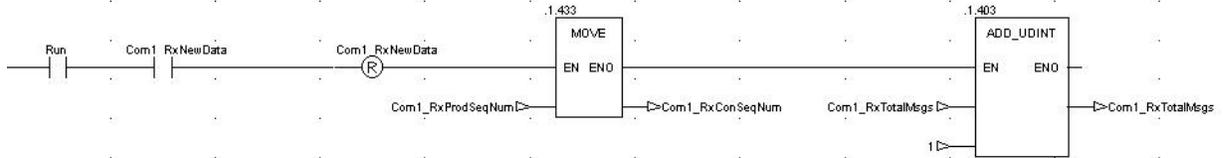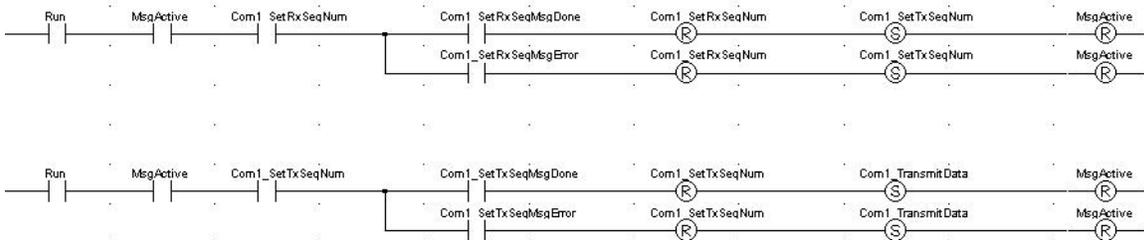

Operational loop: If there is new data to transmit, load the data into the message (starting at the third word), and then increment the Transmit Sequence number.
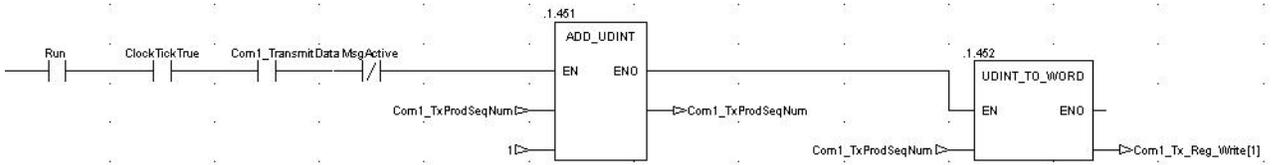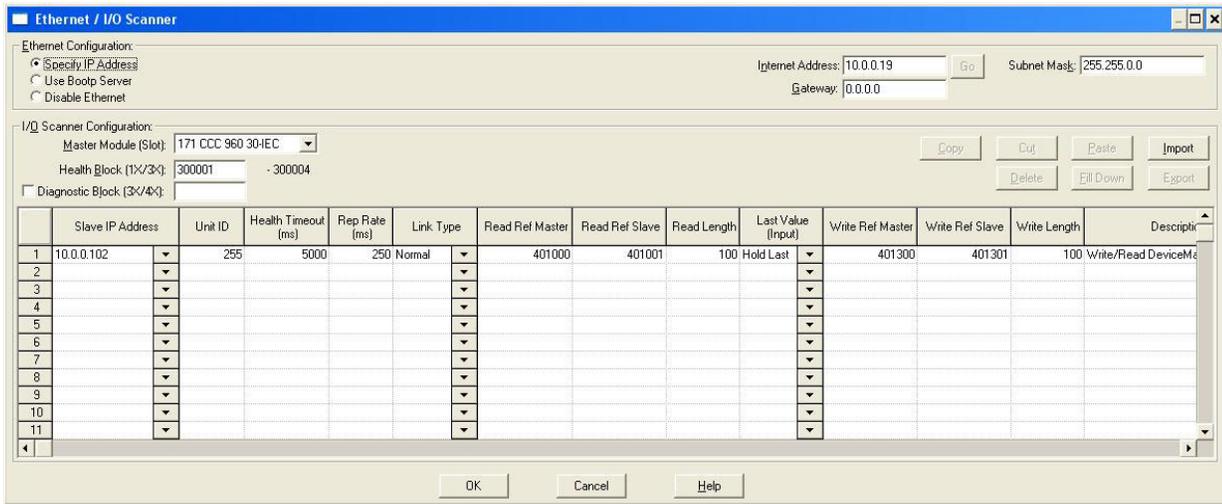

Operational loop: Handle rollover of 16 bit integer sequence number.

Operational loop: Wait for active messages to complete here.
(Initialization messages only for this example program.)

Reset timer.

PEPPERL+FUCHS

The I/O Scanner screen displays.

The variable definitions are the same as for the **LPBKSCAN** program in *Appendix B. LPBKCNCP Example Program* on Page 114.

11/4/20

PEPPERL+FUCHS

FACTORY AUTOMATION – SENSING YOUR NEEDS

www.pepperl-fuchs.com

PEPPERL+FUCHS
SENSING YOUR NEEDS

DOCT-6961

11/4/20