



CDN466-X Series DeviceNet Gateway User Manual



Control & Information Technology Group
134 W Rio Robles Drive
San Jose, CA 95134

Main: 408.750.0300
Fax: 408.750.2990

Manual Rev. 1.1
05/11

Copyright

This manual and the software described in it are copyrighted with all rights reserved. Under the copyright laws, this manual and software may not be copied, in whole or part, without the prior written consent of MKS Instruments. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others whether or not sold, but all of the materials purchased may be sold, given, or loaned to another person. Under the law, copying includes translating into another language or format.

© MKS Instruments - CIT Products Group, 2011

134 W Rio Robles Drive
San Jose, CA 95134

Preface

About this manual

This manual is designed to serve as a guideline for the installation, setup, operation and basic maintenance of the CDN466-X gateway. The information contained within this manual, including product specifications, is subject to change without notice. Please observe all safety precautions and use appropriate procedures when handling the CDN466-X gateway product and its related software.

Table of Contents

Table of Figures	4
Revision History	5
General Information	6
1.1 CONVENTIONS USED IN THIS USER MANUAL	6
2 Overview	7
2.1 HARDWARE	7
3 Theory of Operation	9
3.1 GATEWAY OPERATION	9
3.1.1 Serial Channel Interface	12
3.1.2 Status Overhead Byte	13
3.1.3 Receiving Message	13
3.1.4 Transmitting Message	16
3.1.5 Synchronization	17
4 Gateway Configuration	19
4.1 CONFIGURE DEVICENET INTERFACE	19
4.1.1 DeviceNet Baud Rate Switch	19
4.1.2 MacID Switch Settings	20
4.1.3 Serial Baud Rate Switch	20
4.1.4 Power Up Gateway	21
4.1.5 Configure Serial Channel	22
4.1.6 Configure DeviceNet Master Scanlist	27
5 DeviceNet Specification	28
DEVICENET MESSAGE TYPES	28
DEVICENET CLASS SERVICES	28
DEVICENET OBJECT CLASSES	29
6 Quick Star Guide	50
6.1 WIRING	50
6.1.1 DeviceNet Connector Pin out	50
6.1.2 Serial Channel Pin outs and Connections	51
6.2 REQUIRED HARDWARE	51
6.3 NETWORK SETUP	51
6.4 REGISTER EDS FILE	52
6.5 CDN466-X GATEWAY CONFIGURATION	54
6.5.1 Serial Parameters	57
6.5.2 Message Format	59
6.6 CONFIGURE DEVICENET MASTER SCANLIST	59
7 Configuration Example	65
Troubleshooting	75
Appendix A – Product Specification	77
Appendix B: ASCII Character Codes	78
WARRANTY	80

Table of Figures

Figure 1 CDN466-X High Level Block Diagram	9
Figure 2 Data Exchange Operation	10
Figure 3 Mapping of DeviceNet Poll Command and Poll Response Data	11
Figure 4 Ladder Logic rung to Acknowledge Receive Sequence Number	18
Figure 5 Ladder Logic Rung to Send New Transmit Sequence Number.....	18
Figure 6 Rockwell RSNetworx™ EDS Wizard Screen	52
Figure 7 Select Correct EDS to Register the CDN466-X	53
Figure 8 Click Nex to Finish Registering CDN466-X EDS file	53
Figure 9 Scan Network for Available Node Connected to Network	54
Figure 10 Upload Default Configuration from Node	55
Figure 11 CDN466-X Attributes Data in Parameters Tab	56
Figure 12 Upload Default Configuration from Node	57
Figure 13 Check for Available Node to Add to Scanlist.....	60
Figure 14 Transfer Unit to Scanlist	61
Figure 15 Edit IO Produce and Consume Size	62
Figure 16 Verify IO Mapping on Scanner Memory Map	62
Figure 17 IO Manual Mapping	63
Figure 18 How to Access Class Instance Editor from RSNetworx™	63
Figure 19 Use Class Instance Editor to change Fault or Idle String Attributes.	64
Figure 20 Network Set up with CDN466-X	65

Revision History

Revision	Description of changes	Date
1.0	First Release	08/26/2009
1.1	Updated address and minor format	04/09/2011

General Information

1.1 Conventions used in this User Manual



Warning

The **WARNING** sign denotes a hazard to personnel. It calls attention to a procedure, practice, condition, or the like, which, if not correctly performed or adhered to, could result in injury to personnel.



Caution

The **CAUTION** sign highlights information that is important to the safe operation of the Gateway, or to the integrity of your files.



Note

The **NOTE** sign denotes important information. It calls attention to a procedure, practice, condition, or the like, which is essential to highlight.

On screen buttons or menu items appear in bold and italics.

Example: Click ***OK*** to save the settings.

Keyboard keys appear in brackets.

Example: [ENTER] and [CTRL]

Pages with additional information about a specific topic are cross-referenced within the text.

Example: (See page xxx)

2 Overview

This document describes how to install, configure, and operate the CDN466-X (CDN466-4 and CDN466-5) series of DeviceNet to serial gateways. The following products are covered in this user manual:

Part Number	FW Rev.	Serial Channel
CDN466-4	4.10X	RS232
CDN466-5	4.10X	RS232

The CDN466-X gateways allow you to easily interface a wide variety of serial devices to any DeviceNet industrial control network. Standard CDN466-X products are tightly packaged and sealed in a rugged industrial case. Board-level and customized gateways are also available upon request.

2.1 Hardware

Receive Status LED (RX)

STATE	DESCRIPTION
OFF	Not receiving data
RED BLINK	Not defined
RED	Receive error
GREEN BLINK	Receiving data
GREEN	Not defined

Transmit Status LED (TX)

STATE	DESCRIPTION
OFF	Not transmitting data
RED BLINK	Not defined
RED	Transmit error
GREEN BLINK	Transmitting data
GREEN	Not defined

Isolated Serial Channel
(male DB9 connector)

PIN	CDN466
1	nc
2	RXD
3	TXD
4	DTR/DSR*
5	SGND
6	DTR/DSR*
7	RTS
8	CTS
9	nc

*Pins 4 and 6 connected internally.

Serial Baud Rate Rotary Switch

DeviceNet Address Rotary Switches

DeviceNet Data Rate Rotary Switch

DeviceNet Status LED (NET)

STATE	DESCRIPTION
OFF	No power
RED BLINK	Configuration error
RED	Unrecoverable error
GREEN BLINK	Not allocated to a master
GREEN	Allocated to a master

Module Status LED (MOD)

STATE	DESCRIPTION
OFF	No power
RED BLINK	Configuration error
RED	Unrecoverable error
GREEN BLINK	Not defined
GREEN	Normal operation

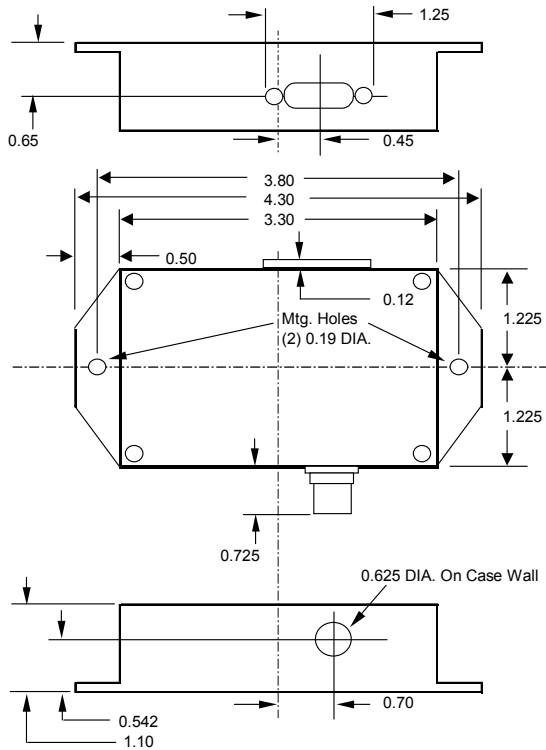
DeviceNet Channel
(male 5-pin micro connector)

PIN	SIGNAL
1	SHIELD
2	V+
3	V-
4	CAN H
5	CAN L

INSTALLATION

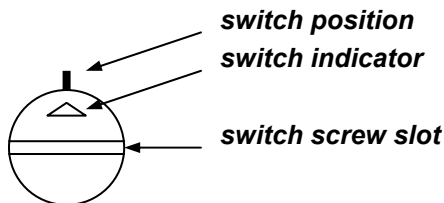
Mount the ToolLink Gateway on a horizontal or vertical surface, in a suitable location or enclosure for your application. Provide sufficient clearance and airflow to maintain 0°C to 70°C ambient operating temperature range. Fasten the ToolLink Gateway to the mounting surface using two screws (not provided) in the 0.19 inch mounting holes.

All dimensions are inches



ROTARY SWITCHES

Set the ToolLink rotary switches to the desired settings. Use a small slotted screwdriver to rotate the switches. Align the indicator arrow to the desired setting, as shown below.



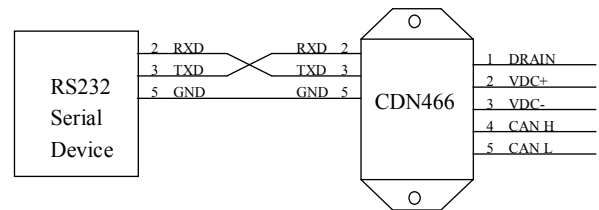
WIRING

The ToolLink Gateway requires two connections – one to the DeviceNet network (male 5-pin micro connector) and one to the serial device (male DB9 connector). DeviceNet and serial cables are available from a variety of industrial sources. Follow all applicable electrical codes in your area when mounting and wiring any electrical device.

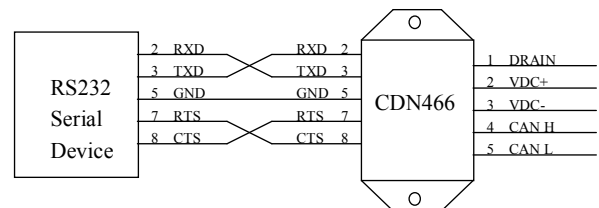
All power is received from the DeviceNet network. The ToolLink Gateway draws up to 200mA from the 24VDC power supply. Select your DeviceNet cables and power supply so that it can provide sufficient current for all networked devices at their peak operating power.

The following are typical ToolLink Gateway wiring examples. Your RS232 or RS485 interface may vary. Refer to your device's documentation for the required data and control signals.

RS232 Interface



RS232 Interface, HW Flow Control



Each rotary switch parameter has a **PGM** option. Setting a switch to PGM allows the parameter to be remotely set over DeviceNet. However, it must first be initialized. To initialize, set the switch to desired value and power up the gateway. The new settings are saved in its memory. Power down and change switch to PGM mode.

3 Theory of Operation

This chapter describes how the CDN466-X gateway operates. You should have a working knowledge of DeviceNet and asynchronous serial communications before continuing. The Open DeviceNet Vendors Association (www.odva.com) is a good source for general DeviceNet information. Refer to your serial device documentation for its protocol information.

3.1 Gateway Operation

The CDN466-X gateway receives asynchronous serial messages over its serial channel and returns the received bytes as input data to the DeviceNet master. The gateway transmits bytes sent as output data from the DeviceNet master out its serial channel. The following diagram shows the major gateway components.

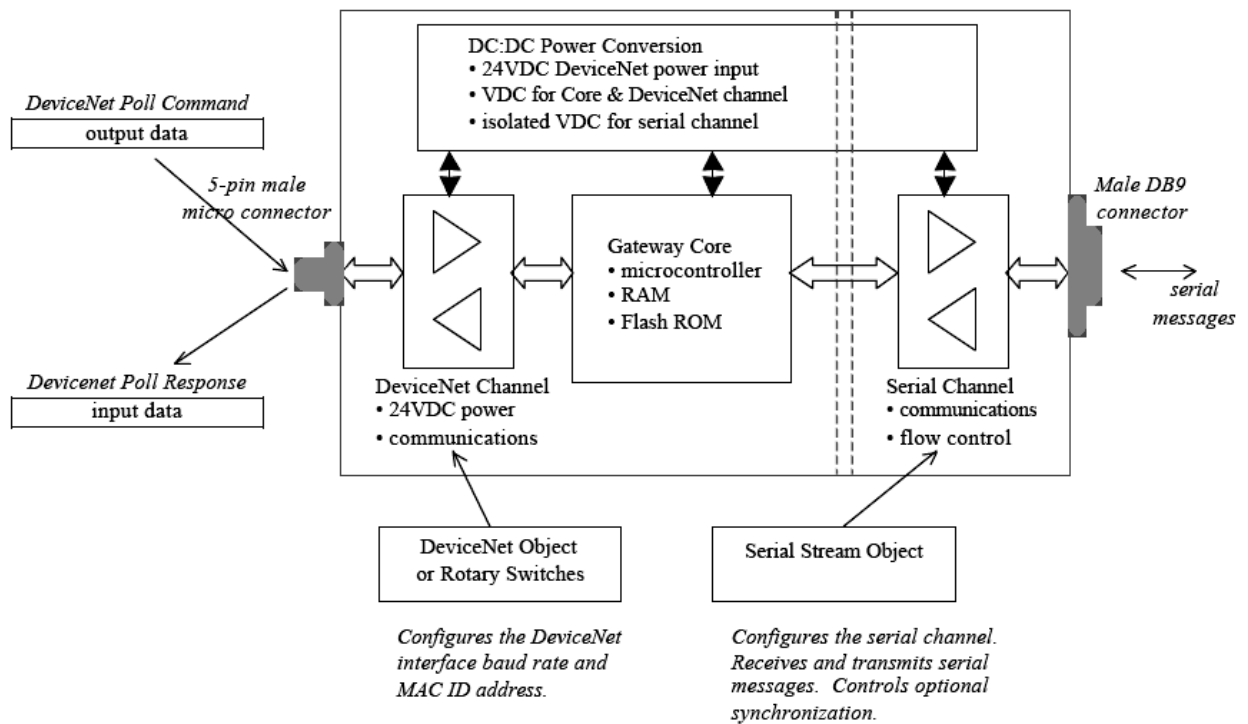


Figure 1 CDN466-X High Level Block Diagram

The DeviceNet Specification defines an Object Model that consists of Objects and Attributes. An Object is a predefined software process, and an Object Attribute is a data value used or generated by that process. An Object Instance is one occurrence of an Object, operating on its unique set of Attribute values. The CDN466-X gateway has six different Object Classes, or types. Five are standard objects defined by the DeviceNet Specification (*Identity, Router, DeviceNet, Assembly, and Connection*). One is a device-specific object defined for the CDN466-X gateway (Serial Stream). The Serial Stream Object configures and controls the serial channel. It receives and packages serial data

into DeviceNet input bytes, and transmits DeviceNet output bytes as serial data. Chapter 5 contains detailed information on each DeviceNet object class, instance, and their associated attributes.

There are four independent processes operating in a CDN466-X gateway application. The first process is the exchange of input and output data between the user application program and the DeviceNet master. The second process is the exchange of input and output data between the gateway and DeviceNet master, using Polled I/O messaging. The third process is receiving serial messages and converting it to input data. The fourth process is converting output data and transmitting it as serial messages.

The DeviceNet Polled I/O Message process consists of the DeviceNet master sending output data to the CDN466-X in the form of a Poll Command message, and the CDN466-X returning input data to the DeviceNet master in a Poll Response message. The output and input data bytes are typically mapped into data files inside the DeviceNet master. These data files are exchanged with the user application program. The application processes the received input data from the gateway and writes new output data to the DeviceNet master, which sends them to the gateway.

The Polled I/O data exchange typically occurs at a faster rate than the serial transmit and receive operation, because the DeviceNet baud rate is much greater than the serial channel baud rate. The CDN466-X has TX and RX buffers to handle the slower serial processes. The gateway also provides synchronization features to ensure delivery of received messages to the application program, and transmission of application messages out the serial channel.

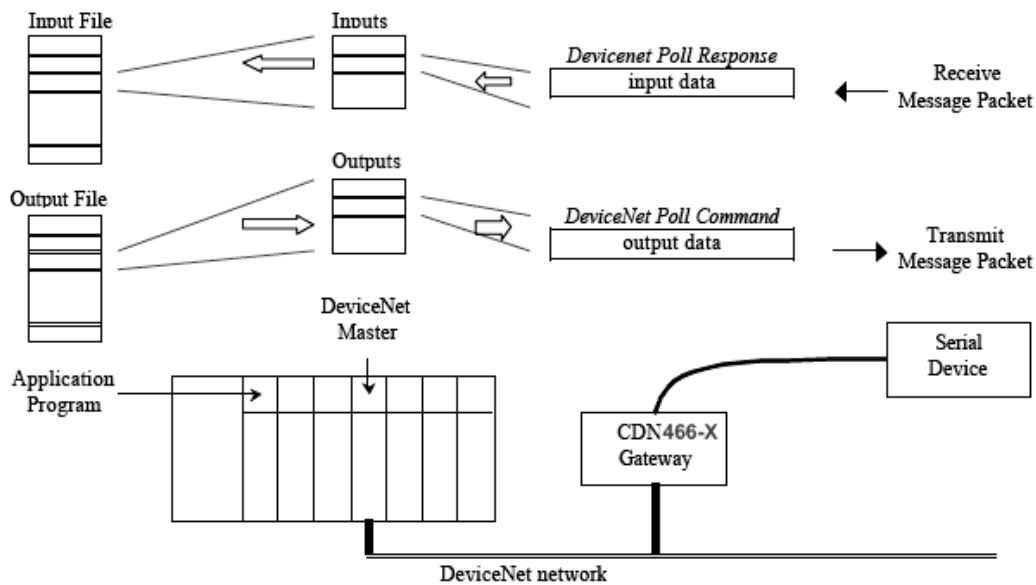


Figure 2 Data Exchange Operation

The CDN466-X configuration defines the number of output bytes in a Poll Command message, and the number of input bytes in a Poll Response message. Each Poll Command and Poll Response message can contain up to 3 overhead bytes for CDN466-X status, data synchronization information, and length byte. The remaining bytes contain output data to be transmitted out the serial channel, or input data received by the serial channel.

The following diagram shows how the input and output bytes map into the Poll Response and Poll Command messages. The gateway supports a maximum of 67 output bytes in a Poll Command message, and a maximum of 67 input bytes in a Poll Response message.

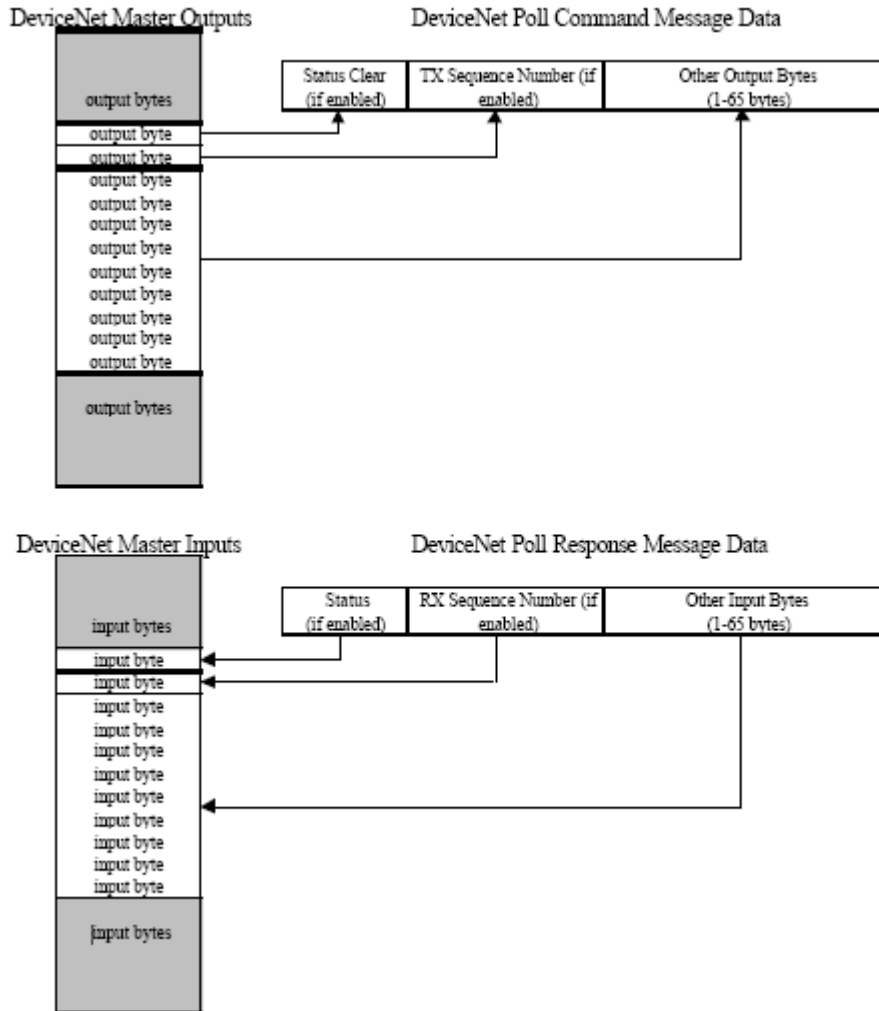


Figure 3 Mapping of DeviceNet Poll Command and Poll Response Data

3.1.1 Serial Channel Interface

The CDN466-X serial channel consists of an asynchronous serial transmitter and receiver. The serial interface is configured and controlled by the *Serial Stream Object*. The *Serial Stream Object* attributes configure the serial channel baud rate, parity, and flow control. This configuration applies to both the serial transmitter and receiver. The CDN466-X gateway has separate 64-byte serial transmit and receive FIFO buffers, allowing full duplex operation when supported by the physical layer media.

Devices communicating on an asynchronous serial link exchange information one bit at a time. Each bit is transmitted for a specific period of time, defined by the baud rate. Devices use internal timing circuitry to generate the baud rate. There is no clocking signal between devices to synchronize the serial data flow, hence the term *asynchronous* serial communications.

Serial data bits are organized into bytes. When a data byte is asynchronously transmitted, it is preceded by a start bit, followed by the data bits, an optional parity bit, and one or more stop bits. There can be a variable transmission delay between successive data bytes, since each byte is framed by its own start and stop bits. The receiver starts saving bits after it receives a valid start bit (0), and stops when it receives the expected number of stop bits (1). The data byte's least-significant bit is transmitted first (data bit 0), and the most-significant bit is last (data bit N).

[start bit] [data bit 0] [data bit 1] ... [data bit N] [optional parity bit] [stop bit(s)]

The parity bit detects single-bit errors in the transmission. The parity bit is calculated and inserted by the transmitter. The receiver calculates the parity of an incoming byte, and compares it to the parity bit sent by the transmitter. If the two bit values do not match, then at least one serial bit value was corrupted during transmission.

Flow control enables the receiving device to regulate the rate of incoming data. Hardware flow control uses RTS/CTS signals between the devices to control the rate of transmission. Software flow control uses serial characters XON/OFF to control the rate. CTS Detect Mode uses the CTS signal to enable serial communications. Flow control helps prevent data loss, if the receiving device cannot store incoming data fast enough, or if its Receive Buffer is full and cannot accept more data until existing data is processed.

The CDN466-X supports baud rates from 300 to 19200 bits per second. It supports 8 data bits with no parity, 7 data bits with parity, and 1 stop bit. CDN466-X supports no flow control, RTS/CTS, XON/XOFF, and CTS Detect Mode flow control options.

3.1.2 Status Overhead Byte

The gateway can be configured to return serial channel status information in the Poll Response message, and receive error-clearing commands in the Poll Command message. When enabled, the *Status* byte is returned as an input byte, and the *Status Clear* byte is received as an output byte. These bytes contain 8 status bits, defined below. Each bit represents either an error or state condition for the serial transmitter and receiver. Clearing the associated error bit in the *Status Clear* output byte will reset Receive Parity Error, Receive Buffer Overflow, Framing Error, and Transmit Buffer Overflow error conditions.

Table 1: Status and Status Clear Overhead Byte Bit-Mapping

Bit	Status Byte (1 st Input byte if enabled)	Status Clear (1 st output byte if enabled)
0	Transmit Channel Blocked	not used
1	Transmit Buffer Empty	not used
2	Receive Parity Error	Set = 0 to clear Receive Parity Error condition
3	Receive Buffer Empty	not used
4	Receive Buffer Overflow	Set = 0 to clear Receive Buffer Overflow condition
5	Framing Error	Set = 0 to clear Framing Error condition
6	Transmit Buffer Overflow	Set = 0 to clear Transmit Buffer Overflow condition
7	CTS Signal State (1 = asserted)	not used

A user application can use the Transmit Buffer Empty and Receive Buffer Empty status bits to monitor the transmitter and receiver states. However, the CDN466-X gateway also has three data synchronization features (*Receive Sequence Number*, *Transmit Sequence Number*, *Handshake Protocol*) that an application can use to better monitor the serial operations.

3.1.3 Receiving Message

The CDN466-X gateway has two modes for receiving serial data: *Stream Mode* and *Block Mode*. *Stream Mode* is best suited for applications with fixed-length serial messages, but it can also be used to capture any stream of serial data. *Block Mode* is intended for both fixed and variable-length message applications, where a *Delimiter* byte denotes the beginning or end of a message.

3.1.3.1 Stream Mode

Stream Mode saves all received message bytes in the Receive Buffer. There is no defined beginning or end to the message *stream*. The only limitation is the gateway must send bytes from the Receive Buffer to the DeviceNet master (Poll Response message) faster than it saves new message bytes in the Receive Buffer, or the 64-byte buffer may eventually overflow.

Incoming data stream: 0x45 0x62 0x02 0x31 0x32 0x33 0x42 0x45 0x02 0x42 0x43 0x44 ...

Stream Mode

Rx Message: 0x45 0x62 0x02 0x31 0x32 0x33 0x42 0x45 0x02 0x42 0x43 0x44 ...

3.1.3.2 Block Mode

Block Mode uses a configurable *Delimiter* byte to signal the start or end of a new message packet. The *Delimiter* cannot be used in any other part of the message, or it would be incorrectly interpreted as the start or end of a message. The gateway can be configured to save the *Delimiter* byte in the Receive Buffer, or discard it. In *Block Mode*, the gateway does not return any new message data to the DeviceNet master until the entire serial message has been received.

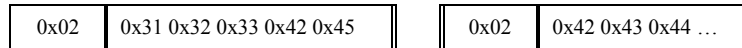
3.1.3.2.1 Pre-Delimiter Block Mode

Gateway expects the *Delimiter* at the start of a message. When a *Delimiter* byte is received, the gateway saves all subsequent bytes in the Receive Buffer until another *Delimiter* is received (signaling the start of another message), or until the *Maximum Receive Size* number of bytes has been saved. All bytes received after the *Maximum Receive Size* and before the next *Delimiter* are discarded. In this mode, the maximum number of bytes in a single message is defined by the *Maximum Receive Size* attribute.

Incoming data stream: 0x45 0x62 0x02 0x31 0x32 0x33 0x42 0x45 0x02 0x42 0x43 0x44 ...

Pre-Delimiter Mode

Rx Messages:



3.1.3.2.2 Post-Delimiter Block Mode

Gateway expects the *Delimiter* at the end of a message. The gateway saves all received bytes in the Receive Buffer until a *Delimiter* is received. In this mode, the maximum number of bytes in a single message is limited by the Receive Buffer size (64 bytes), not the *Maximum Receive Size* attribute.

Incoming data stream: 0x45 0x62 0x02 0x31 0x32 0x33 0x42 0x45 0x02 0x42 0x43 0x44 ...

Post-Delimiter

Rx Messages:



3.1.3.3 Returning Received Data

When the gateway receives a Poll Command message, it removes some or all of the bytes currently in the Receive Buffer and returns them as input bytes in a Poll Response message. The *Maximum Receive Size* attribute defines the maximum number of bytes that can be returned in a single Poll Response message. If the Receive Buffer contains more bytes than can fit into one Poll Response message, the remaining bytes are returned in subsequent Poll Response messages. *RX Message* is the string of valid message bytes returned in a single Poll Response message. The *RX Message* byte string can be formatted as either a Short_String (byte array with 1st byte = length) or a Byte Array (no length byte). The number of bytes in an *RX Message* string can be less than or equal to the *Maximum Receive Size*, but never larger. When the number is less, the remaining Poll Response input bytes are either padded or undefined.

Table 2: Poll Response Message Data

Status Byte (if Enabled)	Rx Sequence Number (if Enabled)	Length Byte (if Short String is used)	Input Data Byte(s)
-----------------------------	------------------------------------	--	-----------------------

In *Stream Mode*, the gateway will always try to fill Poll Response message with bytes from the Receive Buffer. The only time the *RX Message* size is less than the *Maximum Receive Size* is when there are no more bytes in the Receive Buffer.

In *Block Mode*, the gateway will not return any data in a Poll Response message unless it has a complete serial message saved in the Receive Buffer. If the message sizes are small, the gateway may have several messages saved in the Receive Buffer, depending upon how fast the DeviceNet master polls the gateway for data. The messages are returned one at a time in a Poll Response message, regardless of their size. If the message is large, then it is returned in multiple Poll Response messages.

Padding Message Data If the number of *RX Message* bytes currently in the Receive Buffer is less than the *Maximum Receive Size* number, then the remaining input bytes are undefined. The gateway can optionally fill the unused input bytes with a *Pad* character. The *Pad* characters can be added at the beginning or end of the message.

If configured for *Pre-Delimiter Block Mode* and the *Delimiter* byte is saved, the *Pad* characters are added either after the last valid message byte (right justification) or before the *Delimiter* byte (left justification).

If CDN466-X is configured for *Post-Delimiter Block Mode* and the *Delimiter* byte is saved, the *Pad* characters are added either before the first valid message byte (left justification), or after the last valid message byte but before the *Delimiter* byte (right justification).

3.1.3.3.1 Resending Received Data

The CDN466-X gateway can be configured to return received message bytes only once in a Poll Response message and return no data (null value) in subsequent Poll Response messages until new message bytes are received. For the Short_String data type, a null value consists of the length byte = 0. For the Byte Array data type, a null value consists of no data.

The gateway can also be configured to always return received message bytes in a Poll Response message. If no new bytes in the Receive Buffer, then the last received bytes are returned. If new bytes are in the Receive Buffer, then they are returned. The gateway provides *Receive Sequence Number* or *Handshake Protocol* synchronization options to indicate whether the returned bytes represent old or new data.

3.1.4 Transmitting Message

The *Serial Stream Object* receives output bytes (*TX Message*) from the DeviceNet master in a Poll Command message. It saves the output bytes in the Transmit Buffer, to be transmitted when the serial channel is available. The maximum number of bytes that can be sent in one Poll Command message is defined by the *Maximum Transmit Size* attribute. The Transmit Buffer can hold up to 64 bytes. Because the DeviceNet Polled I/O data exchange may occur many times faster than the transmission of serial data, the application may need to synchronize the transmit data exchange with the gateway.

The number of output bytes in the Poll Command message is fixed. The *Status Clear* and *Transmit Sequence Number* bytes are always sent, if enabled. The remaining number of bytes in the Poll Command is defined by the *Maximum Transmit Size* attribute. If the number of *TX Message* bytes sent is less than the *Maximum Transmit Size* number, then the remaining output bytes are undefined. The gateway uses the Short_String length to determine the valid number of bytes to transmit. If Byte Array format is used, all the bytes are transmitted.

Table 3: Poll Command Message Data

Status Clear Byte (if Enabled)	TX Sequence Number (if Enabled)	Length Byte (if Short String is used)	Output Data Byte(s)
-----------------------------------	------------------------------------	--	------------------------

3.1.5 Synchronization

To ensure that no information is lost between the gateway’s serial channel and the user application program, the CDN466-X has three synchronization options: *Receive Sequence Number*, *Transmit Sequence Number*, and *Handshake Protocol*.

3.1.5.1 Receive Sequence Number

When enabled, the gateway returns a *Receive Sequence Number* input byte in the DeviceNet Poll Response message. The 8-bit *Receive Sequence Number* is incremented by the gateway whenever it returns new data in the input bytes. The user application uses the *Receive Sequence Number* to signal the receipt of new message data. Valid numbers are 0-255.

3.1.5.2 Transmit Sequence Number

When enabled, the gateway receives a *Transmit Sequence Number* output byte in the DeviceNet Poll Command message. The gateway will not send the *TX Message* bytes out the serial channel unless the 8-bit *Transmit Sequence Number* is different than the last received value. Valid numbers are 0-255.

3.1.5.3 Synchronous Handshake Protocol

The gateway can be configured with a more robust transmit and receive synchronization process. The Handshake protocol requires the user application to acknowledge the receipt of new *RX Message* input bytes. The protocol also requires the gateway to acknowledge the transmission of the last *TX Message* output bytes. When enabled, both the *Receive Sequence Number* input byte and *Transmit Sequence Number* output byte are used. They are segmented into four 4-bit numbers, shown below. Valid numbers are 1 to 15, with 0 reserved to reset the gateway’s numbers

Table 4: Transmit Sequence Byte in Sync Mode

Bit 4-7 (upper nibble of Sequence Byte)	Bit 0-3 (lower nibble of Sequence Byte)
Receive Acknowledge Number	Transmit Request Number

The *Receive Request Number* is incremented by the gateway when it returns new *RX Message* input bytes in the Poll Response Message. The gateway will increment from 1 to 15, skipping 0. The user application acknowledges receipt of this *RX Message* by setting the *Receive Acknowledge Number* equal to the *Receive Request Number*. The updated *Receive Acknowledge Number* is sent back to the gateway in the next Poll Command Message. When the *Receive Acknowledge Number* equals the *Receive Request Number*, the gateway can return the next set of *RX Message*. If the user application sends 0 as the *Receive Acknowledge Number*, the gateway resets its *Receive Request Number* to 0.

Table 5: Receive Sequence Byte in Sync Mode

Bit 4-7 (upper nibble of Sequence Byte)	Bit 0-3 (lower nibble of Sequence Byte)
Receive Sequence Number	Transmit Acknowledge Number

The following ladder-logic rung shows how the user application program can monitor the gateway's *Receive Request Number* (RX Rqst Num), save the new *RX Message* bytes, and set *Receive Acknowledge Number* (RX Ack Num) equal to *Receive Request Number* (RX Rqst Num).

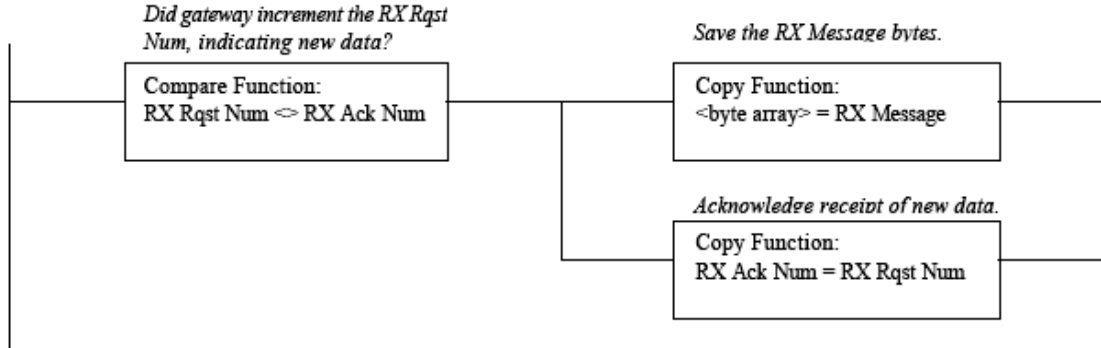


Figure 4 Ladder Logic rung to Acknowledge Receive Sequence Number

The *Transmit Request Number* is incremented by the user application when it sends new *TX Message* output bytes in the Poll Command Message. After the gateway transmits these *TX Message* bytes, it sets the *Transmit Acknowledge Number* equal to the *Transmit Request Number*, acknowledging the transmission. The updated *Transmit Acknowledge Number* is returned in the next Poll Response Message. If the user application sends 0 as the *Transmit Request Number*, the gateway ignores the *TX Message* output bytes and resets its *Transmit Acknowledge Number* to 0.

The following ladder-logic rungs show how the user application program writes a new *TX Message* value, increments the *Transmit Request Number* (TX Rqst Num), and waits for the *Transmit Acknowledge Number* (TX Ack Num) to equal the *Transmit Request Number* (TX Rqst Num). Note the application must wrap the *Transmit Request Number* from 15 to 1.

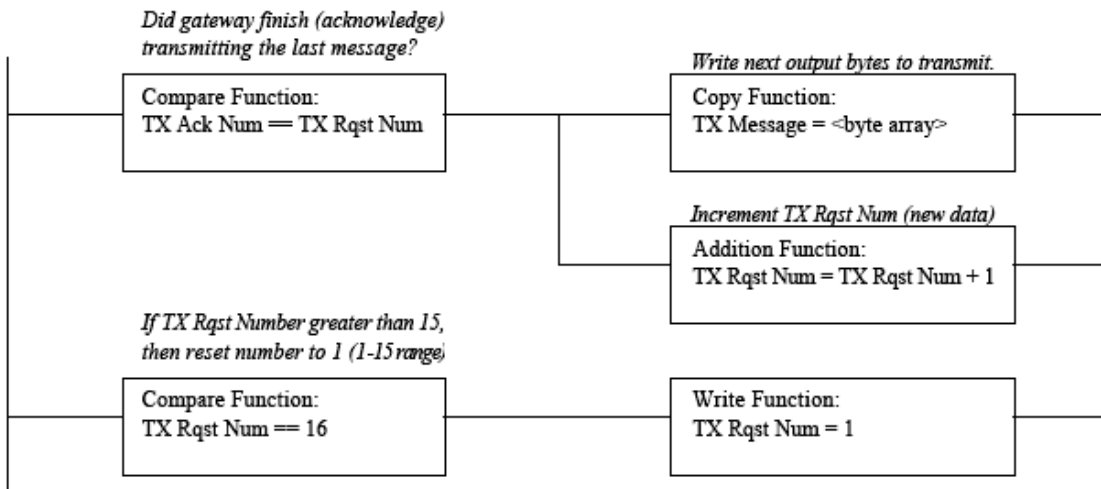


Figure 5 Ladder Logic Rung to Send New Transmit Sequence Number

4 Gateway Configuration

This chapter describes how to configure and operate the CDN466-X gateway. You configure the gateway by reading and writing attribute values over its DeviceNet interface. There are a variety of DeviceNet configuration tools available. Simple configuration tools use GET_ATTRIBUTE and SET_ATTRIBUTE explicit message commands to read and write attribute values, addressing each attribute by its Object, Instance, and Attribute numbers. This information is contained in Chapter 5. More sophisticated configuration tools use EDS files to simplify attribute configuration. You can configure the gateway using pull-down menus, buttons, and data entry fields from the gateway's Electronic Data sheet (EDS) file. Chapter 7 contains configuration examples using the Rockwell Software RSNetworx™ program.

4.1 Configure DeviceNet Interface

Set the DeviceNet Baud Rate and MAC ID Address using the rotary switches. Configure switches before connecting to the DeviceNet network. There is either a small triangular indicator or white indicator on the switch. Use a small screwdriver to align that indicator with the desired setting. Remove the CDN466-X cover if necessary to access the rotary switches.

4.1.1 DeviceNet Baud Rate Switch

Valid settings are 125K, 250K, 500K, or PGM. When PGM is selected, the CDN466-X uses the baud rate saved in its retentive memory. A valid baud rate must be stored before the PGM selection can be used. The baud rate is stored from the previous CDN466-X power cycle. It can also be set over the network (DeviceNet Object Baud Rate attribute).

Table 6: DeviceNet Baud Rate Switch Settings

Position	Settings
0	125K
1	250K
2	500K
3	Invalid
4	Invalid
5	Invalid
6	Invalid
7	Invalid
8	Invalid
9	PGM

4.1.2 MacID Switch Settings

The two MAC ID switches represent decimal numbers from 00 to 99. The LSB switch selects the *Ones* digit and the MSB switch selects the *Tens* digit. Valid MAC IDs are 00 to 63. Setting a MAC ID address greater than 63 forces the gateway to use the MAC ID saved in retentive memory. A valid MAC ID must first be stored before this feature can be used. The MAC ID is stored from the previous CDN466-X power cycle. It can also be set over the network (DeviceNet Object MAC ID attribute).

Table 7: MacID Switch Settings

MSB	LSB	Address
0	0 to 9	0 to 9
1	0 to 9	10 to 19
2	0 to 9	20 to 29
3	0 to 9	30 to 39
4	0 to 9	40 to 40
5	0 to 9	50 to 59
6	0 to 3	60 to 63
6	4 to 9	Stored Address
7	0 to 9	Stored Address
8	0 to 9	Stored Address
9	0 to 9	Stored Address

4.1.3 Serial Baud Rate Switch

The CDN466-X gateway has a rotary switch for the serial channel. This switch has different functions for the CDN466-X and CDN466-X models.

The CDN466-X model uses the rotary switch to select the RS232 channel baud rate. Valid settings are 300, 600, 1200, 2400, 4800, 9600, 19200 bits per second, and PRG (table below). When PRG is selected, the CDN466-X uses the *Baud Rate* attribute in the *Serial Stream Object*. A valid baud rate must be written over DeviceNet to this attribute.

Table 8: MacID Switch Settings

MSB	Setting
0	9600
1	4800
2	2400
3	1200
4	600
5	300
6	19200
7	Invalid
8	Invalid
9	PRG

4.1.4 Power Up Gateway

Connect the gateway to a DeviceNet network to power up the gateway.

4.1.4.1 DeviceNet Status LEDs

The CDN466-X gateway has two bi-color status LEDs (*NET* and *MOD*) that indicate operational status. During power-up, the LEDs cycle through a sequence of alternating red and green. After power-up, the *NET* LED should be flashing green (or solid green if allocated to a DeviceNet master) and the *MOD* LED should be solid green. If this does not occur, disconnect from DeviceNet and verify all the switch settings. See Chapter 8 for additional troubleshooting topics.

State	DeviceNet Status LED (<i>NET</i>)
Off	No power.
Flashing Red	Configuration error. Check DeviceNet switch settings.
Solid Red	Unrecoverable error.
Flashing Green	Device not allocated to a DeviceNet master.
Solid Green	Normal runtime, device allocated as a slave.

State	Module Status LED (<i>MOD</i>)
Off	No power.
Flashing Red	Configuration error. Check object attribute settings.
Solid Red	Unrecoverable error.
Flashing Green	Not defined.
Solid Green	Normal Operation.

4.1.4.2 Serial Channel Status LEDs

The gateway has two bi-color LEDs to indicate serial channel activity. The *TX* LED flashes green when a packet is being transmitted. The *RX* LED flashes green when a packet is being received. A fault is indicated by solid red. After power-up, both LEDs should be off.

State	Transmit Status LED (<i>TX</i>)
Off	No data being transmitted
Flashing Red	Not defined
Solid Red	Transmit error (parity or overrun error)
Flashing Green	Data being transmitted
Solid Green	Not defined

State	Receive Status LED (<i>RX</i>)
Off	No data being received
Flashing Red	Not defined
Solid Red	Receive error (parity or overrun error)
Flashing Green	Data being received
Solid Green	Not defined

4.1.4.3 Register EDS File

If using a DeviceNet configuration tool that supports Electronic Data Sheet (EDS) files, you should now register the gateway's EDS file with the software. The latest EDS file versions can be downloaded from www.mksinst.com. Select the EDS file that matches your gateway's part number and firmware version. Follow your configuration tool instructions to register EDS file.

4.1.5 Configure Serial Channel

The *Serial Stream Object* attributes control the CDN466-X serial channel. These settings apply to all serial transmit and receive operations. Before you can set or change any gateway configuration settings, make sure the gateway is not in the DeviceNet master scanlist.

Table 9: Serial Channel DeviceNet Attributes

Attributes	Name	Data Type	Value
3	Receive Data	Short_String or Byte Array	Received message data. Returned in Poll Response Message.
4	Transmit Data	Short_String or Byte Array	Message data to transmit. Received in Poll Command Message.
5	Status	USINT	Bit 0 – Transmit Channel Blocked Bit 1 – Transmit Buffer Empty Bit 2 – Receive Parity Error (set = 0 to clear) Bit 3 – Receive Buffer Empty Bit 4 – Receive Buffer Overflow Error (set = 0 to clear) Bit 5 – Framing Error (set = 0 to clear) Bit 6 – Transmit Buffer Overflow Error (set = 0 to clear) Bit 7 – CTS Signal State (1 = asserted)
6	Baud Rate	USINT	0 = 9600 bps 1 = 4800 bps 2 = 2400 bps 3 = 1200 bps 4 = 600 bps 5 = 300 bps 6 = 19200 bps
7	Parity	USINT	0 = no parity 5 = mark (force to 1) 1 = even parity 2 = odd parity 6 = space (force to 0)
8	Data Size	USINT	Read-only. 7 bits if parity enabled, 8 bits if no parity.
9	Stop Bits	USINT	Read-only. Fixed at 1 bit.
10	Flow Control	USINT	0 = none 1 = XON / XOFF 2 = CTS / RTS 4 = CTS Detect Mode
11	Receive Count	USINT	Number of bytes in Receive Buffer. Any write clears buffer.
12	Transmit Count	USINT	Number of bytes in Transmit Buffer. Any write clears buffer.

13	Maximum Receive Size	USINT	Defines the maximum #bytes returned by RX Message read.
14	Data Format	USINT	Bit 0 – String Format (0 = Short_String, 1 = Byte Array) Bit 1 – Strip Parity Bits (0 = retain, 1 = strip) Bit 2 – Pad Justification (0 = left justify, 1 = right justify) Bit 3 – Pad Received Message (0 = no, 1 = yes)
15	Block Mode	USINT	Bit 0 – Pre/Post Delimiter (0 = pre-delimiter, 1 = post-delimiter) Bit 1 – Strip Delimiter (0 = keep delimiter, 1 = strip delimiter) Bit 2 – Delimiter Enable (0 = no, 1 = yes) Bit 3 – Enable Receive Sequence Number (0 = no, 1 = yes) Bit 4 – Enable Transmit Sequence Number (0 = no, 1 = yes) Bit 5 – Resend (0 = no, 1 = yes) Bit 6 – Synchronization (0 = no, 1 = handshake protocol)
16	Delimiter	USINT	Delimiter byte value
17	Pad Character	CHAR	Pad byte value
18	Maximum Transmit Size	USINT	Defines the maximum # bytes that can be transmitted.
19	Idle String	Short_String	1-16 byte string transmitted when gateway receives a null Poll (no input bytes). Short_String length = 0 for no Idle String.
20	Fault String	Short_String	1-16 byte string transmitted when gateway's Polled I/O connection times out. Short_String length = 0 for no Fault String.
21	Status Enable	USINT	Set to any nonzero value to enable Status input byte.
22	Status Clear Enable	USINT	Set to any nonzero value to enable Status Clear output byte.

Receive Data – Data from the last valid message packet. Receive Data includes the Status and Receive Sequence Number bytes if enabled, and the RX Message bytes. The RX Message format is either Short_String or Byte Array, defined by Data Format attribute. If no message data is available, the RX Message will be a null packet or Short_String with length = 0. Receive Data is returned in the DeviceNet Poll Response Message.

Transmit Data – Data to transmit out the serial channel by the gateway. Transmit Data includes the Status Clear and Transmit Sequence Number bytes if enabled, and the TX Message bytes. Format is either Short_String or Byte Array, defined by Data Format attribute. Transmit Data is typically received in the DeviceNet Poll Command Message. Reading Transmit Data returns the last byte in the Transmit Buffer.

Status – Contains bit-mapped serial channel status and error bits for transmit and receive operations. Clearing the bits indicated will clear the error condition.

Baud Rate – Sets the serial channel's data or baud rate. Enter number from 1-6 to select corresponding baud rate value. For CDN466-X, the RS232 Baud Rate switch must be set to PRG before this attribute can be used to set the baud rate.

Data Size – Read-only attribute indicates number of data bits in one serial byte.

This number does not include start, parity, or stop bits. If parity is enabled, 7 data bits are used. If no parity, 8 data bits are used.

Stop Bits – Read-only attribute indicates number of stop bits in one serial byte. Fixed at 1.

Flow Control – Selects the method of flow control used across the serial interface.

NONE means there is no flow control over the serial data exchange. The transmitting device can overflow the receiving device's buffer.

XON/XOFF is a software flow control option. Receiving device sends an XOFF character to the transmitting device when its buffer is full, stopping further transmission. It sends an XON character when it can again receive data. The XOFF and XON characters are not saved as message data.

CTS/RTS is an RS232 hardware flow control option, available only on the CDN466-X gateway. The RTS is an output and CTS is an input signal. The gateway keeps RTS active (low) when it can receive data. It only transmits data when CTS is active (low).

CTS Detect Mode is an RS232 hardware flow control option, available only on the CDN466-X gateway. When CTS is asserted, the CDN466-X serial channel can transmit and receive. When CTS is not asserted, the CDN466-X serial channel is disabled and Receive Buffer cleared.

Receive Count – Number of bytes currently available in the Receive Buffer. Writing any value to this attribute will clear the Receive Buffer.

Transmit Count – Number of bytes currently in the Transmit Buffer. Writing any value to this attribute will clear the Transmit Buffer.

Maximum Receive Size – Defines the maximum number of data bytes to be returned when the Receive Buffer is read using either an Explicit Message or a Poll Response Message.

Data Format – Control byte that defines the format of the TX Message and RX Message bytes transferred across DeviceNet.

Bit 3 selects whether the *RX Message* bytes are padded with the Pad bytes. Set this bit = 1 to enable. If there are not enough message bytes in the Receive Buffer to fill up the *RX Message* input bytes, then Pad characters are added at either the beginning or end of the message bytes.

Bit 2 selects whether Pad bytes are added at the beginning of the message (0 = left justify) or at the end of the message (1 = right justify). This bit is used only if the Pad option is enabled.

Bit 1 defines whether the gateway saves the parity bit in received message bytes (set = 0), or if the gateway forces the parity bit to 0 in received message bytes (set = 1). This is typically used when receiving 7-bit ASCII data.

Bit 0 defines String Format for *TX Message* and *RX Message* byte strings. Set to 0 for Short_String format, and 1 for Byte Array format. Short_String defines the first byte as an explicit length byte, containing the number of bytes that follow. Byte Array has an implied length, derived from the *Maximum Receive Size* attribute.

Block Mode – Control byte that defines the serial receive mode, synchronization mode, and resend message option.

Bit 6 enables the Handshake Protocol synchronization option. When enabled, the *Receive Sequence Number* byte is added to *Receive Data* input bytes, and the *Transmit Sequence Number* byte is added to the *Transmit Data* output bytes.

Bit 5 enables the resend message option. When enabled, the gateway continuously returns *RX Message* data in the Poll Response message. If no new data has been received, then the last data bytes are returned.

Bit 4 enables the *Transmit Sequence Number* synchronization option. When enabled, the *Transmit Sequence Number* byte is added to the *Transmit Data* output bytes.

Bit 3 enables the *Receive Sequence Number* synchronization option. When enabled, the *Receive Sequence Number* byte is added to the *Receive Data* output bytes.

Bit 2 selects the serial receive mode. Set = 0 for *Stream Mode*, and set = 1 for *Block Mode*.

Bit 1 selects whether the Delimiter is saved in the Receive Buffer (set = 0), or it is discarded (set = 1). This bit is only used when *Block Mode* is enabled.

Bit 0 selects *Pre-Delimiter Mode* (set = 0) or *Post-Delimiter Mode* (set = 1). This bit is only used when *Block Mode* is enabled.

Delimiter – Byte value used to indicate the start of a new message (Pre-Delimiter Mode), or the end of a received message (Post-Delimiter Mode). This attribute is only used in Block Mode.

Pad Character – Byte value used to pad the RX Message bytes.

Maximum Transmit Size – Defines the maximum size of TX Message output bytes, or the maximum number of data bytes to be transmitted across the RS232 channel from one Poll Command message.

Idle String – Defines the byte string that is transmitted when the gateway receives a null Poll (no input bytes, or a Short_String value with length = 0). Enter the byte string in Short_String data format, with 1st byte = string length. Set the length byte to 0 if you don't want to transmit an Idle String. The Idle String can be from 0 to 16 bytes long, not counting Short_String length byte.

Example Idle String is [0x01 0x41], where string length is 1 and data byte is 0x41 ('A'). You must use the RSNetworx™ Class Instance Editor (Set Attribute Single command) to write a Short_String attribute value.

Fault String – Defines the byte string that is transmitted when the gateway's connection to the DeviceNet master times out. Enter the byte string in Short_String data format, with 1st byte = string length. Set the length byte to 0 if you don't want to transmit a Fault String. The Fault String can be from 0 to 16 bytes long, not counting Short_String length byte.

Example Fault String is [0x02 0x42 0x43], where string length is 2 and data bytes are 0x42 ('B') and 0x43 ('C'). You must use the RSNetworx™ Class Instance Editor (Set Attribute Single command) to write a Short_String attribute value.

Status Enable – Write any nonzero value to include the Status byte in Receive Data input bytes.

Status Clear Enable – Write any nonzero value to include the Status Clear byte in Transmit Data output bytes.

4.1.6 Configure DeviceNet Master Scanlist

You must calculate the number of input and output bytes required by your CDN466-X configuration before you can add the gateway to the DeviceNet master scan list. You need to configure the DeviceNet master to send the specific number of output bytes in its Poll Command Message, and receive the specific number of input bytes in the gateway’s Poll Response Message. Once the input and output bytes are mapped in the DeviceNet master, the user application program will be able to read and write data values to the input and output bytes.

Poll Consume Size The *Poll Consume Size* is the size (in bytes) of the Poll Command Message data field that is sent by the DeviceNet master to the CDN466-X.

Poll Command data:

[Status Clear byte][Transmit Sequence Number byte][Short_String length byte][TX data bytes (0-64)]

The first 3 bytes are present if enabled. The following equation is used to calculate the CDN466-X *Poll Consume Size*. Only include the overhead bytes that are enabled.

Table 10: Poll Command Data Format

+	<i>Status Clear byte (if enabled)</i>	1
+	<i>Transmit Sequence Number byte (if enabled)</i>	1
+	<i>Short_String length byte (if short String format)</i>	1
+	<i>Maximum Transmit Size (0 to 64 bytes)</i>	0-64

Poll Produce Size The *Poll Produce Size* is the size (in bytes) of the Poll Response Message data field that is sent from the CDN466-X to the DeviceNet master.

Poll Response data:

[Status byte][Receive Sequence Number byte][Short_String length byte][RX data bytes (0-64)]

The first 3 bytes are present if enabled. The following equation is used to calculate the CDN466-X *Poll Produce Size*. Only include the overhead bytes that are enabled.

Table 11: Poll Response Data Format

+	<i>Status byte (if enabled)</i>	1
+	<i>Receive Sequence Number byte (if enabled)</i>	1
+	<i>Short_String length byte (if short String format)</i>	1
+	<i>Maximum Receive Size (0 to 64 bytes)</i>	0-64

5 DeviceNet Specification

DeviceNet Message Types

As a group 2 slave device the CDN466-X supports the following message types.

CAN IDENTIFIER	GROUP 2 Message Type
10xxxxxx111	Duplicate MACID Check Message
10xxxxxx110	Unconnected Explicit Request Message
10xxxxxx101	Master I/O Poll Command Message
10xxxxxx100	Master Explicit Request Message

xxxxxx = Node Address

DeviceNet Class Services

As a group 2 slave device the CDN466-X supports the following class services and instance services.

SERVICE CODE	SERVICE NAME
05 (0x05)	Reset
14 (0x0E)	Get Attribute Single
16 (0x10)	Set Attribute Single
75 (0x4B)	Allocate Group 2 Identifier Set
76 (0x4C)	Release Group 2 Identifier Set

DeviceNet Object Classes

The CDN466-X device supports the following DeviceNet object classes.

CLASS CODE	OBJECT TYPE
01 (0x01)	Identity
02 (0x02)	Router
03 (0x03)	DeviceNet
04 (0x04)	Assembly
05 (0x05)	Connection
64 (0x40)	User defined serial interface

Identity Object

Class Code: 01 (0x01)

The Identity Object is required on all devices and provides identification of and general information about the device.

Identity Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	7

Identity Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value
1	Get	Vendor	UINT	59
2	Get	Product Type	UINT	12 = Communications
3	Get	Product Code	UINT	1
4	Get	Revision	STRUCT OF	
		Major Revision	USINT	4
		Minor Revision	USINT	0
5	Get	Device Status	UINT	(1)
6	Get	Serial Number	UINT	Unique Serial Number for every Device
7	Get	Product Name	STRUCT OF	
		Length	USINT	6
		Name	STRING [6]	CDN066

Common Services

Service Code	Class	Instance	Service Name
05 (0x05)	No	Yes	Reset
14 (0x0E)	Yes	Yes	Get_Attribute_Single

(1) Device Status

bit 0	owned	0=not owned 1=owned (allocated)
bit 1	reserved	0
bit 2	configured	0
bit 3	reserved	0
bit 4-7	vendor specific	0
bit 8	minor cfg fault	0=no fault 1=minor fault
bit 9	minor dev.fault	0=no fault 1=minor device fault
bit 10	major cfg.fault	0=no fault 1=major cfg. fault
bit 11	major dev.fault	0=no fault 1=major device fault
bit 12-15	reserved	0

Router Object Class Code: 02 (0x02)

The Message Router Object provides a messaging connection point through which a Client may address a service to any object class or instance residing in the physical device.

Router Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	2

Router Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value
2	Get	Number of Connections	UINT	2

Common Services

Service Code	Class	Instance	Service Name
14 (0x0E)	Yes	Yes	Get_Attribute_Single

DeviceNet Object

Class Code: 03 (0x03)

DeviceNet Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2

Router Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value
1	Get/Set	MACID	USINT	(1)
2	Get/Set	Baud Rate	USINT	(2)
3	Get/Set	Bus Off Interrupt	BOOL	(3)
4	Get/Set	Bus Off Counter	USINT	(4)
5	Get/Spc	Allocation Information	STRUCT of	(5)
		Choice Byte	BYTE	
		Master Node Addr.	USINT	

Common Services

Service Code	Class	Instance	Service Name
14 (0x0E)	Yes	Yes	Get_Attribute_Single
16 (0x10)	No	Yes	Set_Attribute_Single
75 (0x4B)	No	Yes	Allocate Master/Slave
76 (0x4C)	No	Yes	Release Master/Slave

(1) Settable only if the MacID switches are set to a value greater than 63. Value returned will be switch value if less than 64 or the last value set.

(2) Settable only if the Baud Rate switch is set to a value greater than 2. Value returned will be switch value if less than 4 or the last value set.

Switch/Value	Speed
0	125 kbits
1	250 kbits
2	500 kbits
3	Software settable

(3) Bus Off Interrupt (BOI) determines action if Bus Off state encountered. Following values are supported:

BOI	Action
0	Hold chip in OFF state (default)
1	If possible reset CAN chip

(4) Bus Off Counter will be forced to 0 whenever set regardless of the data value provided.

(5) Allocation_byte

bit 0	explicit set to 1 to allocate
bit 1	polled set to 1 to allocate
bit 2	strobed (not supported)
bit 3-7	reserved (always 0)

Assembly Object

Class Code: 04 (0x04)

The Assembly Objects bind attributes of multiple objects to allow data to or from each object to be sent or received over a single connection.

Assembly Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Class ID	UINT	2

Assembly Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value
3	Get	Data Stream (Input)	see notes	(1)

Assembly Object, Instance 2 Attributes

Attribute	Access	Name	Type	Value
3	Get/Set	Data Stream (Output)	see notes	(2)

Common Services

Service Code	Class	Instance	Service Name
14 (0x0E)	Yes	Yes	Get_Attribute_Single
16 (0x10)	Yes	Yes	Set_Attribute_Single

(1) The input data stream is structured as either an array of bytes or as a SHORT_STRING consisting of a single byte length field and 'n' data bytes. Refer to the serial stream object class 64 for further information.

(2) The output data stream is structured as either an array of bytes or as a SHORT_STRING consisting of a single byte length field and 'n' data bytes. Refer to the serial stream object class 64 for further information.

Connection Object

Class Code: 05 (0x05)

The Connection Objects manage the characteristics of each communication connection. As a Group II Only Slave device the unit supports one explicit message connection and a POLL message connection.

Connection Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1

Connection Object, Instance 1 Attributes (Explicit Message)

Attribute	Access	Name	Type	Value
1	Get	State	USINT	(1)
2	Get	Instance Type	USINT	0 = Explicit Message
3	Get	Transport Class Trigger	USINT	0x83
4	Get	Production Connection	UINT	(2)
5	Get	Consumed Connection	UINT	(2)
6	Get	Initial Comm. Char.	USINT	0x21
7	Get	Production Size	UINT	67
8	Get	Consumed Size	UINT	71
9	Get/Set	Expected Packet Rate	UINT	Default 2500 msec
12	Get/Set	Timeout Action	USINT	(3)
13	Get	Prod. Path Length	USINT	0
14	Get	Production Path		(null)
15	Get	Cons. Path Length	USINT	0
16	Get	Consumed Path		(null)
17	Get	Production Inhibit	UINT	0

Connection Object, Instance 2 Attributes (POLL connection)

Attribute	Access	Name	Type	Value
1	Get	State	USINT	(1)
2	Get	Instance Type	USINT	1 = I/O Message
3	Get	Transport Class Trigger	USINT	0x82
4	Get	Production Connection	UINT	(2)
5	Get	Consumed Connection	UINT	(2)
6	Get	Initial Comm. Char.	USINT	0x1
7	Get	Production Size	UINT	See Stream Object
8	Get	Consumed Size	UINT	See Stream Object
9	Get/Set	Expected Packet Rate	UINT	Default 2500 msec

12	Get/Set	Timeout Action	USINT	(3)
13	Get	Prod. Path Length	USINT	6
14	Get	Production Path	STRUCT of	
		Log. Seg., Class	USINT	0x20
		Class Number	USINT	0x04
		Log.Seg., Instance	USINT	0x24
		Instance Number	USINT	0x01
		Log.Seg., Attribute	USINT	0x30
		Attribute Number	USINT	0x03
15	Get	Cons. Path Length	USINT	6
16	Get	Production Path	STRUCT of	
		Log. Seg., Class	USINT	0x20
		Class Number	USINT	0x04
		Log.Seg., Instance	USINT	0x24
		Instance Number	USINT	0x02
		Log.Seg., Attribute	USINT	0x30
		Attribute Number	USINT	0x03
17	Get	Production Inhibit	UINT	0

Common Services

Service Code	Class	Instance	Service Name
05 (0x05)	Yes	Yes	Reset
14 (0x0E)	Yes	Yes	Get_Attribute_Single
16 (0x10)	No	Yes	Set_Attribute_Single

(1) Connection States:

- 0 = non-existent
- 1 = configuring
- 3 = established
- 4 = timed out

(2) Connection ID's:

Connection 1 Produced Connection ID: 10xxxxxx011
Connection 1 Consumed Connection ID: 10xxxxxx100
Connection 2 Produced Connection ID: 01111xxxxxx
Connection 2 Consumed Connection ID: 10xxxxxx101

xxxxxx = Node Address.

(3) Watch Dog TimeOut Activity:

0 = Timeout (Explicit Messaging default)
1 = Auto Delete
2 = Auto Reset (I/O Message default)

(4) If no data is available during the poll response a 0 length (null) packet is returned.

User Defined (Serial Stream) Object

Class Code: 64 (0x40)

The Serial Stream Object model supports a bi-directional serial stream of data. The object includes the transmit FIFO, the receive FIFO and the serial channel configuration attributes.

Serial Stream Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	24

Serial Stream Object, Instance 1 Attributes

Serial Stream Object			Class Code 64 (0x40)	
Class Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	22
Instance Attribute	Access	Name	Type	Value
3	Get	Receive Data	Data Format	Received message data. Returned in Poll Response
4	Get/Set	Transmit Data	Data Format	Message data to transmit. Received in Poll Command.
5	Get/Set	Status	USINT	Bit 0 – Transmit Channel Blocked Bit 1 – Transmit Buffer Empty Bit 2 – Receive Parity Error (0 to clear) Bit 3 – Receive Buffer Empty Bit 4 – Receive Buffer Overflow Error (0 to clear) Bit 5 – Framing Error (0 to clear) Bit 6 – Transmit Buffer Overflow Error (0 to clear) Bit 7 – CTS Signal State (1 = asserted)
6	Get/Set	Baud Rate	USINT	0 = 9600 bps 1 = 4800 bps 2 = 2400 bps 3 = 1200 bps 4 = 600 bps 5 = 300 bps 6 = 19200 bps
7	Get/Set	Parity	USINT	0 = no parity 1 = even parity 2 = odd parity 5 = mark (force to 1) 6 = space (force to 0)

8	Get	Data Size	USINT	7 (parity enabled) or 8 (no parity)
9	Get	Stop Bits	USINT	1
10	Get/Set	Flow Control	USINT	0 = none 1 = XON / XOFF 2 = CTS / RTS 4 = CTS Detect Mode
11	Get/Set	Receive Count	USINT	Number of bytes in Receive Buffer. Write to clear.
12	Get/Set	Transmit Count	USINT	Number of bytes in Transmit Buffer. Write to clear
13	Get/Set	Maximum Receive Size*	USINT	Maximum # bytes returned by Receive Buffer read.
14	Get/Set	Data Format *	USINT	Bit 0 – String Format (0 = Short_String, 1 = Array) Bit 1 – Strip Parity Bits (0 = retain, 1 = strip) Bit 2 – Pad Justification (0 = left, 1 = right) Bit 3 – Pad Received Message (0 = no, 1 = yes)
15	Get/Set	Block Mode *	USINT	Bit 0 – Pre/Post Delimiter (0 = Pre-, 1 = Post-) Bit 1 – Strip Delimiter (0 = keep, 1 = strip) Bit 2 – Delimiter Enable (0 = no, 1 = yes) Bit 3 – Enable Receive Sequence Number Bit 4 – Enable Transmit Sequence Number Bit 5 – Resend (0 = no, 1 = yes) Bit 6 – Synchronization (0 = no, 1 = yes)
16	Get/Set	Delimiter	USINT	Delimiter byte value
17	Get/Set	Pad Character	CHAR	Pad byte value
18	Get/Set	Maximum Transmit Size *	USINT	Defines maximum # bytes that can be transmitted.
19	Get/Set	Idle String	Short_String	Byte string transmitted when gateway receives null Poll (no input bytes). Length = 0 for no Idle String.
20	Get/Set	Fault String	Short_String	Byte string transmitted when gateway's Polled I/O connection times out. Length=0 for no Fault String
21	Get/Set	Status Enable *	USINT	Nonzero value enables Status input byte.
22	Get/Set	Status Clear Enable	USINT	Nonzero value enables Status Clear output byte.

Common Services

Service Code	Class	Instance	Service Name
5 (0x05)	No	Yes	Reset
14 (0x0E)	Yes	Yes	Get_Attribute_Single
16 (0x10)	No	Yes	Set_Attribute_Single

*** Items indicated with an asterisk may affect the Produced or Consumed size of the Poll Connection.**

Received Data (Attribute 3) and Transmitted Data (Attribute 4) are either an array of bytes or a DeviceNet defined SHORT_STRING, consisting of a length byte followed by the specified number of valid data bytes. The format used is determined by the Data Format parameter. Note that reading of the Transmit Data will return a single byte, indicating the last byte of the FIFO.

When a packet is received with 0 data bytes no data is transmitted. If the transmit FIFO does not have sufficient room for the packet no response packet is generated.

When data is read the response packet will be either an array of bytes or a SHORT_STRING. If no data is available either a NULL packet or an array with a length byte of 0 is returned.

Status information (Attribute 5) indicates whether data transfer errors have occurred. It is bit mapped as follows:

Bit	Interpretation
0	Transmit channel blocked
1	Transmit FIFO empty
2*	Receive Parity error
3	Receive FIFO empty
4*	Receive Overflow
5*	Framing Error
6*	Transmit FIFO Overflow
7	RESERVED

* Writing any value to the Status field will clear the error bits.

Baud Rate (attribute 6) may be set by software.

Baud Rate	Interpretation
0	9600 baud
1	4800 baud
2	2400 baud
3	1200 baud
4	600 baud
5	300 baud
6	19.2 Kbaud

Parity (attribute 7) may be set by software. Note that setting the parity to 0 forces the data length size to 8. Setting the parity to non-zero forces the data length size to 7.

Parity	Interpretation
0	No parity
1	Even parity
2	Odd parity
3	N/A
4	N/A
5	Force to 1
6	Force to 0

Data size (Attribute 8) is read only. The CDN466-X serial channel always processes 8 information bits. If parity is set to 0 (no parity) 8 data bits are transmitted/received. If the parity is set to a non-zero value then only 7 data bits are transmitted and the 8th bit is used for the parity bit. The Data Size field is read only.

Stop bits (Attribute 9) is read only. The CDN466-X serial channel always operates with 1 stop bit. The Stop Bits field is read only and fixed at 1 stop bit.

Flow control (Attribute 10) may be set by software.

Flow Control	Interpretation
0	No flow control
1	X-ON/X-OFF flow control (only in 4-wire mode)
2	CTS/RTS
4	Auto Detect Mode

(4 Wire Mode Only) If the flow control is set to 1 the ASCII standard X-OFF (CTRL S) character will force the transmit function to block. Characters will be buffered in the transmit FIFO until the transmitter is re-enabled using the X-ON (CTRL Q) character. Note that the CTRL-S and CTRL-Q characters will be stripped from the incoming data stream, making this protocol unsuitable for binary data transmission.

When the receive FIFO is full the CDN466-X will transmit an X-OFF character. An X-ON character is transmitted when the number of characters in the receive FIFO drops below 50%.

Receive Count (Attribute 11) indicates the number of characters currently available in the receive FIFO. Writing any value will flush the receive FIFO.

Transmit Count (Attribute 12) indicates the number of characters currently in the transmit FIFO. Writing any value will flush the transmit FIFO.

Maximum Receive Size (Attribute 13) indicates the maximum number of data bytes to be returned when the receive FIFO is read (attribute 3) either using EXPLICIT messages or through the POLL connection. Setting this attribute will automatically reset the Produced Connection size as:

$$\begin{aligned}
 \text{Connection size} &= \text{Max Rcv Size (Maximum size is 64)} \\
 &+ 1 \text{ (if Status Byte enabled)} \\
 &+ 1 \text{ (if String Format enabled)} \\
 &+ 1 \text{ (if Receive Seq. Num. enabled)}
 \end{aligned}$$

The maximum connection size is 67 bytes.

This attribute affects the produce size, and is only settable when the the poll connection is in the non-established state.

Data Format (Attribute 14) control byte determines the type of data strings transferred over the DeviceNet channel which may be either an array of bytes or a DeviceNet defined SHORT_STRING, consisting of a length byte followed by the specified number of valid data bytes. Note that the data length byte does not appear on the serial channel. The Data format control byte also determines whether the parity information is retained

in the receive FIFO. If the bit is cleared then the parity information is retained. If set, the parity information is overwritten with a 0, ensuring that only valid ASCII characters (0-7FH) appear in the FIFO.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	PADR	PL/R	Strip Parity	String Format

String Format

0
1

Interpretation

Process FIFO packets as SHORT_STRING variables
Process FIFO packets as an array of bytes. The array length implicitly defines the number of valid bytes.

Strip Parity

0
1

Interpretation

Retain Parity information in receive FIFO
Set MSB of receive FIFO data to 0

PL/R

0
1

Interpretation

Left justify received character string if PADR set
Right justify received character string if PADR set

PADR

0
1

Interpretation

Do not attempt to PAD received characters
Pad received characters strings with PADCHAR

If the PADR bit is set in block mode with the Strip Delimiter bit clear the Pad characters will be inserted between the last valid data bit and the end of the packet.

[This attribute affects the produce and consume size, and is only settable when the the poll connection is in the non-established state.](#)

Block Mode (Attribute 15) control byte determines the whether the unit prepares the RS422/485 serial stream, whether block sequence numbers are pre-pended to the DeviceNet packets and whether received data is retransmitted on subsequent POLL requests. The control byte has the following format:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	Sync	ReSend	Enable Xmit Seq. Number	Enable Rcv. Seq. Number	Delimiter Enable	Strip Delimiter	Pre/Post Delimiter

Pre/Post Delimiter	Interpretation
0	Delimiter (if enabled) occurs at the end of the packet.
1	Delimiter (if enabled) occurs at the start of the packet. The packet length is limited to the <Max Receive Size> length. Excess characters are discarded.

Strip Delimiter	Interpretation
0	The delimiter character appears in the response packet.
1	The delimiter character is removed from the response packet.

Delimiter Enable	Interpretation
0	Disable the delimit character function
1	Enable the delimit character function

Enable Rcv.Seq.Num	Interpretation
0	Disable the receive sequence number
1	Each response packet will have a sequential number pre-pended to allow the scanner to detect new response data.

Enable TX Sequence Number	Interpretation
0	Disable the transmit sequence number
1	The first byte of the poll request must contain a number different than the last request to allow the updating of the scanner data field without generating erroneous data on the RS422/RS485 data.

Resend	Interpretation
0	Valid data is only sent once
1	Valid data is resent during subsequent Poll requests until a new string of valid data is received on the RS422/RS485 serial channel.

Sync	Interpretation
0	Do not apply synchronous Hand-shake protocol
1	Apply synchronous Hand-shake protocol

When a CDN466-X is configured with both <string> formatting and sequence numbers the sequence number is applied as the first byte and the string length information is contained in the second data byte.

If the delimiter function is enabled the receive packet size may have an affect on the data responses. If the Post Delimiter field is zero the CDN466-X will not transmit any response data until the delimiter character is detected or until <receive data size> bytes are available. If the receive data size is set less than the number of available characters the first poll response will contain the first <receive data size> bytes and the second poll response will receive the remaining characters up to delimiter.

If the Post Delimiter field is 1 the CDN466-X will not transmit any response data until a delimiter is detected AND:

- 1) more than <receive data size> bytes have been received, or
- 2) another delimiter is detected

Characters in excess of the receive data size are discarded.

If the Resend bit is set the device will resend data on subsequent Poll requests until another valid data packet has been received.

The Sync bit enables the Synchronous Hand-shake protocol which provides further control over the sequence numbers during Poll Request/Response transactions to allow the Master to determine if

- a) a previous Poll Request packet has been accepted and
- b) the current Poll Response represents a new data string.

When the Sync bit is set to 1 the *TX Sequence Number* and *RX Sequence Number* bits will be forced to 1.

The TX Sequence Number is received in the Poll Request and is interpreted as 2 four bit numbers:

Bit Numbers 4-7	Bit Numbers 0-3
Receive Acknowledge Number	Transmit Request Number

The *Transmit Request Number* acts in the same way as the TX Sequence Number described above. The CDN466-X will ignore any data in the Poll Request until the *Transmit Request Number* is different than previously received *Transmit Request Number*. If a value of 0 is received the current data (if any) will be ignored. A 0 acts as a ‘reset’ function for the *Transmit Request Number*.

The *Receive Acknowledge Number* is compared against the *Receive Request Number* (see below) and if equal it releases the current receive data buffer, allowing the CDN466-X to send new information. A value of 0 will reset the *Receive Request Number*, acting as a reset function.

The *RX Sequence Number* is transmitted in the Poll Response and is interpreted as 2 four bit numbers:

Bit Numbers 4-7	Bit Numbers 0-3
<i>Receive Request Number</i>	<i>Transmit Acknowledge Number</i>

The *Transmit Acknowledge Number* will be the same value as the most recently processed *Transmit Request Number*. When a poll request packet is received the *Transmit Request Number* is compared to the last *Transmit Acknowledge Number* and if different the data contained in the poll request is transmitted (see above). The *Transmit Request Number* is then transferred to the *Transmit Acknowledge Number* to notify the Master that the transaction has been processed.

The *Receive Request Number* is used by the CDN466-X to indicate to the Master that the poll response contains new data. The CDN466-X will increment the most previous *Receive Acknowledge Number* (see above) and return it in the poll response if new data is available. Note that the CDN466-X will generate numbers in the range 1..15, reserving 0 as the reset value.

The Sync mode is typically used with a ‘Scanner’ that generates continuous poll requests. During the first poll request (possibly no valid data) the *TX Sequence Number* should be set to a value of 00, resetting the receive handshaking logic on the CDN466-X. If no receive data is available the poll response will have the *RX Sequence Number* set to 0. If data is available the CDN466-X will generate a poll response with the *RX Sequence Number* set to 1 with the associated data contained in the response packet. Further data will be buffered until the Scanner generates a poll request with a *TX Sequence Number* with a value of 1, acknowledging the receipt and processing of the previous poll. The Scanner should increment the *Receive Acknowledge Number* after processing each poll response, wrapping from 15 to 1.

During transmission, the scanner application code may build the request message in memory and then increment the *Transmit Request Number* (1..15). This allows the background scanner function to send ‘partially complete’ poll requests without generating extraneous RS422/RS485 transmissions. When the scanner application code detects that the *Transmit Acknowledge Number* received as part of the poll response matches the previous *Transmit Request Number* it indicates that the scanner has successfully transmitted the previous poll data and the application may proceed to build new RS422/RS485 transmit data.

This attribute adds 1 byte to the produce and consume size, and is only settable when the poll connection is in the non-established state.

Delimiter character (Attribute 16) determines the start or end of packet character for the RS422/RS485 channel. It is only effective if the Delimiter Enable bit in the Block Control byte is set.

(17) The Pad Char (Attribute 17) is used to pad string formatted receive data. It is typically set to ASCII <space> (020H) or an ASCII <null> (0).

(18) The Maximum Transmit Size (Attribute 18) indicates the maximum number of data bytes to be transmitted across the RS422/RS485 channel. Setting this attribute will automatically reset the Poll Consumed Connection size as:

$$\begin{aligned}
 \text{Connection size} &= \text{Max Xmt Size (Maximum value 64)} \\
 &+ 1 \text{ (if Status Clear enabled)} \\
 &+ 1 \text{ (if String Format enabled)} \\
 &+ 1 \text{ (if Transmit Seq. Num. enabled)}
 \end{aligned}$$

The maximum connection size is 67 bytes.

This attribute affects the consume size, and is only settable when the poll connection is in the non-established state.

Idle String (Attribute 19) will be transmitted on the RS422/RS485 serial channel if the device receives a 'receive_idle' (null Poll). If the string length by is set to 0 no data will be transmitted.

Fault String (Attribute 20) will be transmitted on the RS422/RS485 serial channel if the device experiences a connection timeout. If the string length by is set to 0 no data will be transmitted.

Status Enable (Attribute 21) inserts the serial status (Class 64 Instance 1 Attribute 5) as the first byte of the poll response when set to a non-zero value.

This attribute adds 1 byte to the produce size, and is only settable when the poll connection is in the non-established state.

Status Clear Enable (Attribute 22) allows the first byte in the poll command to clear the status byte (when status clear byte, the first byte in the poll is not equal to 0) or not change the status (status clear byte, the first byte in the poll is =0).

This attribute adds 1 byte to the consume size, and is only settable when the poll connection is in the non-established state.

6 Quick Star Guide

This section describes how to install and connect the CDN466-x gateway to a DeviceNet network and to a serial device.



Caution Follow all applicable electrical codes in your area when mounting and wiring any electrical device.

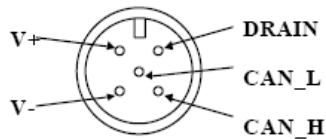
6.1 Wiring

Communications connections:

The CDN466-X requires two connections for communications – one to the DeviceNet network (male 5-pin micro connector) and one to the target serial device (male DB9 connector). The CDN466-X uses the 24-volt power from the DeviceNet network.

6.1.1 DeviceNet Connector Pin out

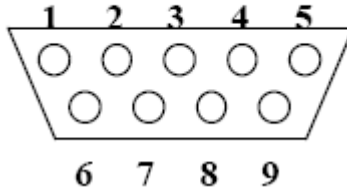
Male 5-Pin Micro Connector



PIN	SIGNAL	COLOR	DESCRIPTION
1	DRAIN	NONE	Cable shield or drain wire.
2	V+	RED	DeviceNet 24VDC(+) power.
3	V-	BLACK	DeviceNet 24VDC(-) power.
4	CAN_H	WHITE	Communication signal.
5	CAN_L	BLUE	Communication signal.

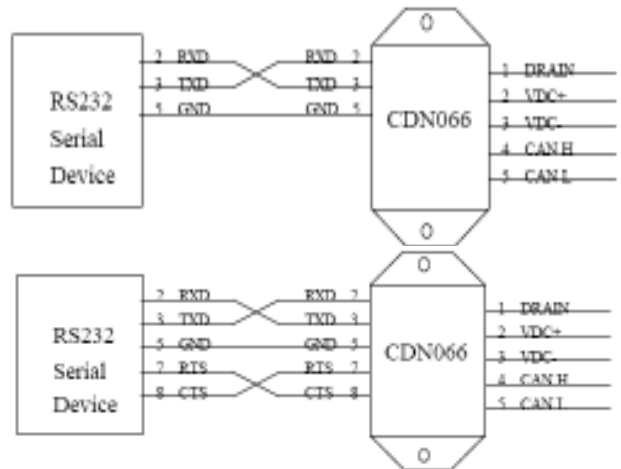
6.1.2 Serial Channel Pin outs and Connections

Male DB9 Serial Connector



CDN466-X (RS232)

PIN	SIGNAL	DESCRIPTION
1	NC	Pin not used
2	RXD	Receive Data (input)
3	TXD	Transmit Data (output)
4	NC	Pin not used
5	GND	Ground
6	NC	Pin not used
7	RTS	Request To Send (output)
8	CTS	Clear To Send (input)
9	NC	Pin not used



6.2 Required Hardware

- MKS Instruments CDN466-X Gateway.
- Allen-Bradley CompactLogix or equivalent CPU, Rack and Power Supply.
- Rockwell Software RSLogix 5000 programming software. Allen-Bradley CompactLogix.
- Rockwell Software RSNetworkx™ DeviceNet network configuration software.
- Serial device.
- RS232 null modem cable.
- DeviceNet Network.

6.3 Network Setup

1. Connect CDN466-X to the DeviceNet network. Use a null modem cable to connect CDN466-X to the serial device.
2. Set CDN466-X rotary switches:

RS232 BAUD RATE: Match baud rate for serial device connected to gateway.

DeviceNet MAC ID: Set MSD and LSD switches to an address (0-63) that is not currently used on the network.

DeviceNet BAUD RATE: Match DeviceNet network baud rate.

3. Use RSNetworkx™ to configure the baud rate and MAC ID for the System.
4. Power up PLC and DeviceNet power supplies.
5. Ensure the PLC is in *Program Mode* and clear any CPU faults.
6. Verify the scanners **MOD** LED is solid green, and **NET** LED is flashing green. If **NET** LED is flashing red, use RSNetworkx™ to remove all DeviceNet nodes from the scanners Scanlist.
7. Verify CDN466-X **MOD** LED is solid green, **NET** LED is flashing green. If **NET** LED is solid green or flashing red, use RSNetworkx™ to remove all DeviceNet nodes from the Scanlist, then cycle CDN466-X power.

6.4 Register EDS file

1. Once RSNetworkx™ is online and the CDN466-X appears with a question mark icon, double click on the CDN466-X icon to launch the EDS wizard



Note

Devices are unrecognized until the EDS file for the device is registered with RSNetworkx™.

5. Select the *Register an EDS file(s)* option and click *Next*.



Figure 6 Rockwell RSNetworkx™ EDS Wizard Screen

6. Select *Register a single file* option. *Browse* for the CDN466-X EDS file. Click *Next* when the file path is in the *Named:* field.



Figure 7 Select Correct EDS to Register the CDN466-X



Note

The latest EDS and icon files can be downloaded from www.mksinst.com

7. The next screen shows the RSNetworkx™ installation results. Click *Next* to continue.



Figure 8 Click Nex to Finish Registering CDN466-X EDS file

- To finish the registration of the EDS file, click *Next* and then *Finish* at the next two screens. Once the EDS wizard closes click the *Online* operation from the *Network* menu. View the DeviceNet network and ensure the CDN466-X is labeled CDN466-X and not Unregistered Device. Click *Cancel* when finished.

6.5 CDN466-X Gateway Configuration

NOTE: Screen capture is from 1747-SDN Scanner which is a SLC500 (16 bits based scanner). Mapping for other controller might be a little different. Also gateway image is from CDN466-X (which the CDN466-X is backward compatible to). The procedure remains exactly the same.

Once the EDS file is registered the CDN466-X can be configured using DeviceNet. Make certain that the CDN466-X is not enabled in the DeviceNet masters scanlist, this will prevent proper configuration of the CDN466-X. The following steps show how to configure the CDN466-X Serial Gateway using RSNetworx™.

- The following screen displays the nodes on the network. The CDN466-X will appear as one of the nodes on the network if all the other steps were followed correctly.

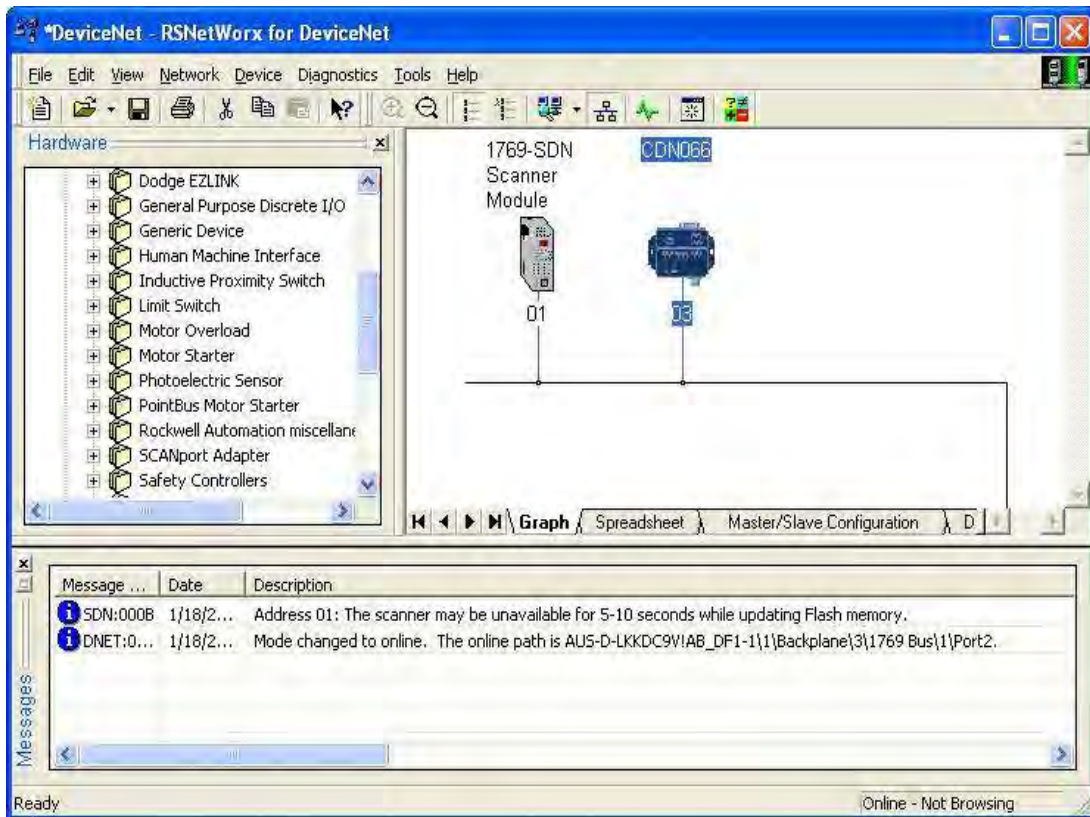


Figure 9 Scan Network for Available Node Connected to Network

3. Use upload to upload default configuration of device to EDS

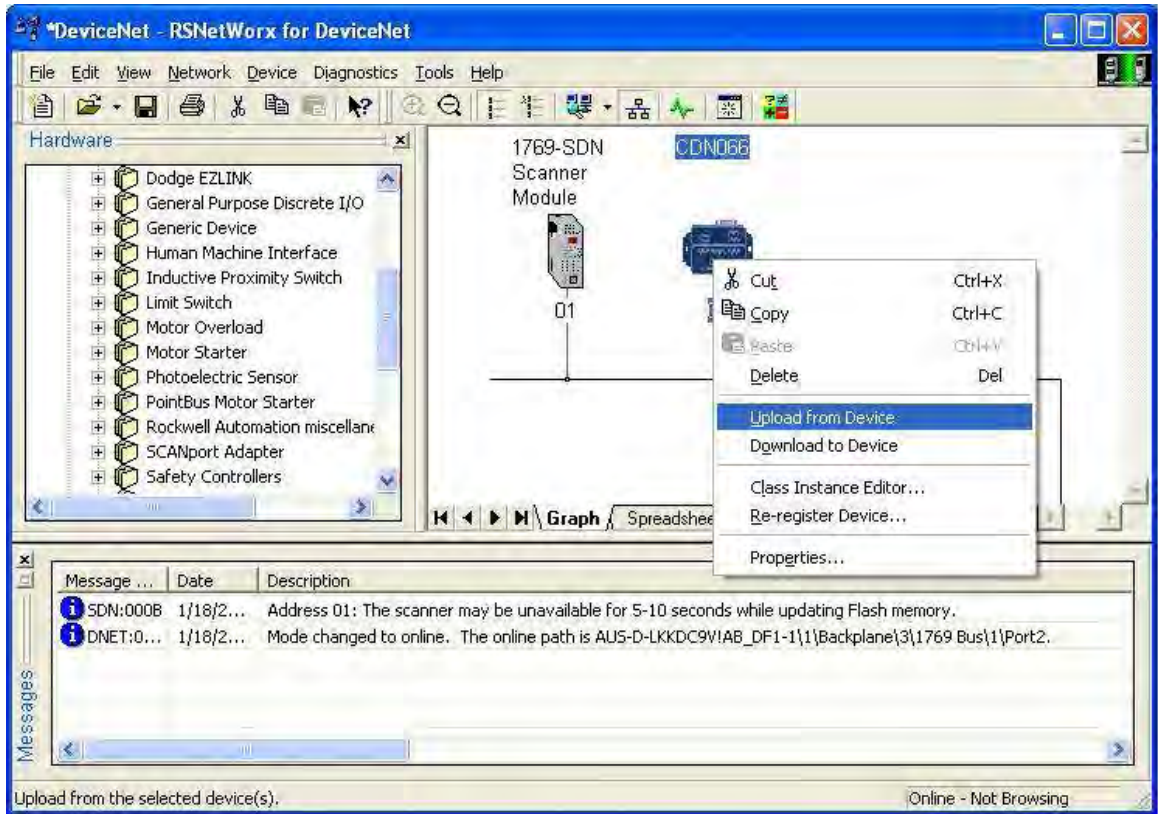


Figure 10 Upload Default Configuration from Node

3. Double-click on the CDN466-X icon to open the *Properties* menu for the device. Click on Parameters Tab to see default configuration data.

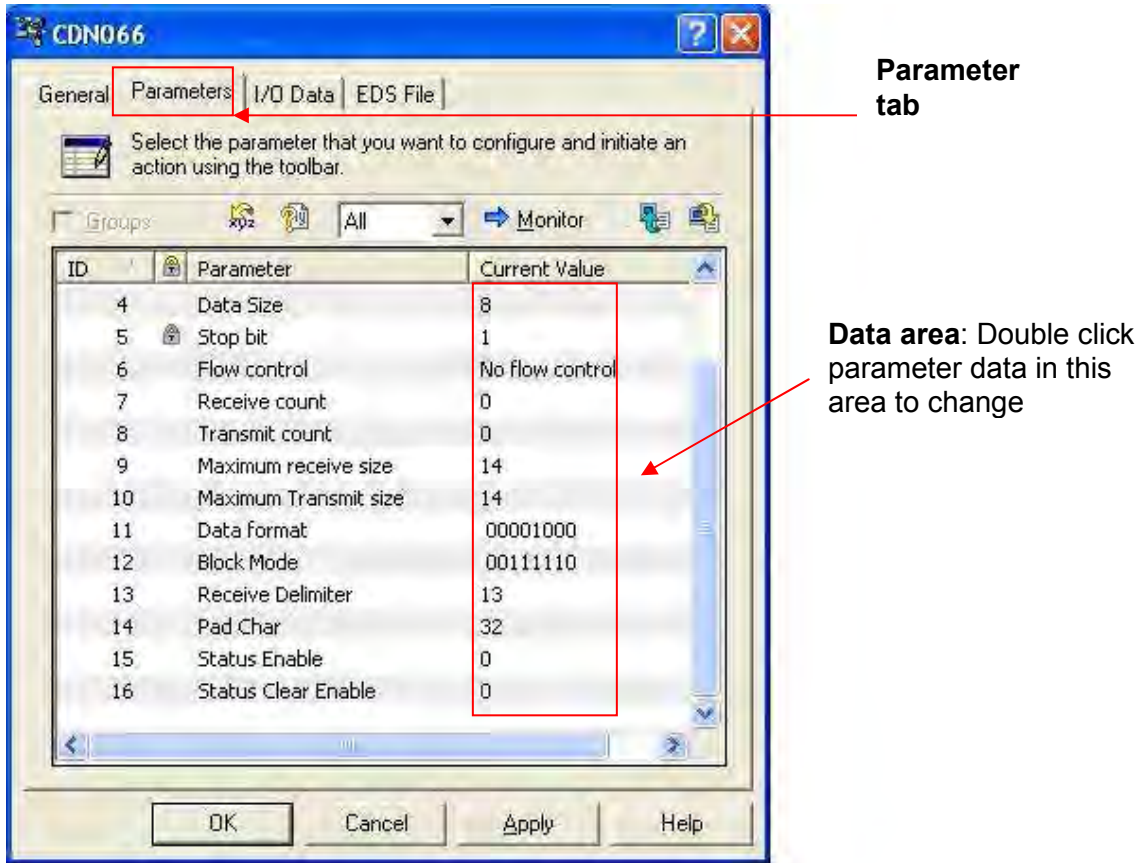


Figure 11 CDN466-X Attributes Data in Parameters Tab

4. Select the *Parameters* tab. You will be prompted for a parameters source. Select the *upload* button to upload factory settings from the CDN466-X. The CDN466-X parameters are now displayed in the *Properties* window.

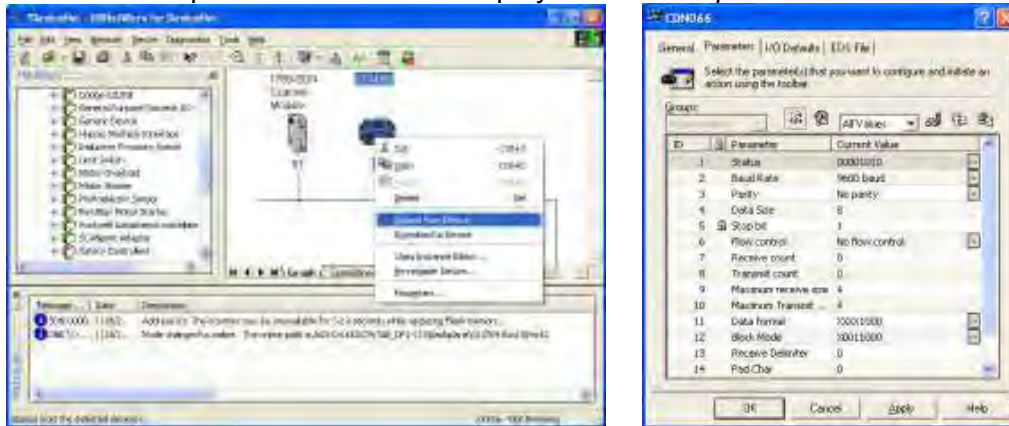


Figure 12 Upload Default Configuration from Node



Caution Selecting *Download* at this point will cause the current settings to be overwritten with the defaults in the EDS file.

6.5.1 Serial Parameters

This section walks through a simple configuration for the CDN466-X. This should serve as a starting point for incorporating the gateway into any application.

1. Click on the beside the *Status* parameter (ID 1). This will open a selection box that shows the status of monitored errors. Uncheck any check marks, this will clear any errors that might have occurred during installation of the CDN466-X.
2. The Baud Rate, Parity, and Flow Control parameters need to be adjusted according to the serial device that is connected to the gateway. Click the button to the right of the *Current value* column to change these parameters from the drop down menu.



Note If you are unsure how your serial device is configured, try using a program such as HyperTerminal to view or send data on the serial line.

3. Click on *Current Value* for the *Maximum Receive size* (ID 9) to set the maximum number of bytes to be received on the serial side. The *Maximum receive size* is determined by the number of bytes the serial device connected to the CDN466-X is expected to send in one message.
4. Click on *Current Value* for the *Maximum Transmit size* (ID 10) to set the maximum number of bytes to be transmitted on the serial side. The *Maximum*

Transmit size is determined by the number of bytes the serial device connected to the CDN466-X is expected to accept in one message.

- Click on the beside the *Data Format* parameter (ID 11). This will open a selection box that displays four data format options. Data Format is a control byte that defines the format of the TX and RX data bytes that are transferred across DeviceNet.

- Check *Pad* and uncheck the other three boxes.



String Format = 0 (unchecked) – enables Short_String format which adds 1 pre-byte to the data being transferred to indicated how many data bytes are in the current message.

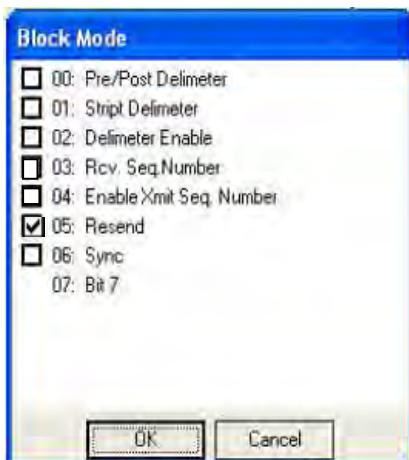
Strip Parity = 0 (unchecked) – Saves the Parity bit.

Pad Left/Right = 0 (unchecked) – Pad bytes are added to the beginning of the message (left justified).

Pad = 1 (checked) – Padding enabled. Adds pad bytes if there are not enough message bytes in the *Receive buffer* to fill the *RX message* input bytes. Enabling Pad is necessary for scanner use.

- Click on the beside the *Block Mode* parameter (ID 12). This will open a selection box that displays block mode configuration options. Block mode is a control byte that defines the serial receive mode, synchronization mode, and resend message option.

-Check Resend and uncheck all other boxes.



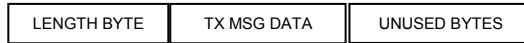
Resend = 1 (checked) – allows the last received data in the CDN466-X to remain as data in the poll response until new data is received. If not enabled, the data will only be sent once.

Delimiter Enable = 0 (unchecked) – Not in delimiter mode, ignore Pre/Post Delimiter and Strip Delimiter.

Rcv. Seq. Number, Enable Xmit Seq Number, Sync = 0 (unchecked) – No data handshaking implemented.

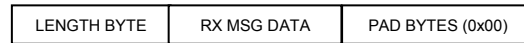
6.5.2 Message Format

Poll Command:



The length bytes equals the number of bytes in the data message. If the TX message size is less than the Max Transmit Size setting, the CDN466-X ignores the unused bytes.

Poll Response:



The LENGTH byte indicates the number of received message bytes for the current RX message. If the RX message size is less than the Max Receive Size setting, then Pad characters are added.



Note

If the serial object configuration differs from the instructions in the Serial Parameters section of this guide, the message structure could contain more or less bytes.

6.6 Configure DeviceNet Master Scanlist

After the serial object has been configured, the DeviceNet master needs to be configured to poll the CDN466-X.

1. Before using RSNetworkx™ to map the CDN466-X's Polled I/O connection to the DeviceNet scanner, the Poll Produce and Consume size must be calculated.

$ \begin{array}{r} \text{Short_String length byte} \\ + \text{Maximum Transmit Size} \\ \hline \text{Poll Consume Size} \end{array} $	$ \begin{array}{r} \text{Short_String length byte} \\ + \text{Maximum Receive Size} \\ \hline \text{Poll Produce Size} \end{array} $
---	--

2. Double click on the DeviceNet scanner icon to open the *Properties* box.

- 3) Select the Scanlist tab. RSNetworkx™ prompt for a Scanner Configuration. Click Upload to upoad current configuration settings from the scanner.
4. The next window shows the devices that are available to add to the DeviceNet scanlist.

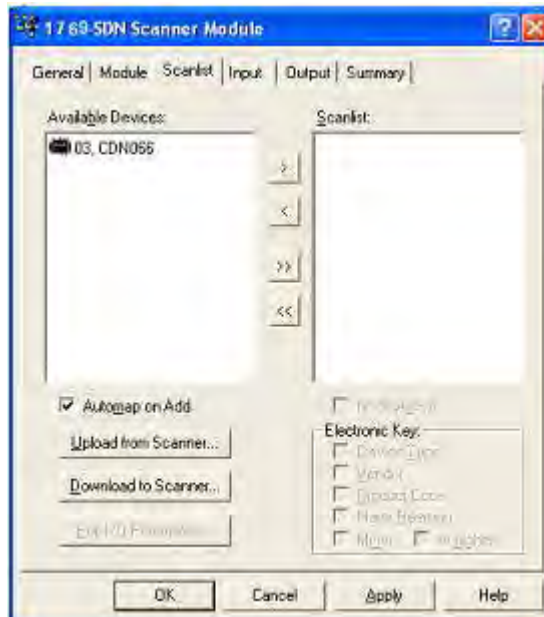



Figure 13 Check for Available Node to Add to Scanlist

5. Select the *Automap on Add* checkbox if you want RSNetworkx™ to automatically map the CDN466-X input and output bytes to the scanners memory.
6. Select the CDN466-X under *Available Devices* and click the  button to transfer the CDN466-X to the *Scanlist*.

NOTE Remove CDN466-X from the scan list by Right-click on scanner module (the 1769-SDN Scanner Module)->Properties menu. On the pop-up configuration window, go to Scanlist tab, highlight CDN066 Node, and uncheck Node Active box. This step is **necessary** before any configuration can be downloaded to a Devicenet device. User will get an error message from downloading if the node is still in the scan list.

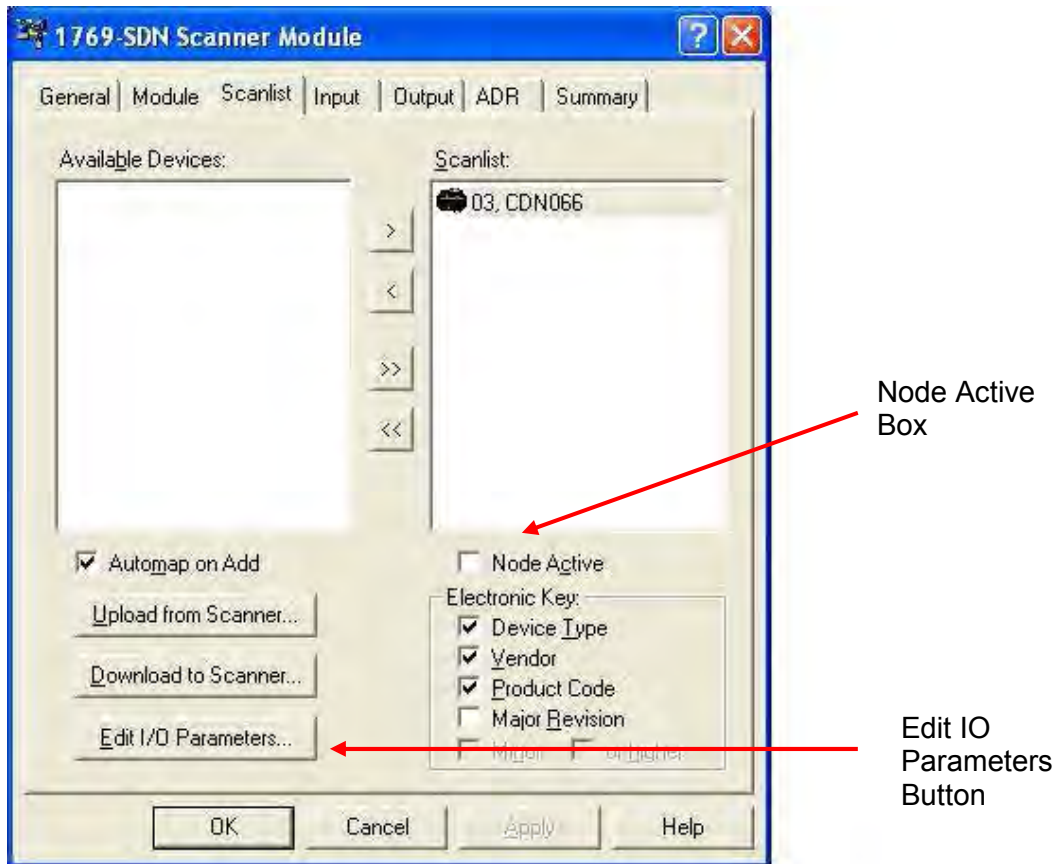


Figure 14 Transfer Unit to Scanlist

7. RSNetworkx™ warns that the CDN466-X does not contain any I/O data. Click **OK** to continue.
8. Once the CDN466-X is in the Scanlist click on the *Edit I/O Parameters* button. In the *Polled* section set the Rx Size: and TX Size to the calculated Poll Consume size and Poll Produce size respectively. Then click **OK** to update the I/O parameters.



Figure 15 Edit IO Produce and Consume Size

9. The software prompts to automap the I/O data bytes. Select Yes to automap. If you select No, you must manually map the I/O bytes in the memory tables. Then, the software prompts to download the changes to the scanner, select Yes.
10. Select the Input tab to view the automapped CDN466-X input bytes.

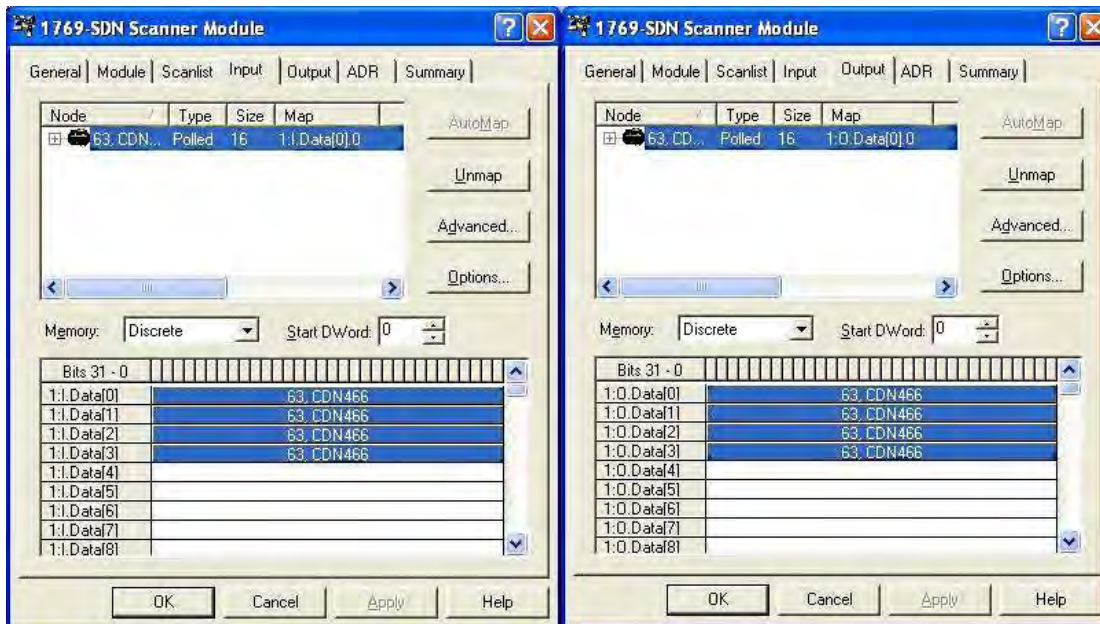


Figure 16 Verify IO Mapping on Scanner Memory Map

11. Click *Advanced...* to view current input mapping details. Change the mapping to suit your application. Click *Apply Mapping* after any changes have been made, then click *Yes* to download the changes. Click *Close* to continue.



Figure 17 IO Manual Mapping

12. Select the *Output* tab to view the automapped CDN466-X output bytes.

Note that the **Idle String** and **Fault String** attributes are not listed. These attributes use Short_String data type, which is not supported by RSNetworkx™ EDS File interface. Use the [Class Instance Editor](#) to configure Short_String attributes (Class Instance Editor is two options below the high-lighted shown in the image below)

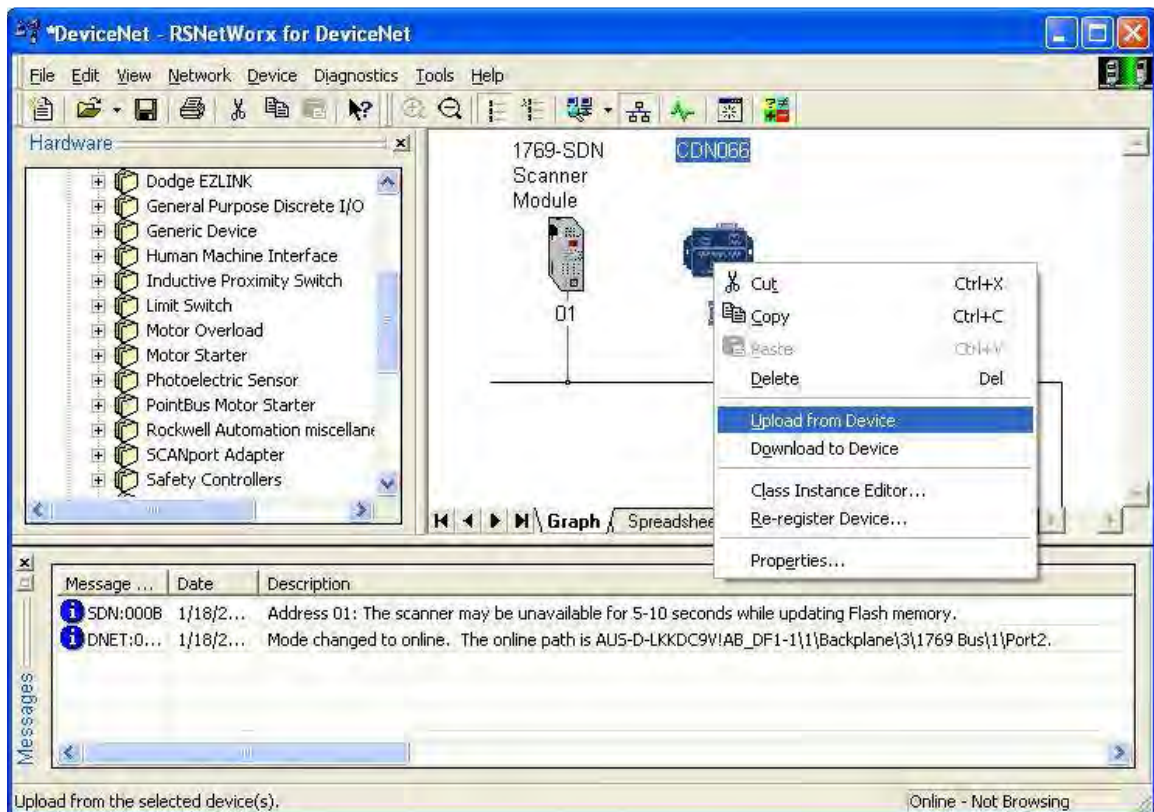


Figure 18 How to Access Class Instance Editor from RSNetworkx™

Select the Set_Attribute_Single service code to write an attribute value, and the Get_Attribute_Single service code to read an attribute value. Check *Values in decimal* box to enter class, instance, attribute, and data values in decimal. The Idle String address is Class 64, Instance 1, Attribute Number 19. The Fault String address is Class 64, Instance 1, Attribute Number 20. Enter the Short_String data as length byte, then data bytes. Example is [0x01 0x02] for a single byte string 0x02 (ASCII STX).

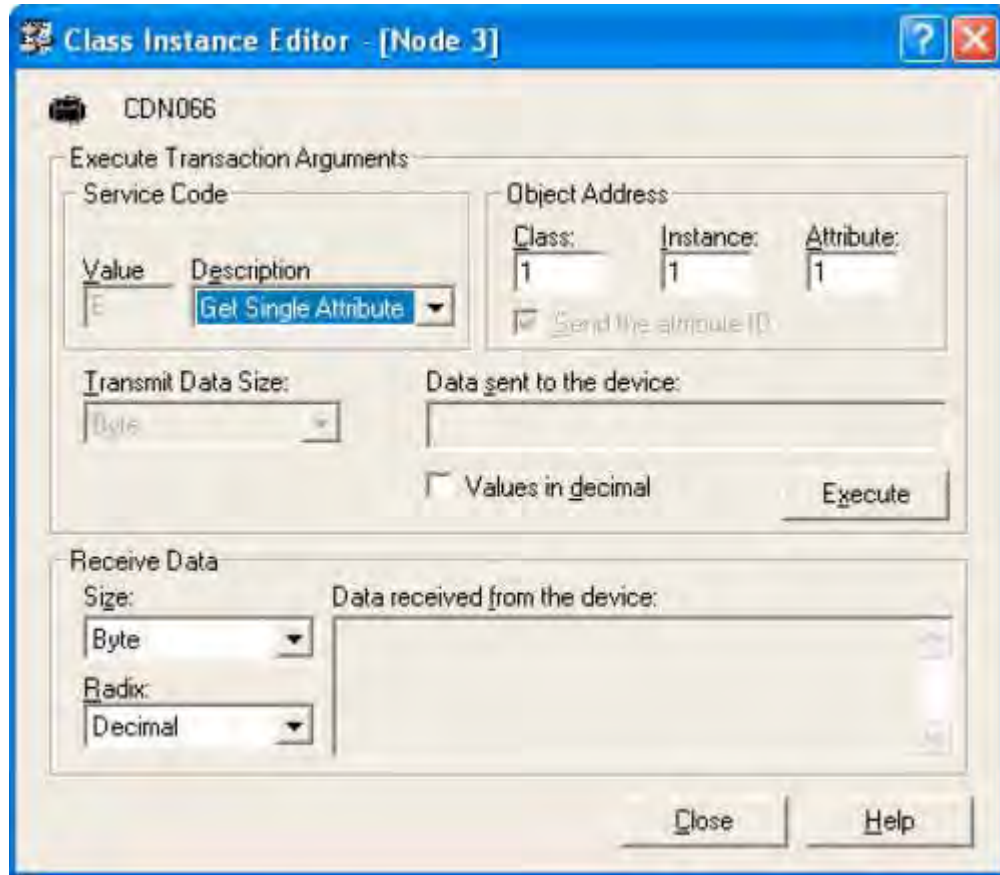


Figure 19 Use Class Instance Editor to change Fault or Idle String Attributes.

7 Configuration Example

This chapter contains five example gateway configurations.

Example 1 – Receiving Fixed-Length Data

Read UPC labels into a PLC using a serial barcode scanner, a CDN466-X gateway, and a DeviceNet scanner (master). The barcode scanner RS232 channel is connected to a CDN466-X serial channel. The CDN466-X DeviceNet channel is connected to the PLC DeviceNet scanner. The DeviceNet network is powered by an external 24VDC power supply.

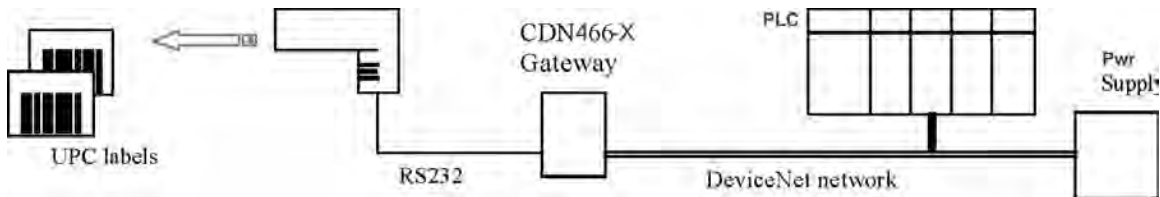


Figure 20 Network Set up with CDN466-X

Barcode Scanner

The barcode scanner’s RS232 channel is set for 9600 bps, 8 data bits, no parity, and 1 stop bit. When it reads a UPC label, it transmits a 5-byte serial message, which consists of the 5 ASCII characters printed on the UPC label.

CDN466-X Gateway

The receive mode will be *Stream Mode*, since there is no defined Delimiter for the start of a message or the end of a message. All received data bytes will be returned as DeviceNet input bytes. The *Maximum Receive Size* is 5, because the Barcode Scanner messages have a fixed length of 5 bytes. The data bytes will be returned as a Short_String. The gateway will only return the data bytes once in a Poll Response Message.

The *Serial Stream Object* can now be configured. The following shows the *Serial Stream Object* attribute settings for this application. The 3rd column lists the address string if using Set_Attribute_Single commands to write the attribute values.

Serial Stream Object Configuration (Class Code 64 or 0x40)

Attribute	Data	Class/Inst/Attr/Data	Description
6 = Baud Rate	0	0x40 0x01 0x06 0x00	0 = 9600 baud
7 = Parity	0	0x40 0x01 0x07 0x00	0 = No Parity
10 = Flow Control	2	0x40 0x01 0x0A 0x02	2 = RTS/CTS
13 = Max Rx Size	5	0x40 0x01 0x0D 0x05	Fixed message size of 5 bytes
14 = Data Format	b00000001	0x40 0x01 0x0E 0x01	String Format
15 = Block Mode	b00000000	0x40 0x01 0x0F 0x00	

The gateway will return 6 bytes of Receive Data because the Maximum Receive Size is set to 5 and the data format is Short String (add 1 for length byte). The Status and Receive Sequence Number bytes are not enabled. The Poll Produce Size can now be calculated for this CDN466-X configuration.

<i>Status byte</i>	0
<i>Receive Sequence Number byte</i>	0
<i>Short_String length byte</i>	1
<i>Maximum Receive Size</i>	5
<i>Produce Size</i>	6

The format of the Poll Response Message input bytes is as follows:

[Short_String length] [Short_String data]
1 byte 5 bytes

The gateway always returns 6 input bytes in the Poll Response Message, even if a new barcode message has not been received. The gateway will return new message data only once, and return a null data string if there is no new message data. The application should check the Short_String length byte to determine if a new message is being returned. A length of 5 indicates valid data bytes (new message data). A length of 0 indicates no valid data bytes (no new message).

The Barcode Scanner sends the following 5-byte serial message when it reads a UPC label printed with '12345' (ASCII numbers).

0x31 0x32 0x33 0x34 0x35

The gateway generates the following Poll Response Message in response to the first Poll Command Message after its receives the Barcode message. The Short_String length is 5, since 5 bytes were received.

0x05	0x31 0x32 0x33 0x34 0x35
------	--------------------------

The gateway generates the following Poll Response Message in response to the first Poll Command Message after it receives the Barcode message. The Short_String length is 5, since 5 bytes were received.

0x00	0x00 0x00 0x00 0x00 0x00
------	--------------------------

Example 2 – Receiving Pre-Delimited Data

Same configuration as Example 1.

Barcode Scanner

The barcode scanner’s RS232 channel is set for 9600 bps, 8 data bits, no parity, and 1 stop bit. When it reads a UPC label, it transmits following ASCII message format. The message always begins with the ASCII STX start-of-text (0x02) character. The barcode data will consist of a variable number of 1 to 14 ASCII characters, depending upon the UPC label being scanned. It will not transmit a 0x02 in the barcode data field.

[STX] [ASCII barcode data]

CDN466-X Gateway

The receive mode will be *Pre-Delimiter Mode*, because the barcode messages always begin with the same character. The *Delimiter* is 0x02 (STX). The *Maximum Receive Size* is 15, because the largest message contains 1 STX byte and 14 ASCII bytes. The received bytes will be returned as a *Short_String*. An ASCII NUL Pad character (0x00) will be added at the end of the message if needed. The gateway will always return the data bytes in the *Poll Response Message*. The *Receive Sequence Number* will be used to indicate when a new message is returned.

The *Serial Stream Object* can now be configured. The following shows the *Serial Stream Object* attribute settings for this application. The 3rd column lists the address string if using *Set_Attribute_Single* commands to write the attribute values.

Serial Stream Object Configuration (Class Code 64 or 0x40)

Attribute	Data	Class/Inst/Attr/Data	Description
6 = Baud Rate	0	0x40 0x01 0x06 0x00	0 = 9600 baud
7 = Parity	0	0x40 0x01 0x07 0x00	0 = No Parity
10 = Flow Control	2	0x40 0x01 0x0A 0x02	2 = RTS/CTS
13 = Max Rx Size	15	0x40 0x01 0x0D 0x0F	Fixed msg size of 15 bytes
14 = Data Format	b00001101	0x40 0x01 0x0E 0x0D	Pad receive message Pad justificaiton = right (end of msg) String Format = Short_String
15 = Block Mode	b00101101	0x40 0x01 0x0F 0x2D	Resend = enabled RX Sequence Number = enabled Delimiter = enabled Pre-Delimiter
16 = Delimiter	STX	0x40 0x01 0x10 0x02	0x02 = ASCII STX Character
17 = Pad Character	NULL	0x40 0x01 0x11 0x00	0x00 = ASCII NULL Character

The gateway will return up to 16 bytes of Receive Data, because the Maximum Receive Size is set to 15 and the data format is Short String (add 1 for length byte). The Status byte is not enabled. The Receive Sequence Number byte is enabled. The Poll Produce Size can now be calculated for this CDN466-X configuration.

<i>Status byte</i>	0
<i>Receive Sequence Number byte</i>	1
<i>Short_String length byte</i>	1
<i>Maximun Receive Size</i>	15
<i>Produce Size</i>	17

The format of the Poll Response Message input bytes is as follows:

[Receive Sequence Number] [Short_String length] [Short_String data] [Pad bytes]
 1 byte 1 byte 0-15 bytes

The gateway always returns 17 input bytes in the Poll Response Message, even if the scanned barcode data contains fewer bytes. The application should check the Short_String length byte to determine the number of valid data bytes being returned in a particular Poll Response Message. The remaining input bytes have undefined values.

The gateway will always return the last received Short_String data in its Poll Response Message. The gateway increments the Receive Sequence Number when new Short_String data is returned. The application can use the Receive Sequence Number to determine if the Short_String data is new or old information.

The Barcode Scanner sends the following 8-byte serial message when it reads a UPC label printed with ‘1234567’ (ASCII numbers).

0x02 0x31 0x32 0x33 0x34 0x35 0x36 0x37

The gateway generates the following Poll Response Message. The Receive Sequence Number is 1, since this is the first message received from the Barcode Scanner. The Short_String length is 8, since 8 bytes were received. 7 Pad characters are added at the end of the message.

0x01	0x08	0x02 0x31 0x32 0x33 0x34 0x35 0x36 0x37	0x00 0x00 0x00 0x00 0x00 0x00 0x00
------	------	---	------------------------------------

Example 3 – Receiving Post-Delimited Data

Same configuration as Example 1.

Barcode Scanner

The barcode scanner’s RS232 channel is set for 9600 bps, 8 data bits, no parity, and 1 stop bit. When it reads a UPC label, it transmits following ASCII message format. The message always begins ends with the ASCII ETX end-of-text (0x03) character. The barcode data will consist of a variable number of 1 to 14 ASCII characters, depending upon the UPC label being scanned. It will not transmit a 0x03 in the barcode data field.

[ASCII barcode data] [ETX]

CDN466-X Gateway

The receive mode will be *Post-Delimiter Mode*, because the barcode messages always end with the same character. The *Delimiter* is 0x03 (ETX), and will not be included in the receive data. The *Maximum Receive Size* is 15, because the largest message contains 14 ASCII bytes and 1 ETX byte. The received bytes will be returned a Short String. The gateway will only return new data bytes once in the Poll Response Message. The *Status* byte will be enabled.

The *Serial Stream Object* can now be configured. The following shows the *Serial Stream Object* attribute settings for this application. The 3rd column lists the address string if using *Set_Attribute_Single* commands to write the attribute values.

Serial Stream Object Configuration (Class Code 64 or 0x40)

Attribute	Data	Class/Inst/Attr/Data	Description
6 = Baud Rate	0	0x40 0x01 0x06 0x00	0 = 9600 baud
7 = Parity	0	0x40 0x01 0x07 0x00	0 = No Parity
10 = Flow Control	2	0x40 0x01 0x0A 0x02	2 = RTS/CTS
13 = Max Rx Size	15	0x40 0x01 0x0D 0x0F	Fixed message size of 15 bytes
14 = Data Format	b00000001	0x40 0x01 0x0E 0x01	String Format = Byte Array
15 = Block Mode	b00000110	0x40 0x01 0x0F 0x06	Strip Delimiter = enabled Delimiter = enabled Post-Delimiter
16 = Delimiter	ETX	0x40 0x01 0x10 0x03	0x03 = ASCII ETX Character

The gateway will return up to 16 bytes of Receive Data, because the Maximum Receive Size is set to 15 and the data format is Short String (add 1 for length byte). The Status byte is enabled. The Receive Sequence Number byte is not enabled. The Poll Produce Size can now be calculated for this CDN466-X configuration.

<i>Status byte</i>	1
<i>Receive Sequence Number byte</i>	0
<i>Short_String length byte</i>	1
<i>Maximun Receive Size</i>	15
<i>Produce Size</i>	17

The format of the Poll Response Message input bytes is as follows:

[Status] [Short String length] [Short String data] [Undefined bytes]
 1 byte 1 byte 0-15 bytes

The gateway always returns 16 input bytes in the Poll Response Message, even if the scanned barcode data contains fewer bytes, or if a new barcode message has not been received. The gateway returns new message data only once, and returns a null data string if there is no new message data. The application should use the Short_String length byte to determine if a new message is being returned. A length greater than zero indicates the number of valid data bytes (new message data). A length of 0 indicates no valid data bytes (no new message).

The Barcode Scanner sends the following 6-byte serial message when it reads a UPC label printed with '12345' (ASCII numbers).

0x31 0x32 0x33 0x34 0x35 0x03

The gateway generates the following Poll Response Message in response to the first Poll Command Message after its receives the Barcode message. The Status Byte is 0x0A, indicating no transmit or receive errors, an empty Transmit Buffer, and an empty Receive Buffer. The Delimiter is stripped, so the Short_String length is 5. There are 5 valid data bytes, and the remaining 10 input bytes are undefined.

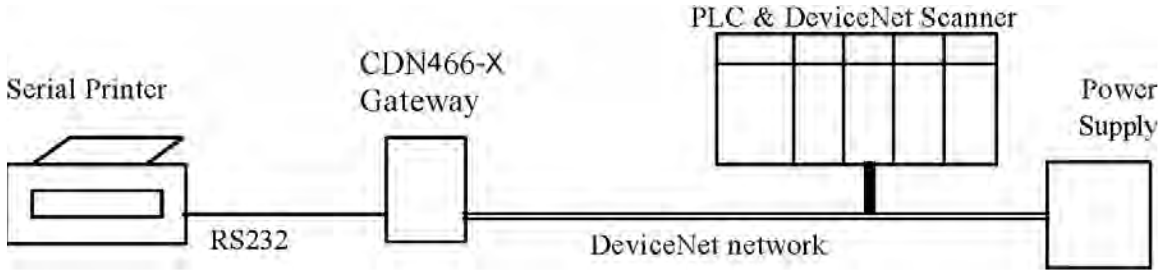
0x0A	0x05	0x31 0x32 0x33 0x34 0x35	XX XX XX XX XX XX XX XX XX XX
------	------	--------------------------	-------------------------------

The gateway generates the following Poll Response Message in response to subsequent Poll Command Messages, until it receives another Barcode message. The Short_String length is 0, indicating a null data string. The 15 other input bytes are undefined.

0x0A	0x00	XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
------	------	--

Example 4 – Transmitting Fixed-Length Data

Print an ASCII string from a PLC to a serial printer, using a CDN466-X gateway and a DeviceNet scanner (master). The text message string is always 25 characters long, including any ASCII control characters. The serial printer RS232 channel is connected to a CDN466-X serial channel. The CDN466-X DeviceNet channel is connected to the PLC DeviceNet scanner. The DeviceNet network is powered by an external 24VDC power supply.



Serial Printer

The serial printer’s RS232 channel is set for 300 bps, 7 data bits, even parity, and 1 stop bit. It uses XON / XOFF software flow control.

CDN466-X Gateway

The CDN466-X serial channel is configured to transmit this RS232 message format. A string format will be Byte Array, since the message size is fixed. Transmit Sequence Numbers will be used to signal a new message to transmit. The Maximum Transmit Size is 25, which is the number of message bytes. The *Serial Stream* Object attributes are shown below for this application. The 3rd column lists the address string if using Set_Attribute_Single commands to write the attribute values.

Serial Stream Object Configuration (Class Code 64 or 0x40)

Attribute	Data	Class/Inst/Attr/Data	Description
6 = Baud Rate	5	0x40 0x01 0x06 0x05	5 = 300 baud
7 = Parity	1	0x40 0x01 0x07 0x01	1 = Even Parity
10 = Flow Control	1	0x40 0x01 0x0A 0x01	1 = XON/XOFF
14 = Data Format	b00000000	0x40 0x01 0x0E 0x00	String Format = Byte Array
15 = Block Mode	b00010000	0x40 0x01 0x0F 0x10	TX Sequence Number = enabled
18 = Max TX Size	25	0x40 0x01 0x12 0x19	Fixed message size of 25 bytes

The gateway will transmit 25 output bytes received in a Poll Command Message. The Status Clear byte is not enabled. The Transmit Sequence Number is enabled. The Length Byte is not enabled (Byte Array format). The Poll Consume Size can now be calculated for this CDN466-X configuration.

<i>Status byte</i>	0
<i>Receive Sequence Number byte</i>	1
<i>Short_String length byte</i>	0
<i>Maximun Receive Size</i>	25
<i>Produce Size</i>	26

The format of the Poll Command Message input bytes is as follows:

[TX Sequence Number]	[Data Bytes]
1 byte	0-25 bytes

The gateway always receives 26 output bytes in the Poll Command Message. It will not transmit a new serial message until the Transmit Sequence Number received in the Poll Command is different than the number received in a previous Poll Command. The application should increment the Transmit Sequence Number when it sends new output byte values in the Poll Command Message, to enable the transmission of the new message.

Example 5 – Transmitting Variable-Length

Same configuration as Example 4, except the text message string can be from 1 to 25 characters long, including ASCII control characters.

Serial Printer

The serial printer’s RS232 channel is set for 300 bps, 7 data bits, even parity, and 1 stop bit. It uses XON / XOFF software flow control.

CDN466-X Gateway

The CDN466-X serial channel is configured to transmit this RS232 message format. A string format will be Short_String, since the message size is variable. The Maximum Transmit Size is 25, since the largest text message contains 25 characters. The *Serial Stream Object* attributes are shown below for this application. The 3rd column lists the address string if using Set_Attribute_Single commands to write the attribute values.

Serial Stream Object Configuration (Class Code 64 or 0x40)

Attribute	Data	Class/Inst/Attr/Data	Description
6 = Baud Rate	5	0x40 0x01 0x06 0x05	5 = 300 baud
7 = Parity	1	0x40 0x01 0x07 0x01	1 = Even Parity
10 = Flow Control	1	0x40 0x01 0x0A 0x01	1 = XON/XOFF
14 = Data Format	b00000001	0x40 0x01 0x0E 0x01	String Format = Short String
15 = Block Mode	b00000000	0x40 0x01 0x0F 0x00	TX Sequence Number = Disabled
18 = Max TX Size	25	0x40 0x01 0x12 0x19	Fixed message size of 25 bytes

The gateway will transmit the output bytes received in a Poll Command Message. The Status Clear byte is not enabled. The Transmit Sequence Number is not enabled. The Length Byte is enabled (Short_String format). The Poll Consume Size can now be calculated for this CDN466-X configuration.

<i>Status byte</i>	0
<i>Receive Sequence Number byte</i>	0
<i>Short_String length byte</i>	1
<i>Maximun Receive Size</i>	25

<i>Produce Size</i>	26

The format of the Poll Command Message input bytes is as follows:

[Short String Length] [Data Bytes]
1 byte 0-25 bytes

The gateway always receives 26 output bytes in the Poll Command Message, regardless of the variable length messages. The gateway uses the Short_String length byte to determine the valid number of message bytes in the Poll Command Message. It will only transmit the valid message bytes. All remaining output bytes are ignored. If the gateway receives a Poll Command Message with Short_String length = 0, no output bytes are transmitted. The application can send variable-length Short_Strings to be transmitted, and send Null Data (length = 0) when there is no message to transmit.

Troubleshooting

Problem	Possible Cause
DeviceNet Configuration Program does not recognize Gateway.	<ul style="list-style-type: none"> • Register Gateway EDS file with Configuration Program.
DeviceNet Configuration Program does not recognize Gateway after loading EDS file.	<ul style="list-style-type: none"> • Check Major and Minor Revisions for Gateway and EDS file, to see if you have correct EDS file for your Gateway's firmware version.
Gateway does not appear on DeviceNet network.	<ul style="list-style-type: none"> • Check wiring and cable connections. • Check DeviceNet power supply voltage. <input type="checkbox"/> Make sure Gateway baud rate matches network baud rate. <input type="checkbox"/> Verify Gateway baud rate is set from rotary switches or retentive memory value. <input type="checkbox"/> Make sure Gateway MAC ID is not used by another device.
After setting Gateway MAC ID, DeviceNet Master does not recognize Gateway.	<ul style="list-style-type: none"> • Disconnect Gateway from network before changing MAC ID. • Make sure Gateway MAC ID is not used by another device. • Verify Gateway MAC ID is set from rotary switches or retentive memory value. • Verify DeviceNet baud rate.
<i>NET</i> LED is flashing red.	<ul style="list-style-type: none"> • Gateway is removed from DeviceNet Master scanlist or network. Power cycle Gateway to reset.
<i>NET</i> LED is solid red.	<ul style="list-style-type: none"> • Make sure Gateway MAC ID is not used by another device. Possible DeviceNet network failure.
<i>NET</i> LED is off.	<ul style="list-style-type: none"> • Check wiring and cable connections. • Check DeviceNet power supply voltage. <input type="checkbox"/> Make sure Gateway baud rate matches network baud rate. • Verify Gateway baud rate is set from rotary switches or retentive memory value.
<i>MOD</i> LED is flashing or solid red.	<ul style="list-style-type: none"> • Gateway has failed. Cycle power to reset. Replace Gateway if necessary.
<i>RX</i> LED does not flash green when data is sent to the Gateway.	<ul style="list-style-type: none"> • If Sync enabled, make sure Receive Request Number and Receive Acknowledge Number are equal. Application must acknowledge last received message before gateway will receive the next message. • Verify data is being received in Receive Data. • Verify source device is transmitting data to Gateway. • Make sure hardware flow control signals are properly connected.
<i>RX</i> LED is solid red after Gateway receives data.	<ul style="list-style-type: none"> • Check Status byte for any Receiver errors. Reset Gateway or clear Status error bits if necessary. <input type="checkbox"/> Make sure parity is set to match transmitting device settings.

<p><i>TX</i> LED is solid red after receiving data from DeviceNet Master.</p>	<ul style="list-style-type: none"> • Check Status byte for Transmitter errors. Reset Gateway or clear Status error bits if necessary. • Make sure parity is set to match receiving device settings.
<p><i>TX</i> LED does not flash green when Gateway should be transmitting data</p>	<ul style="list-style-type: none"> • If Transmit Sequence Number enabled, make sure number is being incremented by the application. Gateway will not transmit new data unless the Transmit Sequence Number is changed. • Verify data is being saved in Transmit Data.
<p>DNET Scanner displays error code 77.</p>	<ul style="list-style-type: none"> • Gateway Poll Produce Size and/or Poll Consume Size value do not match scanner Poll Rx/Tx settings.

Appendix A – Product Specification

DeviceNet Interface

Power Requirements:	11 - 28 Vdc @ 50 mA
Loss of Ground:	Yes
Reverse Polarity:	-30 Vdc
Signal Levels:	ISO11898

Serial Channel

Isolation:	500 Volts
ESD Protection:	+/- 10 kV
Overload Protection:	+/- 30 Volts
Short Circuit:	Indefinite
RS232 Output Levels:	+/- 7.9 Volts (unloaded, typical)

Environmental

Operating Temperature:	0° C to 70° C
Storage Temperature:	-25° C to 85° C
Size (inches):	3.25 x 2.37 x 1.08
Mounting (inches)	0.5 tabs, 3/16 diameter mounting holes

Appendix B: ASCII Character Codes

Table 11. Printable Characters

Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x20	32	Space	0x40	64	@	0x60	96	`
0x21	33	!	0x41	65	A	0x61	97	a
0x22	34	"	0x42	66	B	0x62	98	b
0x23	35	#	0x43	67	C	0x63	99	c
0x24	36	\$	0x44	68	D	0x64	100	d
0x25	37	%	0x45	69	E	0x65	101	e
0x26	38	&	0x46	70	F	0x66	102	f
0x27	39	'	0x47	71	G	0x67	103	g
0x28	40	(0x48	72	H	0x68	104	h
0x29	41)	0x49	73	I	0x69	105	i
0x2A	42	*	0x4A	74	J	0x6A	106	j
0x2B	43	+	0x4B	75	K	0x6B	107	k
0x2C	44	,	0x4C	76	L	0x6C	108	l
0x2D	45	-	0x4D	77	M	0x6D	109	m
0x2E	46	.	0x4E	78	N	0x6E	110	n
0x2F	47	/	0x4F	79	O	0x6F	111	o
0x30	48	0	0x50	80	P	0x70	112	p
0x31	49	1	0x51	81	Q	0x71	113	q
0x32	50	2	0x52	82	R	0x72	114	r
0x33	51	3	0x53	83	S	0x73	115	s
0x34	52	4	0x54	84	T	0x74	116	t
0x35	53	5	0x55	85	U	0x75	117	u
0x36	53	6	0x56	86	V	0x76	118	v
0x37	55	7	0x57	87	W	0x77	119	w
0x38	56	8	0x58	88	X	0x78	120	x
0x39	57	9	0x59	89	Y	0x79	121	y
0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x3B	59	;	0x5B	91	[0x7B	123	{
0x3C	60	<	0x5C	92	\	0x7C	124	
0x3D	61	=	0x5D	93]	0x7D	125	}
0x3E	62	>	0x5E	94	^	0x7E	126	~
0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Table 12. Non-Printable Characters

Hex	Dec	Char	Name	Kybd
0x00	0	NUL	Null	Ctrl @
0x01	1	SOH	Start of heading	Ctrl A
0x02	2	STX	Start of text	Ctrl B
0x03	3	ETX	End of text	Ctrl C
0x04	4	EOT	End of transmit	Ctrl D
0x05	5	ENQ	Enquiry	Ctrl E
0x06	6	ACK	Acknowledge	Ctrl F
0x07	7	BEL	Bell	Ctrl G
0x08	8	BS	Backspace	Ctrl H
0x09	9	HT	Horizontal tab	Ctrl I
0x0A	10	LF	Line feed	Ctrl J
0x0B	11	VT	Vertical tab	Ctrl K
0x0C	12	FF	Form feed	Ctrl L
0x0D	13	CR	Carriage return	Ctrl M
0x0E	14	SO	Shift out	Ctrl N
0x0F	15	SI	Shift in	Ctrl O
0x10	16	DLE	Data line escape	Ctrl P
0x11	17	DC1	Device control 1	Ctrl Q
0x12	18	DC2	Device control 2	Ctrl R
0x13	19	DC3	Device control 3	Ctrl S
0x14	20	DC4	Device control 4	Ctrl T
0x15	21	NAK	Negative acknowledge	Ctrl U
0x16	22	SYN	Synchronous idle	Ctrl V
0x17	23	ETB	End of transmit block	Ctrl W
0x18	24	CAN	Cancel	Ctrl X
0x19	25	EM	End of medium	Ctrl Y
0x1A	26	SUB	Substitute	Ctrl Z
0x1B	27	ESC	Escape	Ctrl [
0x1C	28	FS	File separator	Ctrl \
0x1D	29	GS	Group separator	Ctrl]
0x1E	30	RS	Record separator	Ctrl ^
0x1F	31	US	Unit separator	Ctrl _

WARRANTY

MKS Instruments, Inc. (**MKS**) warrants that for one year from the date of shipment the equipment described above (the “equipment”) manufactured by **MKS** shall be free from defects in materials and workmanship and will correctly perform all date-related operations, including without limitation accepting data entry, sequencing, sorting, comparing, and reporting, regardless of the date the operation is performed or the date involved in the operation, provided that, if the equipment exchanges data or is otherwise used with equipment, software, or other products of others, such products of others themselves correctly perform all date-related operations and store and transmit dates and date-related data in a format compatible with **MKS** equipment. THIS WARRANTY IS **MKS’** SOLE WARRANTY CONCERNING DATE-RELATED OPERATIONS.

For the period commencing with the date of shipment of this equipment and ending one year later, **MKS** will, at its option, either repair or replace any part which is defective in materials or workmanship or with respect to the date-related operations warranty without charge to the purchaser. The foregoing shall constitute the exclusive and sole remedy of the purchaser for any breach by **MKS** of this warranty.

The purchaser, before returning any equipment covered by this warranty, which is asserted to be defective by the purchaser, shall make specific written arrangements with respect to the responsibility for shipping the equipment and handling any other incidental charges with the **MKS** sales representative or distributor from which the equipment was purchased or, in the case of a direct purchase from **MKS**, with the **MKS-CIT** home office in San Jose, CA

This warranty does not apply to any equipment, which has not been installed and used in accordance with the specifications recommended by **MKS** for the proper and normal use of the equipment. **MKS** shall not be liable under any circumstances for indirect, special, consequential, or incidental damages in connection with, or arising out of, the sale, performance, or use of the equipment covered by this warranty.

THIS WARRANTY IS IN LIEU OF ALL OTHER RELEVANT WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING THE IMPLIED WARRANTY OF MERCHANTABILITY AND THE IMPLIED WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY AGAINST INFRINGEMENT OF ANY PATENT.