BALLUFF

sensors worldwide

BIS Z-GW-001_ Gateway

Technical Description, Manual



CONTENTS

REFERENCES 1
Conventions 1
Reference Documentation 1
Services and Support 1

REGULATORY AND COMPLIANCE NOTICES 2
Power Supply 2

GENERAL VIEW 3

1 OVERVIEW 7
1.1 Introduction 7
1.2 Subnet16™ Gateway Features 7
1.3 About this Manual 7
1.3.1 Who Should Read This Manual? 8
1.3.2 HEX Notation 8
1.4 Models and Accessories 8

2 INSTALLATION 11
2.1 Mechanical Dimensions 11
2.1.1 RS232 Models 11
2.1.2 IND / TCP Models 12
2.1.3 DNT Models 13
2.1.4 PBS Models 14
2.2 Electrical Connectors 15
2.2.1 Subnet16™ 15
2.2.2 RS232 16
2.2.3 Industrial Ethernet IP 17
2.2.4 DeviceNet 18
2.2.5 Profibus 19
2.3 Power & Wiring 20
2.3.1 Power Requirements 20
2.3.2 Total System Current Consumption 20
2.3.3 Cable Voltage Drop 21
2.3.4 Maximum Supported Trunk and Drop Cable Lengths 21
2.3.5 Current Rating for Cables 21
2.4 Installation Guidelines 22
2.4.1 Hardware Requirements 22
2.4.2 Installation Precautions 22
2.5 Typical Layouts and Installation Procedures 23
2.5.1 Installing the Subnet16™ Gateway 23
2.5.2 Installing the Industrial Ethernet (IND) Subnet16™ Gateway 25
2.5.3 Installing the DeviceNet (DNT) Subnet16™ Gateway 27
2.5.4 Installing the Profibus (PBS) Subnet16™ Gateway 29

3 LED INDICATORS 31
3.1 Front Panel LEDs 31
3.1.1 RS232 Models 31
3.1.2 INDUSTRIAL and TCP/IP Ethernet Models 32
3.1.3 DEVICENET Models 33
3.1.4 PROFIBUS Models 34

4 CONFIGURATION METHODS 35
4.1 Node ID Configuration Using Configuration Tags 35
4.2 Gateway and Subnet Node Naming 36
4.3 Configuration Tools 36
4.3.1 Configuration Using Balluff Dashboard™ 37
4.3.2 Creating and Using RFID Macros with C-Macro Builder™ 39
4.4 USB Driver Installation 42

5 ETHERNET/IP INTERFACE 47
5.1 Ethernet/IP Configuration Overview 48
5.2 HTTP Server & OnDemand PLC Support 48
5.3 HTTP Server and OnDemand Utilities 49
5.4 IP Configuration via HTTP Server 50
5.5 OnDemand Configuration for Ethernet/IP 52
5.6 Configuring PLC Controller Tags 55
5.7 Checking OnDemand Status 56
5.8 Verifying Data Exchange with RSLogix 5000 57
5.8.1 Ethernet/IP Handshaking 57
5.8.2 Ethernet/IP Handshaking Example 58
5.9 Ethernet/IP: Object Model 59
5.9.1 Ethernet/IP Required Objects 60
5.9.2 EtherNet/IP: Vendor Specific Objects 64
5.9.3 Application Object (0x67 _ 10 Instances) 67

6 MODBUS TCP INTERFACE 69
6.1 Modbus TCP Overview 69
6.2 Modbus TCP Configuration via HTTP Server 69
6.2.1 Modbus TCP - Command Packet Structure 72
6.2.2 Modbus TCP - Response Packet Structure 72
6.2.3 Modbus TCP - Mapping for Node 65 73
6.3 Modbus TCP - Handshaking 74
6.3.1 Modbus TCP - Host/processor unit Handshaking 75
6.3.2 Modbus TCP - Handshaking Example 76

7 STANDARD TCP/IP INTERFACE 77
7.1 Standard TCP/IP Overview 77
7.2 IP Configuration via HTTP Server 77
7.3 IP Configuration via Digi Discovery 80
7.4 Standard TCP/IP - Command & Response Examples 81
7.4.1 Standard TCP/IP - Command Structure Example 82
7.4.2 Standard TCP/IP - Response Structure Example 83

8 DEVICENET INTERFACE 85
8.1 DeviceNet Overview 85
8.2 DeviceNet Configuration 85
8.2.1 Importing the Controller.EDS File 85
8.2.2 Configuring Gateway and PLC DeviceNet Communications 86
8.2.3 Configuring Data Rate and Node Address 91
8.2.4 DeviceNet - Exchanging Data and Handshaking 92
8.2.5 DeviceNet - Handshaking Example 93

9 PROFIBUS INTERFACE 95
9.1 Profibus Overview 95
9.2 Profibus-DP 95
9.3 Data Exchange 96
9.4 Protocol Implementation 97

9.4.1 Definitions 97
9.4.2 Control Field 98
9.4.3 SAP Field 101
9.4.4 Length Field 101
9.4.5 Application Data Buffer 102
9.5 Examples of Profibus Command/Response Mechanism 102
9.5.1 Example 1: Normal Command/Response Sequence 104
9.5.2 Example 2: Unsolicited Responses (Continuous Read Mode) 114
9.5.3 Example 3: Fragmentation of Responses 118
9.5.4 Example 4: Fragmentation of Commands 127
9.5.5 Example 5: Resynchronization 138

10 TECHNICAL FEATURES 143

REFERENCES

CONVENTIONS

This manual uses the following conventions:

“User” or “Operator” refers to anyone using a Subnet16™ Gateway.

“Device” refers to the Subnet16™ Gateway.

“You” refers to the System Administrator or Technical Support person using this manual to install, mount, operate, maintain or troubleshoot a Subnet16™ Gateway.

BIS M-41_ , BIS M-62_ and BIS U-62_ RFID Processors are referred to as Processors, or just “the Processor ”.

In addition, the terms “Subnet Node Number”, “Node ID” and “Processor units ID” are used interchangeably.

BIS Z-GW-001-DNT	correspond to the old name	GWY-01-DNT-01
BIS Z-GW-001-IND	correspond to the old name	GWY-01-IND-01
BIS Z-GW-001-PBS	correspond to the old name	GWY-01-PBS-01
BIS Z-GW-001-RS232	correspond to the old name	GWY-01-232-01
BIS Z-GW-001-TCP	correspond to the old name	GWY-01-TCP-01

REFERENCE DOCUMENTATION

The documentation related to the Subnet16™ Gateway management is available on the specific product page at the website:

www.balluff.com

SERVICES AND SUPPORT

Balluff provides several services as well as technical support through its website. Log on to **www.balluff.com** and click on the links indicated for further information including:

- **PRODUCTS**
  Search through the links to arrive at your product page which describes specific Info, Features, Applications, Models, Accessories, and Downloads including:
  - **Dashboard™**: a Windows-based utility program, which allows system testing, monitoring, and configuration using a PC. It provides Serial (RS232 or USB) and Ethernet interface configuration.
  - **C-Macro Builder™**: an easy to use GUI-driven utility for Windows. This software tool allows users with minimal programming experience to “build” their own macro programs (which are stored internally on and executed directly by RFID Processors).

REGULATORY AND COMPLIANCE NOTICES

This product is intended to be installed by Qualified Personnel only.

This product must not be used in explosive environments.

Only connect Ethernet and data port connections to a network which has routing only within the plant or building and no routing outside the plant or building.

CE

POWER SUPPLY

This product is intended to be installed by Qualified Personnel only.

This device is intended to be supplied by a UL Listed or CSA Certified Power Unit with «Class 2» or LPS power source.

GENERAL VIEW

RS232 Models



Figure A

1. Power On LED
2. Subnet16 BUS LED
3. Subnet16 Connector
4. RS232 Mode LED
5. Configuration Error LED
6. Mounting Bracket
7. RS232 Connector

IND / TCP Models



Figure B

1. Power On LED
2. Subnet16 BUS LED
3. Subnet16 Connector
4. Ethernet Mode LED
5. Configuration Error LED
6. Mounting Bracket
7. Ethernet Connector

DNT Models



Figure C

1. Power On LED
2. Subnet16 BUS LED
3. Subnet16 Connector
4. DeviceNet Mode LED
5. Configuration Error LED
6. Mounting Bracket
7. USB Connector
8. DeviceNet Connector

PBS Models

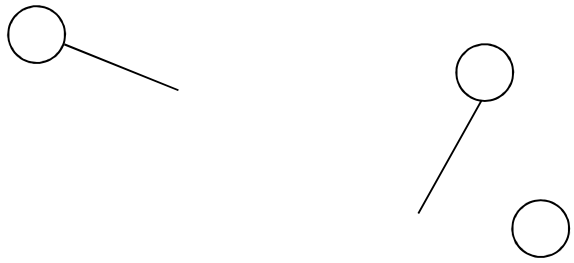


Figure D

1. Power On LED
2. Subnet16 BUS LED
3. Subnet16 Connector
4. Profibus Mode LED
5. Profibus Status LED
6. Configuration Error LED
7. Mounting Bracket
8. USB Connector
9. Profibus Connector

1 OVERVIEW

1.1 INTRODUCTION

Welcome to the **Subnet16™ Gateway BIS Z-GW-001_ Manual**. This manual will assist you in the installation, configuration and operation of the Subnet16™ Gateway Interface Modules.

The Gateway can control up to 16 passive ultra-high or high frequency read/write Radio-Frequency Identification (*RFID*) controllers. In order to meet and exceed the requirements of the industrial automation industry, the Gateway and Balluff RFID controllers are designed to be compact, reliable and rugged.

1.2 SUBNET16™ GATEWAY FEATURES

- Multidrop capable; controls up to 16 RFID reader/writers, each functioning independently and simultaneously.
- Operational power is supplied directly from the Subnet16™ network
- Small footprint provides ease of mounting
- Available support for multiple communication protocols: Subnet16™, standard TCP/IP, Ethernet/IP, MODBUS TCP, DeviceNet, Profibus-DP-V1
- Supports several interface connections: RS232, Ethernet, DeviceNet, Profibus
- Supports Balluff's CBx RFID command protocols
- LED status indicators for Power/Ready, Subnet16 bus activity, network status, Configuration error indication
- Supports processor unit macro functionality
- Flash memory for software updates
- Real-time Calendar/Date functions
- Auto configuration of RFID processor units, automatic Node ID number assignment
- Node Fault Detection
- Isolated bus interfaces

1.3 ABOUT THIS MANUAL

This manual provides guidelines and instructions for installing, configuring and operating Subnet16™ Gateway Interface Modules.

This document does NOT include explicit details regarding the Gateway BIS Z-GW-001_ RFID commands. Specific RFID command related information such as: the process of issuing commands from a host PC or Programmable Logic Controller (PLC) to the Gateway Interface Module is available in the CBx Command Protocol – Manual, which is available at www.balluff.com.

1.3.1 Who Should Read This Manual?

This manual should be read by those who will be installing, configuring and operating the Gateway. This may include the following people:

- Hardware Installers
- System Integrators
- Project Managers
- IT Personnel
- System and Database Administrators
- Software Application Engineers
- Service and Maintenance Engineers

1.3.2 HEX Notation

Throughout this manual, numbers expressed in Hexadecimal notation are prefaced with "**0x**". For example, the number "10" in decimal is expressed as "0x0A" in hexadecimal.

1.4 MODELS AND ACCESSORIES

Balluff designs, manufactures and distributes a wide range of RFID equipment including RFID controllers, network interface modules (Gateways and Hubs), RFID tags and the cables needed to make it all work.

Listed here are the products and accessories relative to the Subnet16™ Gateway product family. For a complete list of products and accessories relative to the RFID processor units see the relative Processor unit Manual.

To purchase any of the Balluff products listed below contact your Balluff distributor or visit our Web site: http://www.balluff.com.

Name	Description	Order Code
RFID Controllers		
BIS M-410-067-001-04-S92	Compact processors- 4x5 cm w/Antenna Subnet16™	BIS00W1
BIS M-411-067-001-04-S92	Compact processors- 10x7 cm w/Antenna Subnet16™	BIS00W5
BIS M-620-067-A01-04-S92	BIS M-62_ Processor units - RS485 Subnet16™	BIS00ZL
BIS M-620-067-A01-04-ST30	BIS M-62_ Processor units - RS485 Subnet16™ w I/O	BIS00ZK
BIS U-620-067-101-04-S92	BIS U-62_ Processor unit- RS485 Subnet16™	BIS00Z9
BIS U-620-067-101-04-ST30	BIS U-62_ Processor unit- RS485 Subnet16™ w I/O	BIS00Z8
BIS U-620-067-111-04-S92	BIS U-62_ Processor unit- RS485 Subnet16™	BIS00Z7
BIS U-620-067-111-04-ST30	BIS U-62_ Processor unit- RS485 Subnet16™ w I/O	BIS00Z6
Cables & Connectors		
BCC M418-D279-BF-701-PS0825-020	RS232 Cable: M12, DB9-pin, PS wires	BCC0ETJ
BCC M415-M415-3A-330-PS85N6-003	Cable: M12, 5-pin, Male/Female, ThinNet, 0.3 m	BCC0ERY
BCC M415-M415-3A-330-PS85N6-010	Cable: M12, 5-pin, Male/Female, ThinNet, 1 m	BCC0ERZ

Name	Description	Order Code
BCC M415-M415-3A-330-PS85N6-020	Cable: M12, 5-pin, Male/Female, ThinNet, 2 m	BCC0ET0
BCC M415-M415-3A-330-PS85N6-050	Cable: M12, 5-pin, Male/Female, ThinNet, 5 m	BCC0ET1
BCC M415-M415-6A-330-PS85N6-002	Cable: M12, 5-pin, Male/Male, ThinNet, 0.2 m (Gateway to Drop-T)	BCC0ET2
BCC M415-M415-6A-330-PS85N6-010	Cable: M12, 5-pin, Male/Male, ThinNet, 1 m (Gateway to Drop-T)	BCC0ET3
BCC M415-M415-6A-330-PS85N6-020	Cable: M12, 5-pin, Male/Male, ThinNet 2 m (Gateway to Drop-T)	BCC0ET4
BCC A315-A315-30-330-PS85N4-020	Cable: 7/8–16, 5-pin, Male/Female, ThickNet, 2 m	BCC095A
BCC A315-A315-30-330-PS85N4-050	Cable: 7/8–16, 5-pin, Male/Female, ThickNet, 5 m	BCC095F
BCC M415-0000-1A-030-PS85N6-020	Cable: M12, 5-pin, Female / Bare Wires, ThinNet, 2 m	BCC0ETA
BCC M415-0000-1A-030-PS85N6-050	Cable: M12, 5-pin, Female / Bare Wires, ThinNet, 5 m	BCC0ETC
BCC A315-0000-10-030-PS85N4-050	Cable: 7/8-16, 5-pin, Female / Bare Wires, 5M	BCC096Y
BCC A315-0000-10-030-PS85N6-050	Cable: M12, 5-pin, Male / Bare Wires, ThinNet, 2M	BCC08WT
BCC M414-E834-8G-672-ES64N8-050	Industrial Ethernet Cable: M12, RJ45 5 m	BCC0CT1
Subnet16™ Ts, Terminators, Connectors		
BDN T-DTE-AD-01	Drop-T Connector: 5-pin, 7/8-16 F / M12 F / 7/8-16 M (ThickNet to ThinNet)	BCC07WZ
BDN T-DTN-DD-01	Drop-T Connector: M12, 5-pin, F/F/M (ThinNet to ThinNet)	BCC07WR
BCC M435-0000-1A-000-41X575-000	Field Mountable Connector: M12, 5-pin, Female, Straight	BCC06ZF
BCC A315-0000-2A-R04	Termination Resistor Plug: 7/8-16, 5-pin, Male, (ThickNet)	BCC0A09
BCC M415-0000-2A-R04	Termination Resistor Plug: M12, 5-pin, Male, (ThinNet)	BCC09MR
BCC M438-0000-1A-000-51X850-000	RS232 Connector: M12, 8-pin, Female	BCC0A03
BCC A315-0000-1A-R04	Plug: Termination Resistor, M12, 5-pin, Female (ThinNet)	BCC0A0A
BCC M415-0000-1A-R04	Plug: Termination Resistor, 7/8-16, 5-pin, Female (ThickNet)	BCC0A08
BCC A335-0000-10-000-61X5A5-000	Field Mountable Connector: 7/8-16, 5-pin, Female, Straight	BCC070F
BDN T-DTE-AA-01	T Connector: 7/8-16/5P M/F/F (ThickNet to ThickNet)	BCC07WP

2 INSTALLATION

2.1 MECHANICAL DIMENSIONS

2.1.1 RS232 Models

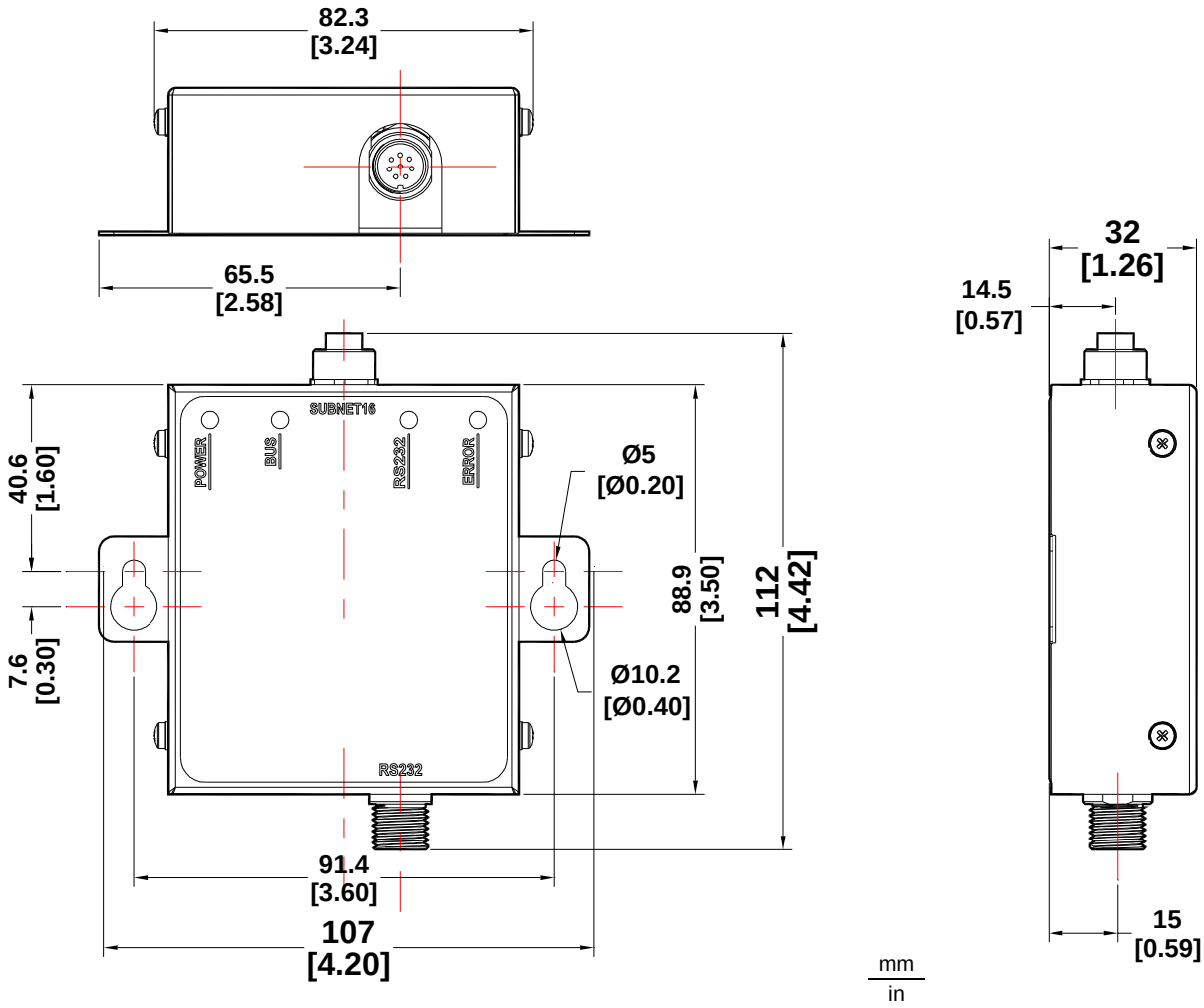


Figure 1 - RS232 Dimensions

2.1.2 IND / TCP Models

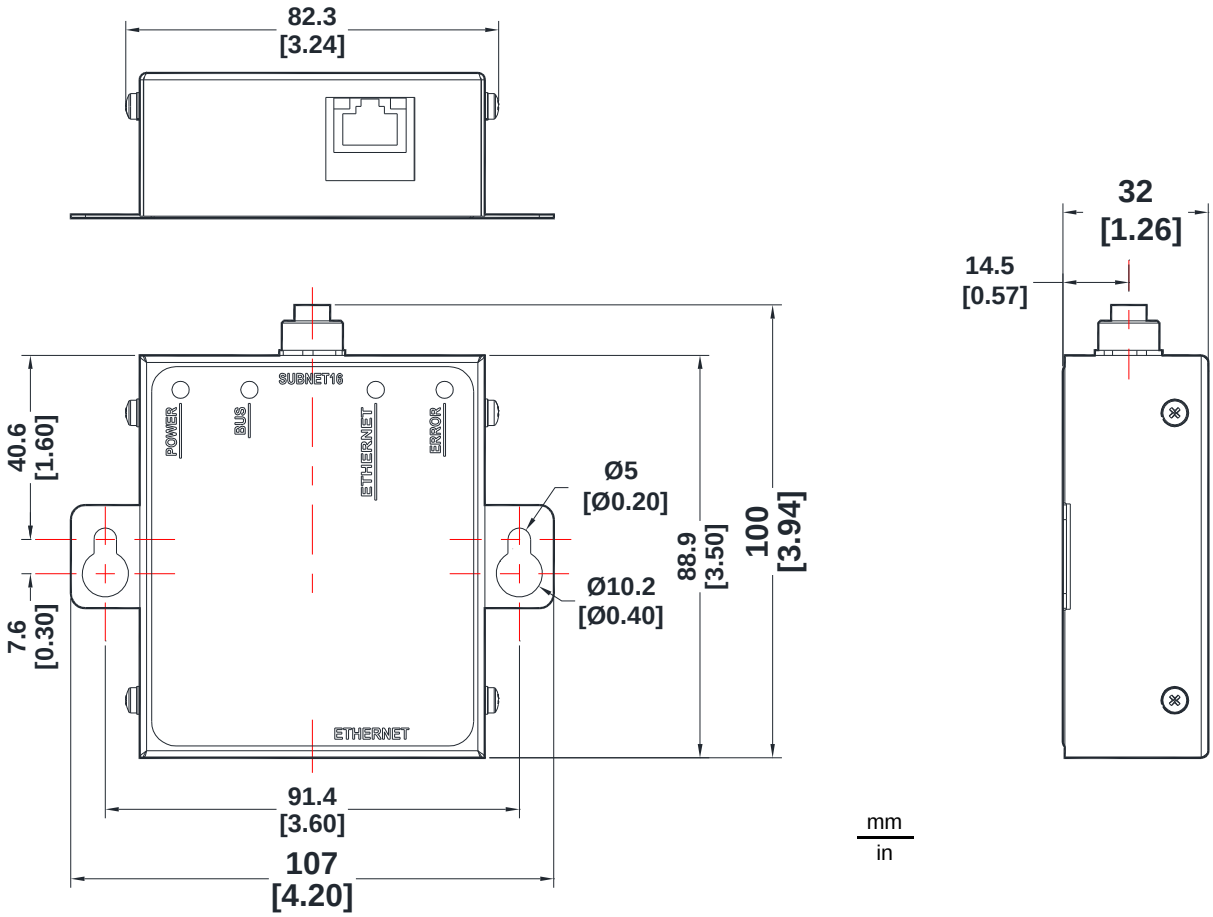


Figure 2 - IND / TCP Dimensions

2.1.3 DNT Models

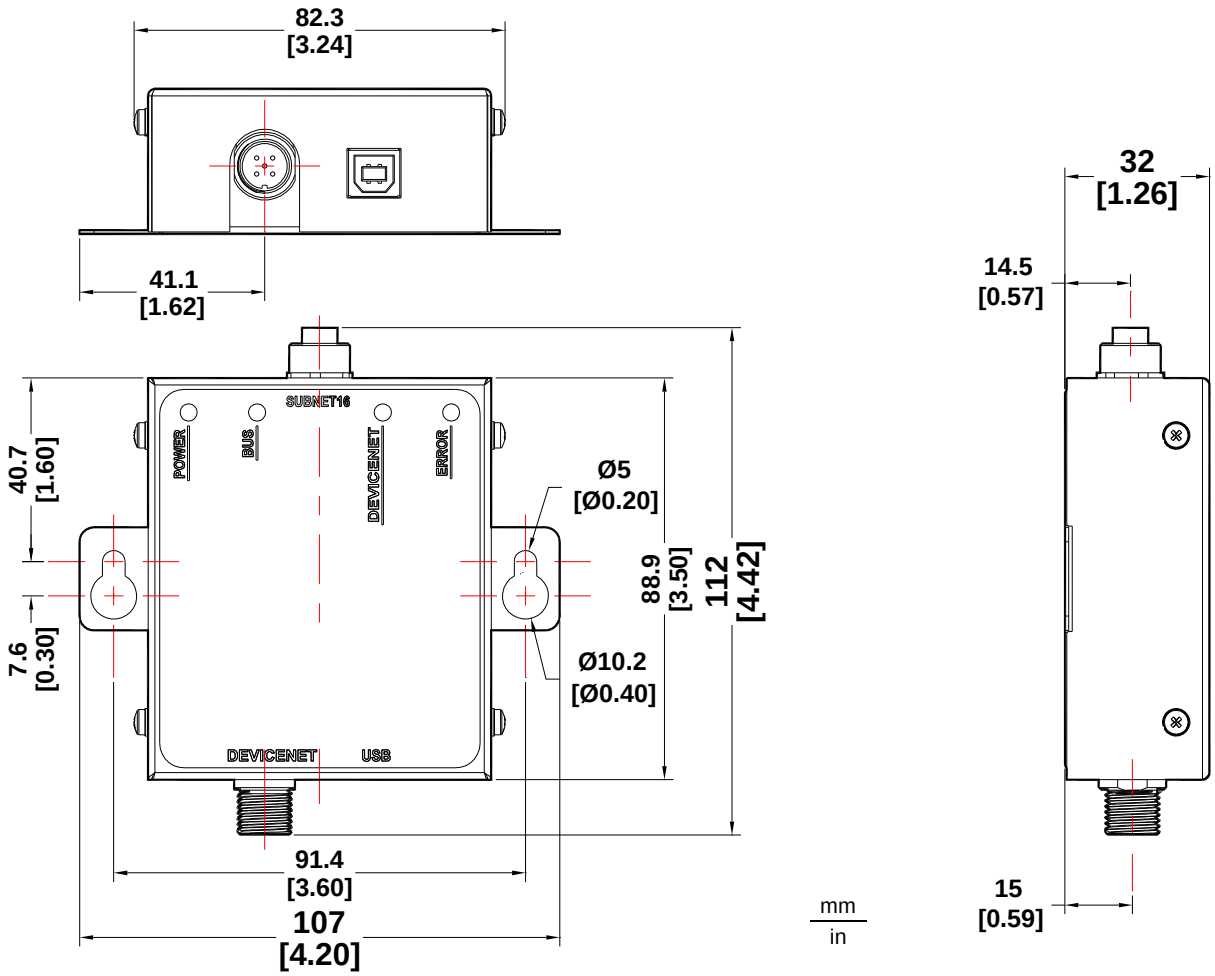


Figure 3 - DNT Dimensions

2.1.4 PBS Models

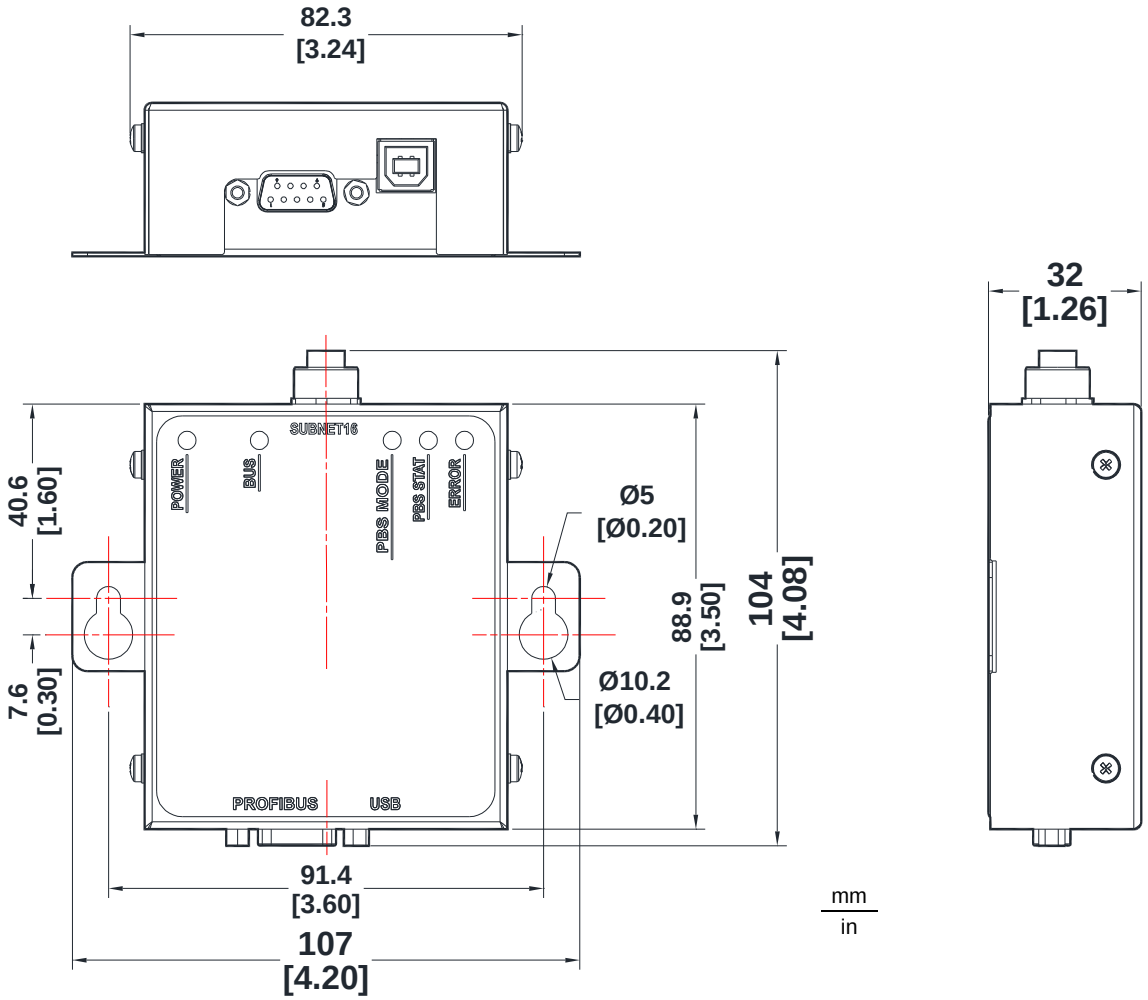


Figure 4 - PBS Dimensions

2.2 ELECTRICAL CONNECTORS

2.2.1 Subnet16™

The Subnet16™ Connector (M12 5-pin, Female) is used for Data and Power Supply connections for the entire Subnet16™ network (Gateway[1] and Processor units).

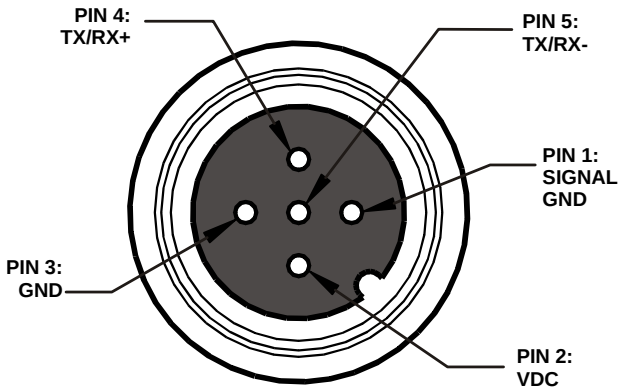


Figure 5 - Subnet16™ M12 5-pin Female Connector

Pin	Name	Function
1	SGND	Subnet16™ Signal Ground
2	VDC	Subnet16™ Input Power
3	GND	Subnet16™ Power Ground
4	TX/RX+	RS485 Transmit/Receive Data positive
5	TX/RX-	RS485 Transmit/Receive Data negative

[1] Except for DeviceNet models. See par. 1.2.4.

2.2.2 RS232

The RS232 Connector (M12 8-pin, Male) is used for a point-to-point serial connection between a host computer and the Subnet16™ RS232 Gateway Interface Module.

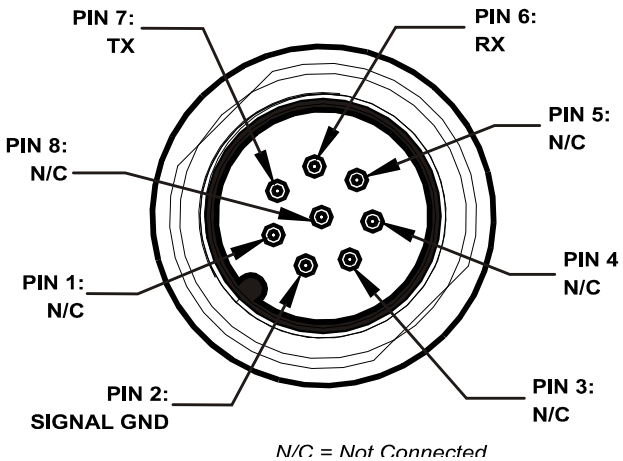


Figure 6 - RS232 Interface M12 8-pin Male Connector

Pin	Name	Function
1	nc	
2	SGND	Signal Ground
3	nc	
4	nc	
5	nc	
6	RX	RS232 Receive Data
7	TX	RS232 Transmit Data
8	nc	

2.2.3 Industrial Ethernet IP

The Ethernet Connector (RJ45, Female) is used for connecting the Subnet16™ IND or TCP Gateway Interface Module to an Ethernet network.

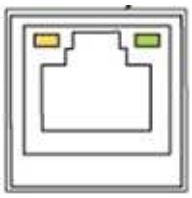


Figure 7 - Ethernet Interface RJ45 Female Connector

Pin	Name	Function
1	TX+	Transmit Data positive
2	TX-	Transmit Data negative
3	RX+	Receive Data positive
4		
5		
6	RX-	Receive Data negative
7		
8		

The Gateway can also be connected to a portable PC for configuration through this Ethernet Connection.

2.2.4 DeviceNet

The DeviceNet Connector (M12 5-pin, Male) is used for connecting the Subnet16™ DeviceNet Gateway Interface Module to a DeviceNet network.

NOTE

These models are powered from the DeviceNet network power supply (they are isolated from the Subnet16™ network power). The Subnet16™ Connector provides power to the Subnet16™ processor units.

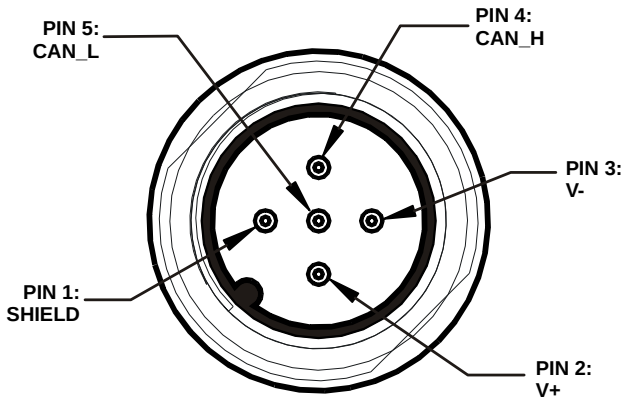


Figure 8 - DeviceNet M12 5-pin Male Connector

Pin	Name	Function
1	Shield	DeviceNet Bus Shield
2	V+	DeviceNet Bus Power (Powers Gateway)
3	V-	DeviceNet Bus Ground (Powers Gateway)
4	Can_H	Data positive
5	Can_L	Data negative

The USB Connector (Type B, Female) on the DeviceNet models is used for connecting the Gateway to a portable PC for configuration.

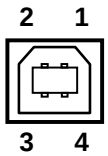


Figure 9 - USB Type B Female Connector

Pin	Name	Function
1	Vdc	5 + Vdc USB power source
2	D-	Data negative
3	D+	Data positive
4	GND	USB power source Ground

2.2.5 Profibus

The Profibus Connector (D-sub 9-pin, Female) is used for connecting the Subnet16™ Profibus Gateway Interface Module to a Profibus network.

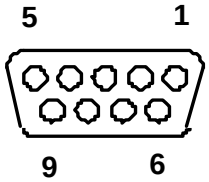


Figure 10 - D-sub 9-pin Female Connector (Data and Bus Termination Power)

Pin	Name	Function
1	nc	
2	nc	
3	B Line (+)	Data positive
4	RTS	Request To Send
5	GND	Bus Ground for termination
6	+5 Vdc	Bus Power for termination
7	nc	
8	A Line (-)	Data negative
9	nc	

The USB Connector (Type B Female Connector) on the Profibus models is used for connecting the Gateway to a portable PC for configuration.

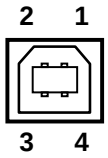


Figure 11 - USB Type B Female Connector

Pin	Name	Function
1	Vdc	5 + Vdc USB power source
2	D-	Data negative
3	D+	Data positive
4	GND	USB power source Ground

2.3 POWER & WIRING

The information presented below is provided to assist the installer in determining the amount of power that will be required by the Gateway and its Subnet16™ network depending on the application.

2.3.1 Power Requirements

The Gateway requires an electrical supply voltage of 12 to 30 Vdc. In addition, each RFID processor unit connected to the Gateway via the Subnet16™ network will also require power. Use a regulated power supply that is capable of delivering the requirements listed in the Technical Features.

The following information is provided to assist you in determining the power requirements of an RFID network application, in particular a Subnet16™ network.



NOTE

Power is applied directly to the Subnet16™ Network trunk and distributed through drop cables to the Gateway and RFID processor units. By positioning the power supply near the middle of the network, you can limit voltage drop at the ends.

2.3.2 Total System Current Consumption



NOTE

The current consumption values of each product are given in the Technical Features paragraph of the relative Installation manual and refer to the min and max input voltage range. These values already include an adequate safety margin. The consumption values given in the following examples have been interpolated for an input voltage of 24 Vdc.

Max Gateway Current: 200 mA @ 12 Vdc (133 mA @ 24 Vdc).

Max Processor unit Current: 366 mA @ 24 Vdc for BIS M-62_ and 87 mA @ 24 Vdc for BIS M-41_, etc. (refer to processor unit's spec).

Calculating Total System Current Consumption:

Total System Current Consumption = [Max Gateway Current + (Max Processor unit Current x Number of Processor units)]

Example

A Subnet16™ network powered at 24 Vdc is composed of a BIS Z-GW-001_ connecting eight BIS M-620-067-A01-04_ Processor units.

Total System Current Consumption = [0.133 A + (0.366 A X 8)] = 3.061 A

2.3.3 Cable Voltage Drop

In addition, each RFID processor unit on the Subnet will experience a certain amount of voltage drop depending on the length of the cable.

Cable Resistance per Meter

- ThinNet = **0.058 ohms** per meter per wire
- ThickNet = **0.0105 ohms** per meter per wire

Calculating Voltage Drop

Voltage Drop = (Max Processor unit Current x Number of Processor units) x (Cable Resistance per Meter per Wire[2] x Cable length in Meters)

Example

A Subnet16™ network is composed of a BIS Z-GW-001_ connecting eight BIS M-620-067-A01-04_ Processor units (366 mA each @ 24 Vdc). A total of 20 meters of ThinNet cables are used to connect the devices, which have Cable Resistance = 0.058 Ohms per meter per wire. The network power is 24 Vdc.

The voltage drop calculation must be conducted on the RFID processor unit that is farthest from the Power Supply, as it will experience the greatest voltage drop.



NOTE

It is always recommended to power the network from the middle (T-configuration), to reduce total voltage drop at the ends. In the example below this allows the fourth processor unit and not the eighth to be the furthest from the power supply.

Voltage Drop = [0.133 A GWY + (0.366 A x 8 processor units)] x [(0.058 x 2) x 20 meters] = 7.10 Vdc total voltage drop for 8 processor units

24 Vdc - 7.10**/2** = 20.45 Vdc at processor unit number 4 of each branch

2.3.4 Maximum Supported Trunk and Drop Cable Lengths

- ThickNet trunk length up to 300 m.
- ThinNet trunk length up to 20 m.
- ThinNet drop cable length up to 2 m.

2.3.5 Current Rating for Cables

The maximum current rating for the Subnet16™ network using Balluff's cables and accessories (BCCxxxx), is **4.0 A**.

[2] The resistance calculation must include both wires (Vdc and GND).

2.4 INSTALLATION GUIDELINES

2.4.1 Hardware Requirements

The following is a list of minimum components required to create an RFID Subnet16™ reading system.

- Host computer with specific interface (Serial, or Fieldbus); Programmable Logic Controller unit (PLC) or PC
- (Serial or Fieldbus) Subnet16™ Gateway Interface Module
- RFID processor units (*BIS M-41_, BIS M-62_ or BIS U-62_ Series - **only RS485 models**)*
- Adequate length cabling, connectors and terminators
- Sufficient power capable of powering all the RFID components
- Balluff RFID data carrier or labels: BIS M-1xx or BIS U-1xx

2.4.2 Installation Precautions

- Do not route cables near other unshielded cables or near wiring carrying high voltage or high current. Cross cables at perpendicular intersections and avoid routing cables near motors and solenoids.
- Review the power requirements of your RFID network and provide a suitable power supply.
- Avoid mounting the processor unit near sources of EMI (electro-magnetic interference) or near devices that generate high ESD (electro-static discharge) levels. Always use adequate ESD prevention measures to dissipate potentially high voltages.
- If electrical interference is encountered (as indicated by a significant reduction in read/write performance), relocate the processor unit to an area free from potential sources of interference.
- Perform a test phase by constructing a small scale, independent network that includes only the essential devices required to test your RFID application (use Balluff approved Subnet16™ cables and accessories).

2.5 TYPICAL LAYOUTS AND INSTALLATION PROCEDURES

The BIS Z-GW-001-RS232 Gateway unit is designed for point-to-point applications, where the distance from host to Gateway is less than 15 meters (50 feet). The Gateway connects directly to a serial communications port on a host computer via an RS232-compatible serial interface cable.

2.5.1 Installing the Subnet16™ Gateway

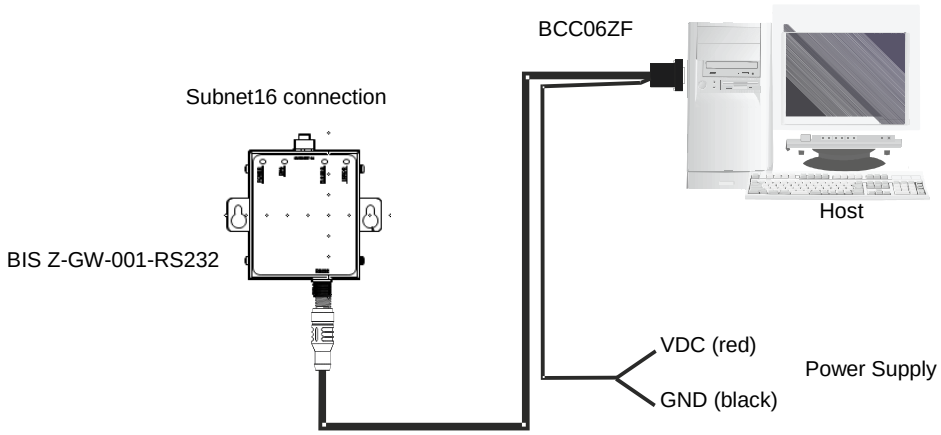


Figure 12 - RS232 Typical Layouts

1. **Preliminary Notes**: Read this procedure in its entirety and note the *Installation Guidelines* in par. 1.4.

2. **Mounting**: Mount the Gateway to your chosen location using two M5 (#10) screws, lock washers and nuts. The Gateway may be mounted in any orientation, but should be aligned in such a manner that the LED indicators can be seen during operation.

3. **Gateway Connection**: Attach one end of a 5-pin, male-to-male, M12, ThinNet drop cable (***P/N: BCC0ET4***) to the 5-pin, female, M12 connector on the Gateway. Connect the other end of this 5-pin, male-to-male, M12, ThinNet drop cable to the 5-pin, female, M12 connector on a ***ThinNet to ThinNet Drop-T Connector*** (as per your network and RFID application requirements).

4. **Trunk Wiring**: Attach one end of a male-to-female trunk cable to each mating connector on the Drop-T Connector. Continue connecting trunk cables and Drop-T connectors as needed.

   Note: trunk length should not exceed 300 m for ThickNet and 20 m for ThinNet.

5. **Termination Resistors**: For ThinNet Networks: Connect a Terminating Resistor (***P/N: BCC09MR, male)*** to the first and last Drop-T Connector on the trunk line.

6. **RFID Processor unit Connection**: Connect the male end of a 5-pin, male-to-female, ThinNet drop cable to the female end on your Drop-T connector(s). Attach the remaining female end of the ThinNet drop cable to the 5-pin, male, M12 connector on a ***BIS M-41x, BIS M-62x or BIS U-62x- Series Processors - (RS485 models)*** Repeat this step for each RFID processor unit you plan to install.

Note: maximum drop cable length is 2 m.

7. **Power Supply Wiring**: For ThinNet Networks: Using a 5-pin, female, M12, ThinNet connector (***P/N: BCC06ZF***), make a power cable and connect it to your power supply (SHIELD wire connected to Earth). Attach the female, ThinNet end to the 5-pin, male, ThinNet end on a Drop-T connector (***P/N: BCC07WR)***.

8. **Host Connection**: Connect the Gateway to your host computer via an RS232-compatible serial interface cable.

9. **Power On**: Turn the power supply ON. The POWER LED on the Gateway will remain lit while power is applied to the unit.

10. **Automatically Configure Subset16™ Node IDs**: At this point all processor units are powered and should have Node IDs set to 00, (all processor unit Node LEDs = OFF), and Subnet16™ baudrate = 9600 (factory defaults).

    a. Place the RFID Processor unit Configuration Tag in front of an Processor unit (the processor unit's RF Activity LED blinks once indicating the tag has been read), and wait for the Gateway to assign a valid Node ID to it. The processor unit's Node LEDs now indicate a valid Node ID. Remove the Configuration Tag from the processor unit.

    b. Repeat this step for each node in the Subnet16™ network (one processor unit at a time). The first is Node ID 1, then 2 and so on up to 16 (binary).

The Subnet16™ network is now configured with the default values and can communicate with the RS232 Gateway which in turn communicates with the RS232 Host.

To verify operations or for application specific configuration, use the Balluff Dashboard™ Configuration Tool downloadable from www.balluff.com. The Balluff Dashboard™ Configuration Tool allows users to configure and control their Subnet16™ Gateway network and send RFID commands for testing purposes. See the Balluff Dashboard™ Manual for details.

Figure 13 – RS232 Typical Layouts

2.5.2 Installing the Industrial Ethernet (IND) Subnet16™ Gateway

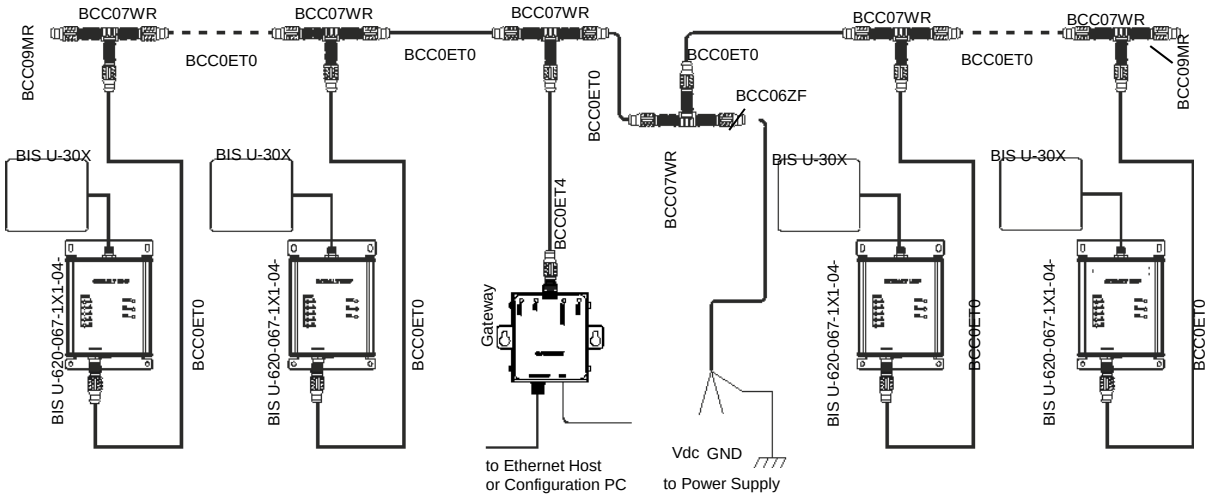


Figure 14 - IND Typical Layouts

The BIS Z-GW-001-IND Gateway is designed for Industrial Ethernet IP applications, where the processor unit is connected in an Ethernet IP TCP/IP network via compatible cables through a hub or directly to an Ethernet IP host.

1. Select a suitable location for the Gateway and Processor unit/Antenna
2. Mount the Gateway to your chosen location using two M5 (#10) screws, lock washers and nuts. The Gateway may be mounted in any orientation, but should be aligned in such a manner that the LED indicators can be seen during operation.
3. Gateway Connection: Attach one end of a 5-pin, male-to-male, M12, ThinNet drop cable (P/N: BCC0ET4) to the 5-pin, female, M12 connector on the Gateway. Connect the other end of this 5-pin, male-to-male, M12, ThinNet drop cable to the 5-pin, female, M12 connector on a ThinNet to ThinNet Drop-T Connector (as per your network and RFID application requirements).
4. Trunk Wiring: Attach one end of a male-to-female trunk cable to each mating connector on the Drop-T Connector. Continue connecting trunk cables and Drop-T connectors as needed.
5. Note: trunk length should not exceed 300 m for ThickNet and 20 m for ThinNet.
6. Termination Resistors: For ThinNet Networks: Connect a Terminating Resistor (P/N: BCC09MR, male) to the first and last Drop-T Connector on the trunk line.
7. Power Supply Wiring: For ThinNet Networks: Using a 5-pin, female, M12, ThinNet connector (P/N: BCC06ZF), make a power cable and connect it to your power supply (SHIELD wire connected to Earth). Attach the female, ThinNet end to the 5-pin, male, ThinNet end on a Drop-T connector (P/N: BCC07WR).
8. Host Connection: Connect the Gateway to your host computer via Category 5e Ethernet cabling. A crossover cable may be required if you are connecting the Gateway directly to a computer (rather than to a switch, network hub or router).
9. Power On: Turn the power supply ON. The POWER LED on the Gateway will remain lit while power is applied to the unit.

10. Automatically Configure Subset16™ Node IDs: At this point all processors are powered and should have Node IDs set to 00, (all processor Node LEDs = OFF), and Subnet16™ baudrate = 9600 (factory defaults).
    - **a.** Place the BIS M- or BIS U-Series Configuration Tag in front of an Processor (the Processor 's RF Activity LED blinks once indicating the tag has been read), and wait for the Gateway to assign a valid Node ID to it. The processor's Node LEDs now indicate a valid Node ID. Remove the Configuration Tag from the processor.
    - **b.** Repeat this step for each node in the Subnet16™ network (one processor at a time). The first is Node ID 1, then 2 and so on up to 16 (binary).

The Subnet16™ network is now configured with the default values and can communicate with the Industrial Ethernet Gateway which in turn communicates with the Industrial Ethernet Host.

BIS Z-GW-001-IND FACTORY DEFAULT ADDRESS:
(192.168.253.110)

To verify operations or for application specific configuration, use the Balluff Dashboard™ Configuration Tool downloadable from www.balluff.com. The Balluff Dashboard™ Configuration Tool allows users to configure and control their Subnet16™ Gateway network and send RFID commands for testing purposes. See the Balluff Dashboard™ Manual for details.

2.5.3 Installing the DeviceNet (DNT) Subnet16™ Gateway

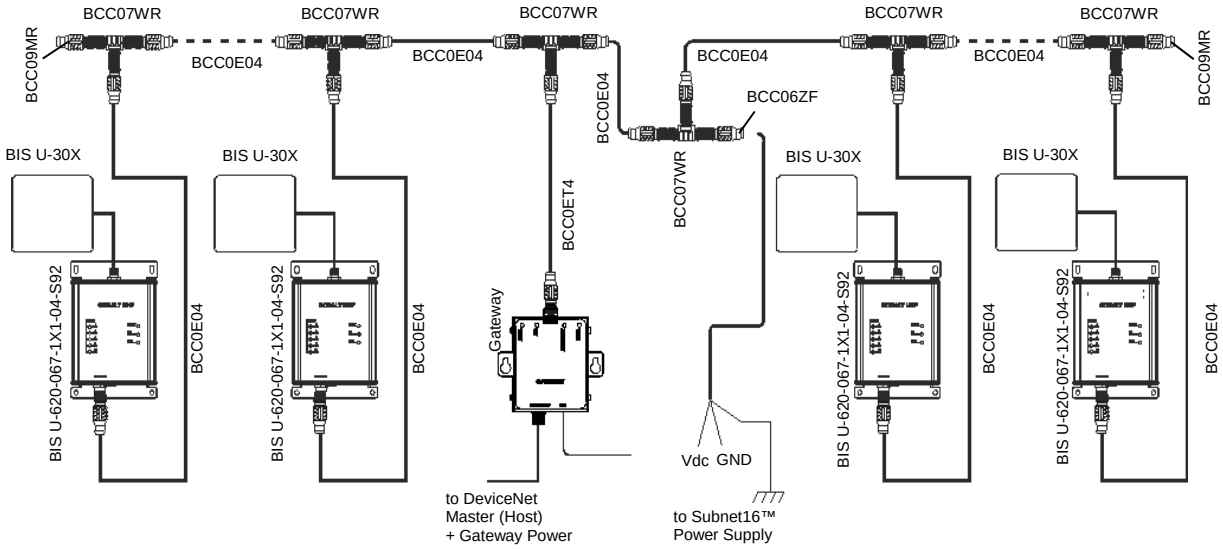


Figure 15 - DNT Typical Layouts

The HF-CNTL-DNT-02 RFID Processor unit is designed for DeviceNet RFID applications, where the processor unit is connected as a slave node in a DeviceNet network via compatible cables directly to a DeviceNet Master/Scanner (host). **The default Node ID is 63**.

1. Select a suitable location for the Gateway and Processor unit/Antenna.
2. Mounting: Mount the Gateway to your chosen location using two M5 (#10) screws, lock washers and nuts. The Gateway may be mounted in any orientation, but should be aligned in such a manner that the LED indicators can be seen during operation.
3. Gateway Connection: Attach one end of a 5-pin, male-to-male, M12, ThinNet drop cable (P/N: BCC0ET4) to the 5-pin, female, M12 connector on the Gateway. Connect the other end of this 5-pin, male-to-male, M12, ThinNet drop cable to the 5-pin, female, M12 connector on a ThinNet to ThinNet Drop-T Connector (as per your network and RFID application requirements).
4. Trunk Wiring: Attach one end of a male-to-female trunk cable to each mating connector on the Drop-T Connector. Continue connecting trunk cables and Drop-T connectors as needed.

   Note: trunk length should not exceed 300 m for ThickNet and 20 m for ThinNet.
5. Termination Resistors: For ThinNet Networks: Connect a Terminating Resistor (P/N: BCC9MR, male) to the first and last Drop-T Connector on the trunk line.
6. RFID Controller Connection: Connect the male end of a 5-pin, male-to-female, ThinNet drop cable to the female end on your Drop-T connector(s). Attach the remaining female end of the ThinNet drop cable to the 5-pin, male, M12 connector on a *BIS U*, *BIS M-6xx* or *BIS M-41x (RS485 models)*. Repeat *Step 5* for each RFID controller you plan to install.

   Note: maximum drop cable length is 2 m.

   Subnet16™ Power Supply Wiring: For ThinNet Networks: Using a 5-pin, female, M12, ThinNet connector (*P/N: BCC06ZF*), make a power cable and connect it to your power

supply (SHIELD wire connected to Earth). Attach the female, ThinNet end to the 5-pin, male, ThinNet end on a Drop-T connector *(P/N: BCC07WR).*

7. Gateway Power and Host Connection: Connect the Gateway to the DeviceNet network via a DeviceNet-compatible interface cable. The Gateway must be powered through pins V+ and V- of the DeviceNet connector.
8. Power On: Turn both power supplies ON. The POWER LED on the Gateway will remain lit while power is applied to the unit through the DeviceNet connector.
9. Automatically Configure Subset16™ Node IDs: At this point all controllers are powered and should have Node IDs set to 00, (all controller Node LEDs = OFF), and Subnet16™ baudrate = 9600 (factory defaults).
    - **a.** Place the RFID Controller Configuration Tag in front of an RFID Controller (the controller's RF Activity LED blinks once indicating the tag has been read), and wait for the Gateway to assign a valid Node ID to it. The controller's Node LEDs now indicate a valid Node ID. Remove the Configuration Tag from the controller.
    - **b.** Repeat this step for each node in the Subnet16™ network (one controller at a time). The first is Node ID 1, then 2 and so on up to 16 (binary).

The Subnet16™ network is now configured with the default values and can communicate with the DeviceNet Gateway which in turn communicates with the DeviceNet Master.

BIS Z-GW-001-DNT DEVICENET FACTORY DEFAULTS:
Node Address: 63; Baud Rate: 125 kbps

For further information or for application specific configuration using the Dashboard Configuration Tool program, see the Gateway Manual.

To configure and control the BIS Z-GW-001-DNT Gateway and send RFID commands for testing purposes, download and install the Balluff Dashboard™ Configuration Tool from www.balluff.com. The Dashboard™ Configuration Tool uses the USB port to communicate to the gateway USB port (Type B). To enable communication:

The USB Connector (Type B, Female) on the DeviceNet models is used for connecting the Gateway to a portable PC for configuration.

1. Connect the gateway USB port (Type B, Female) to the PC by using a standard USB A/B cable.
2. On the host computer, set COM port parameters to: 9600 baud, 8 data bits, 1 stop bit, no parity and no handshaking.
3. Run the Balluff Dashboard™ Configuration Tool.

2.5.4 Installing the Profibus (PBS) Subnet16™ Gateway

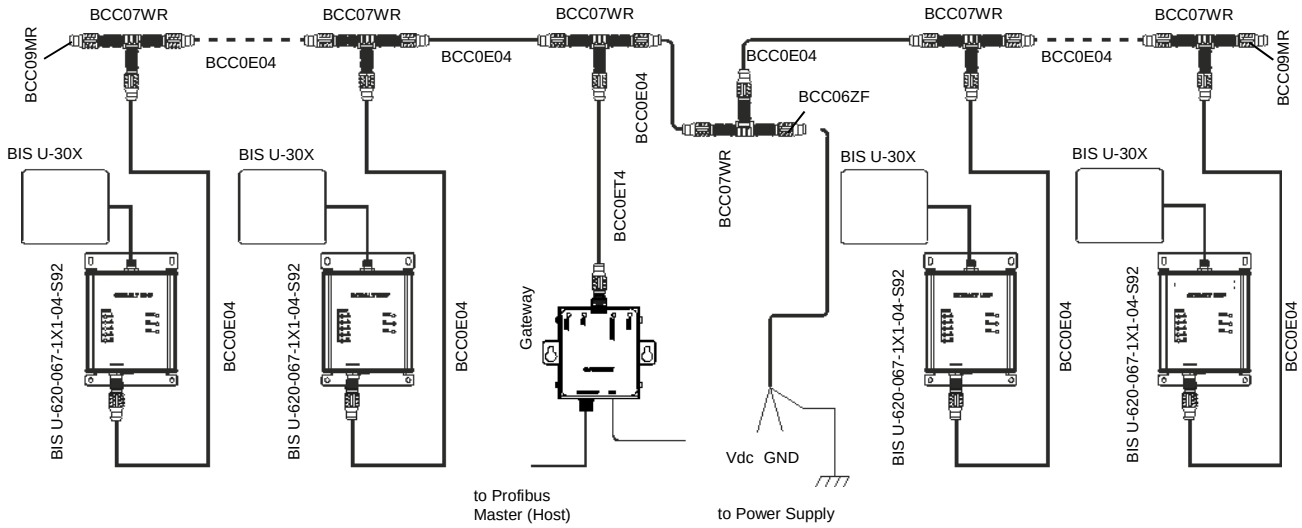


Figure 16 - PBS Typical Layouts

The BIS Z-GW-001-PBS Gateway is designed for Profibus RFID applications, where the Gateway is connected as a slave node in a Profibus (DP) network via compatible cables directly to a Profibus Master (host). **The default Node ID is 63**.

1. Select a suitable location for the Gateway and Processor unit/Antenna.
2. Mounting: Mount the Gateway to your chosen location using two M5 (#10) screws, lock washers and nuts. The Gateway may be mounted in any orientation, but should be aligned in such a manner that the LED indicators can be seen during operation.
3. Gateway Connection: Attach one end of a 5-pin, male-to-male, M12, ThinNet drop cable (*P/N: BCC0ET4*) to the 5-pin, female, M12 connector on the Gateway. Connect the other end of this 5-pin, male-to-male, M12, ThinNet drop cable to the 5-pin, female, M12 connector on a *ThinNet to ThinNet Drop-T Connector* (as per your network and RFID application requirements).
4. Trunk Wiring: Attach one end of a male-to-female trunk cable to each mating connector on the Drop-T Connector. Continue connecting trunk cables and Drop-T connectors as needed.

Note: trunk length should not exceed 300 m for ThickNet and 20 m for ThinNet.

5. Termination Resistors: For ThinNet Networks: Connect a Terminating Resistor (*P/N: BCC9MR, male)* to the first and last Drop-T Connector on the trunk line.
6. RFID Processor Connection: Connect the male end of a 5-pin, male-to-female, ThinNet drop cable to the female end on your Drop-T connector(s). Attach the remaining female end of the ThinNet drop cable to the 5-pin, male, M12 connector on a *BIS U*, *BIS M-6xx* or *BIS M-41x (RS485 models)*. Repeat *Step 5* for each RFID processor you plan to install.

Note: maximum drop cable length is 2 m.

7. Power Supply Wiring: For ThinNet Networks: Using a 5-pin, female, M12, ThinNet connector (*P/N: BCC06ZF*), make a power cable and connect it to your power supply (SHIELD wire connected to Earth). Attach the female, ThinNet end to the 5-pin, male, ThinNet end on a Drop-T connector (*P/N: BCC07WR)*.

8. Host Connection: Connect the Gateway to the Profibus network via a Profibus-compatible interface cable.
9. Power On: Turn the power supply ON. The POWER LED on the Gateway will remain lit while power is applied to the unit.
10. Automatically Configure Subset16™ Node IDs: At this point all processors are powered and should have Node IDs set to 00, (all processor Node LEDs = OFF), and Subnet16™ baudrate = 9600 (factory defaults).
    - **a.** Place the RFID Processor Configuration Tag in front of an RFID Processor (the processor's RF Activity LED blinks once indicating the tag has been read), and wait for the Gateway to assign a valid Node ID to it. The processor's Node LEDs now indicate a valid Node ID. Remove the Configuration Tag from the processor.
    - **b.** Repeat this step for each node in the Subnet16™ network (one processor at a time). The first is Node ID 1, then 2 and so on up to 16 (binary).

The Subnet16™ network is now configured with the default values and can communicate with the Profibus Gateway which in turn communicates with the Profibus Master.

BIS Z-GW-001-PBS PROFIBUS FACTORY DEFAULTS:
Node Address: 63; Input / Output Buffer size: 64 bytes

To configure and control the BIS Z-GW-001-PBS Gateway and send RFID commands for testing purposes, download and install the Balluff Dashboard™ Configuration Tool from www.balluff.com. The Dashboard™ Configuration Tool uses the USB port to communicate to the gateway USB port (Type B). To enable communication:

4. Connect the gateway USB port (Type B, Female) to the PC by using a standard USB A/B cable.
5. On the host computer, set COM port parameters to: 9600 baud, 8 data bits, 1 stop bit, no parity and no handshaking.
6. Run the Balluff Dashboard™ Configuration Tool.

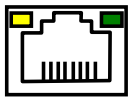
3 LED INDICATORS

3.1 FRONT PANEL LEDS

3.1.1 RS232 Models

LED Name	LED Color		LED Description
POWER		**GREEN**	The POWER LED is ON whenever power is applied to the Gateway.
Subnet16™ **BUS**		**AMBER**	The BUS LED will flash ON and OFF to indicate that data is being transmitted between the Gateway and one or more RFID controllers on the Subnet16™ network.
RS232		**AMBER**	The RS232 LED will flash ON and OFF to indicate that data is being transmitted between the host and the Gateway.
ERROR		**RED**	The ERROR LED is solid red when a Gateway configuration error has occurred, i.e. an invalid or unrecognized command. This LED will be cleared when a valid command is sent.

3.1.2 INDUSTRIAL and TCP/IP Ethernet Models

LED Name	LED Color		LED Description
POWER		**GREEN**	The POWER LED is ON whenever power is applied to the Gateway.
Subnet16™ **BUS**		**AMBER**	The BUS LED will flash ON and OFF to indicate that data is being transmitted between the Gateway and one or more RFID controllers on the Subnet16™ network.
ETHERNET		**AMBER**	The ETHERNET LED will flash ON and OFF to indicate that data is being transmitted between the host and the Gateway.
ERROR		**RED**	The ERROR LED is solid red when a Gateway configuration error has occurred, i.e. an invalid or unrecognized command. This LED will be cleared when a valid command is sent.
LINK		**AMBER**	The LINK LED on the left is the 10/100 Indicator LED, which will turn ON whenever an Ethernet link is established and will remain ON for the duration of the connection.
TRAFFIC		**GREEN**	The TRAFFIC LED on the right is the Ethernet Data LED, which will flash ON and OFF when Ethernet traffic is detected by the Gateway (regardless of origin and destination).

3.1.3 DEVICENET Models

LED Name	LED Color		LED Description
POWER		**GREEN**	The POWER LED is ON whenever power is applied to the Gateway.
Subnet16™ **BUS**		**AMBER**	The BUS LED will flash ON and OFF to indicate that data is being transmitted between the Gateway and one or more RFID controllers on the Subnet16™ network.
DEVICENET		**GREEN/RED**	**SOLID GREEN:** Gateway is operational, online AND a connection is established. **FLASHING GREEN:** Gateway is operational and online, but with no established I/O connections (idle on the network) or Gateway is online and needs commissioning. **FLASHING RED:** Recoverable fault detected and/or the Gateway's I/O connection timed out. **SOLID RED:** Unrecoverable fault detected (for example, a duplicate node address was encountered rendering Gateway unable to communicate).
ERROR		**RED**	The ERROR LED is solid red when a Gateway configuration error has occurred, i.e. an invalid or unrecognized command. This LED will be cleared when a valid command is sent.

3.1.4 PROFIBUS Models

LED Name	LED Color		LED Description
POWER		**GREEN**	The POWER LED is ON whenever power is applied to the Gateway.
Subnet16™ **BUS**		**AMBER**	The BUS LED will flash ON and OFF to indicate that data is being transmitted between the Gateway and one or more RFID controllers on the Subnet16™ network.
PROFIBUS OP MODE		**GREEN/RED**	**SOLID GREEN:** on-line, data exchange **FLASHING GREEN:** on-line, but idle. **FLASHING RED (1 FLASH):** parametrization error **FLASHING RED (2 FLASHES):** Profibus configuration error
PROFIBUS STATUS		**GREEN/RED**	**SOLID GREEN:** initialized. **FLASHING GREEN:** initialized, diagnostic event(s) present. **SOLID RED:** exception error
ERROR		**RED**	The ERROR LED is solid red when a Gateway configuration error has occurred, i.e. an invalid or unrecognized command. This LED will be cleared when a valid command is sent.

4 CONFIGURATION METHODS

4.1 NODE ID CONFIGURATION USING CONFIGURATION TAGS

Only RS485-based Processor units can be connected to a Gateway's Subnet network and each must be assigned a unique Node ID value between 1 and 16.

When an Processor unit is connected to the Gateway's Subnet network, the Gateway will query the new controller to obtain certain configuration values (specifically the Node ID number). If the Gateway does not detect a Node ID conflict, it will "allow" the Processor unit onto the Subnet network.

By using the ***BIS M Configuration Tag*** that is included with each RS485-based BIS M-41x, BIS M-62x Processor units, or the ***BIS U Configuration Tag*** that is included with each RS485-based BIS U-62x Processor units, the Node ID value can be dynamically assigned by the Gateway or can be manually assigned by the user.



Figure 17 – BIS M-62_ and BIS U-62_ Configuration Tags

For the Gateway to dynamically assign a Node ID value to a controller, the controller must first be initialized with the Node ID value of zero. This is the equivalent of having no Node ID assigned.

NOTE

All Balluff RS485-based controllers ship with their Node ID value set to 0.

When a powered controller (that is set to Node ID 0) is connected to the Subnet, it will not initially be recognized by the Gateway until the Configuration Tag is placed in the antenna's RF field. After a few seconds the controller will display its new assigned Node ID value in

binary code from right to left or (top to bottom) using the five amber Node LEDs on the controller.
When dynamically assigning a Node ID value for a new controller, the Gateway will either assign the next available Node ID value or the value that the Gateway recognizes as offline or "*missing*" – that is, a Node ID value that previously existed, but has since disappeared from the network.

Because the Gateway stores a backup of each Subnet Node's configuration, should an Processor unit ever fail, a replacement controller can be installed quickly and easily. The new controller will be automatically assigned the same Node ID value and configuration as the replaced controller, provided the Configuration Tag is introduced to the antenna field after startup and then removed.



NOTE

Avoid that the configuration tag is simultaneously read by more than one controller, especially for UHF controllers.

4.2 GATEWAY AND SUBNET NODE NAMING

The Gateway can store a 64-byte ASCII string for each of the 16 Subnet Nodes and one 64-byte ASCII string for the Gateway itself. These text strings can be used to assign logical or "*user friendly*" names to the Gateway and its Subnet Nodes.

For example, you could assign the Gateway a logical name such as "*PRODUCTION LINE 1*" and then name the controller connected to Subnet Node 01 "*PRODUCTION STATION 1*". The controller at Subnet Node 02 could then be named "*PRODUCTION STATION 2*" (and so forth).

Gateway and Node names can be retrieved and edited by issuing specific commands to the Gateway, see the CBx Protocol Manual for details.

Gateway and Node naming can also be accomplished through the *Balluff Dashboard*™ software utility, see the Balluff Dashboard™ Manual for more details.

4.3 CONFIGURATION TOOLS

Balluff offers the following powerful RFID configuration utilities for Microsoft Windows 2000, XP, Vista and 7 systems:

- **Balluff Dashboard™**
- **C-Macro Builder™**

These configuration tools can be downloaded from the Balluff website: www.balluff.com

4.3.1 Configuration Using Balluff Dashboard™

The ***Balluff Dashboard™ Configuration Tool*** is a software application that allows users to view, modify, save and update the configuration settings of their processor units. Follow the instructions below to operate the *Balluff Dashboard Configuration Tool* and to set the device's configuration.

1. Install the Gateway as described in the relevant sub-paragraph in 2.5.
2. For DeviceNet and Profibus models, install the Bis4D_com drive to your PC as described in par. 4.4.
3. Connect the Gateway to your PC, power up and wait for the boot procedure to finish.
4. Run the *Balluff Dashboard*™.
5. From the Connection screen, choose your Gateway from the list.

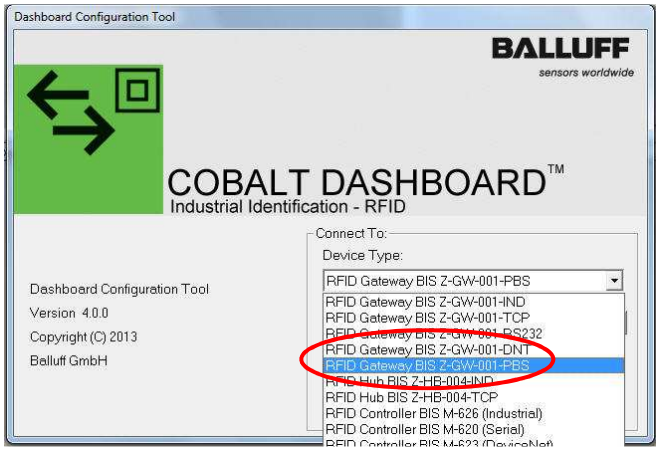


Figure 18 - Balluff Dashboard™Gateway Model Selection

6. Choose the appropriate COM port and Baudrate (or IP Address for Ethernet models); then click "Connect".
   - 115200 Baud is the default serial baud rate for USB connection
   - 9600 Baud is the default serial baud rate for RS232

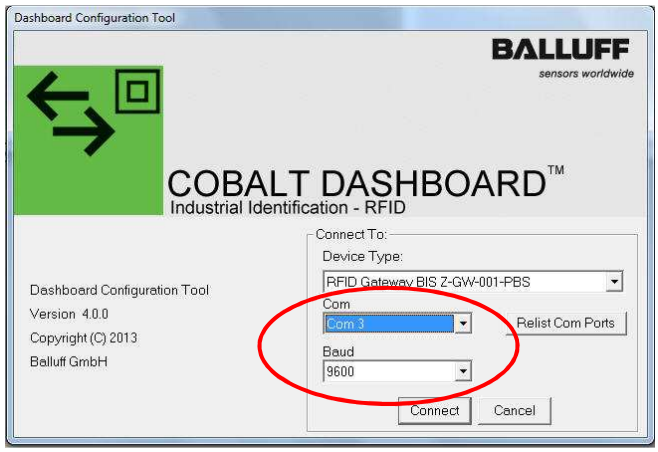


Figure 19 - Balluff Dashboard™ COM Port and Baudrate Selection

The Dashboard should send some commands to retrieve device and configuration information from the device. If communications are set up correctly, the device configuration area within the Balluff Dashboard™ should now look something like this:

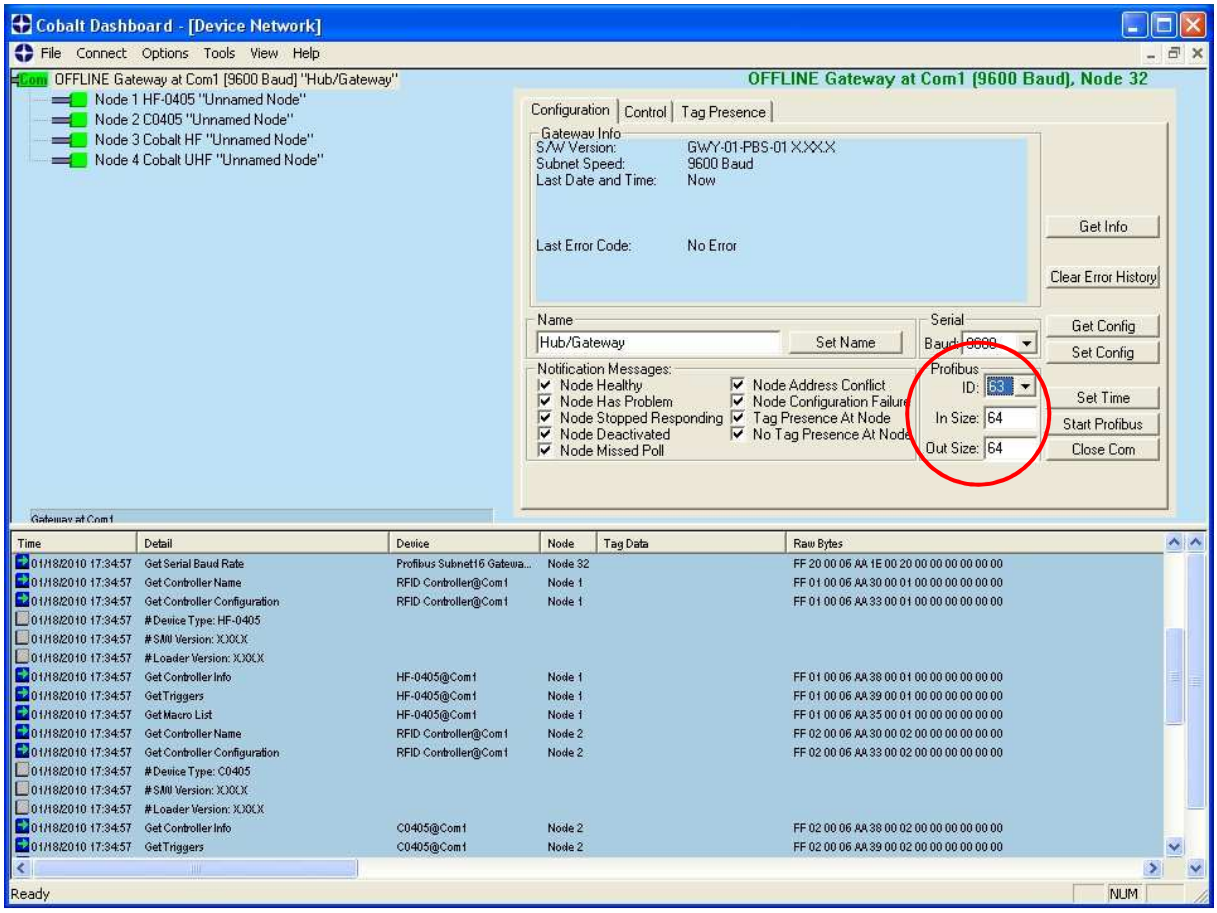


Figure 20 - Balluff Dashboard™ Gateway Profibus Configuration

In this example the Gateway is a Profibus model. The default Profibus ID is 63, and the default Input and Output Buffer sizes are 64 bytes each.

To change the Profibus ID of the device, and/or the Input Buffer/Output Buffer sizes, enter the desired values in the appropriate boxes in the configuration tab, and click “Set Config”.

For Profibus, the new ID and buffer sizes will not take effect until the device is reset one more time, or the power is cycled. Turn the unit off and back on to use the new ID and buffer size settings. For other settings (such as Tag Type, continuous read parameters, etc), the unit does not need to be reset again or have the power cycled.

See the Balluff Dashboard™ User's Manual for more configuration details.

4.3.2 Creating and Using RFID Macros with C-Macro Builder™

What are RFID Command Macros?

RFID Command Macros are a powerful feature of Balluff processor unit. Macros are simple programs that direct a controller to execute multiple pre-programmed instructions.

Because macros reside within the controller's internal memory, they can be programmed to instruct the controller to automatically read and/or write a specified set of data to an RFID tag without the controller ever having to receive a command from the host. In fact, the controllers do not even require a connection to a host in order to execute macros.

Each macro can contain up to 255 bytes of data and each supported controller can store up to eight macros at a time. Though they are stored locally on the controller, macros are also backed up in the Gateway's flash memory as well.

Why use macros?

The power of macros is in distributed intelligence, the reduction in network bus traffic and the ability to accelerate routine decision making at the point of data collection.

What can macros do?

In addition to the automated reading and writing of data, macro capabilities include:

- The ability to write time stamps to RFID tags
- The ability to filter command responses to only those of interest to the host (such as when an error occurs or when a tag has arrived in the RF field)
- The ability to harness powerful logic and triggering capabilities such as; read, write, start/stop continuous read, data compare, branch, transmit custom string, and set outputs.

What is a macro trigger?

Macros are initiated by "triggers." Triggers can be configured in numerous ways. A simple command from the host, such as "execute macro number three" can be considered a trigger. Triggers can be configured, for example, to activate a macro when a tag enters or leaves a controller's RF field.

Balluff Processor units can store up to eight separate triggers in addition to the eight macros they can also house. Any trigger can activate any of the eight stored macros.

How are macros created?

Macros are created using the powerful, yet simple, C-Macro Builder™ utility from Balluff. The easy to use GUI allows the user to create powerful RFID macro programs quickly and easily. When used with Balluff Balluff Dashboard™ Configuration Tool, users can effortlessly download, erase, and manage their macros and triggers, as well as set the operational configurations of their Processor units and Subnet16™ Gateways.

Which communication interfaces support the use of macros?

Macros are supported on the following processor units: Ethernet, Profibus, Profinet, DeviceNet, RS232 and USB interfaces.

What happens to existing Macros if a controller must be replaced?

When using a Subnet16™ Gateway, users do not need to worry. Macros and triggers normally residing in an Processor unit's flash memory are always backed up in the Gateway's flash memory as well. Therefore, if a controller should ever require replacement, all existing macro and trigger settings are automatically exported from the Gateway to the new Processor unit.

In short, when an Processor unit is initially connected to the Gateway, macro and trigger data from the controller's flash memory is compared to the macro and trigger data backed up in the Gateway from the previous Processor unit. If the data does not match that which is stored on the Gateway, the controller's flash memory will be overwritten with the backed up data stored in the Gateway's flash memory.

How can I learn more about the Dashboard and C-Macro Builder?

More information regarding macros, triggers, uploading, downloading, configuring and monitoring Balluff RFID equipment is available in the respective User's Manuals for these products, which are available on the Balluff website at:

www.balluff.com

C-Macro Builder™ is an easy to use GUI-driven utility for Windows that allows users to create powerful RFID command macro programs.

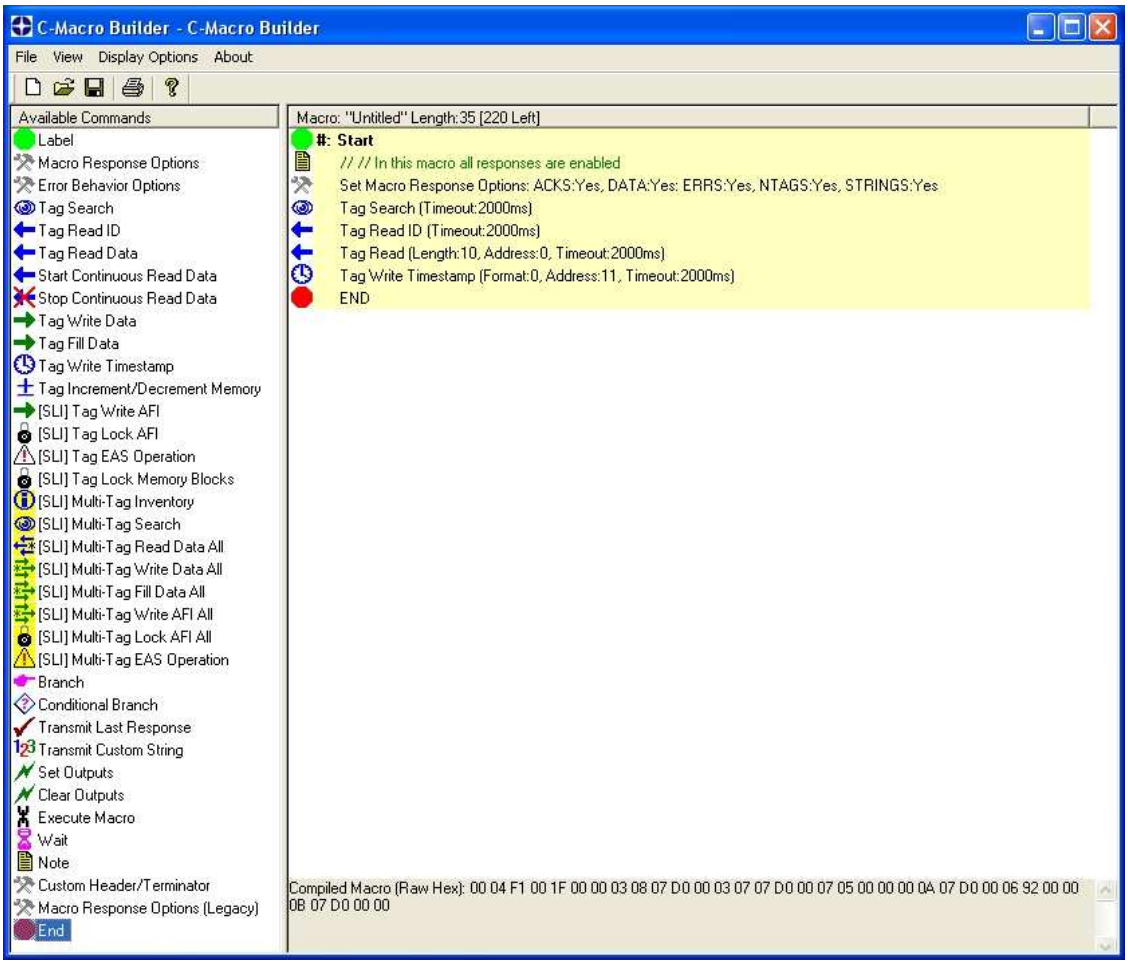


Figure 21 - C-Macro Builder™

NOTE

For specific information regarding the configuration and use of either of these utilities, please see the accompanying documentation included when downloading each software application.

4.4 USB DRIVER INSTALLATION

This paragraph contains instructions for installing the Windows 2000/XP USB dri ver for Balluff devices, and in particular for DeviceNet and Profibus Gateways which use the USB interface for device configuration. Complete the following steps before installing the Bis4D_com drive.

1. Download the Bis4D_com drive package from www.balluff.com.

2. Extract the driver files to a separate folder on your host computers' Desktop.

3. Connect the USB interface cable (and power supply, if applicable) as described in par. 2.5.

4. Apply power to your RFID device. Windows should detect the new hardware and start the Found New Hardware Wizard. If it does not, run the Add Hardware applet in Windows' Control Panel.



Figure 22 - Found New Hardware Wizard – Do Not Connect to Windows Update

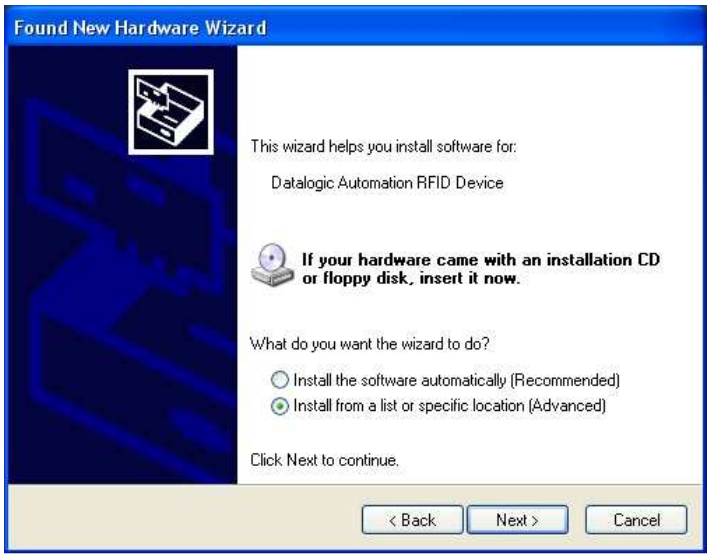


Figure 23 - Found New Hardware Wizard – Install From List

5. Select “Install from a list or specific location” and then click Next.
6. Check the box labeled: “Include this location in the search” and then click the Browse button.

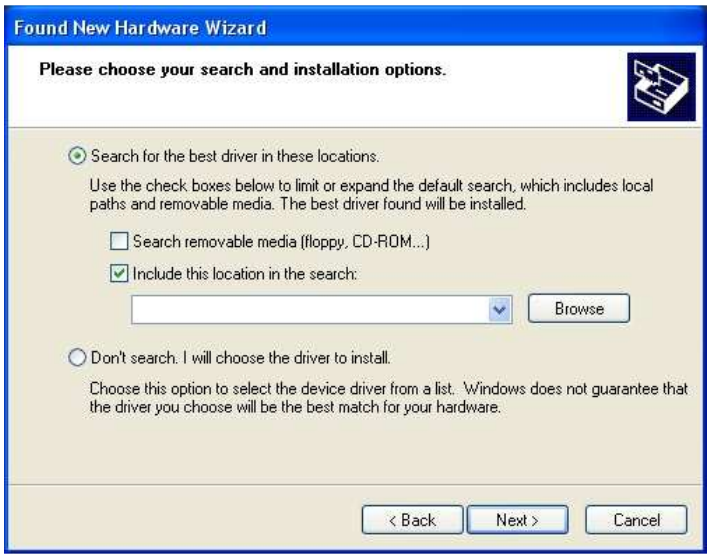


Figure 24 - Found New Hardware Wizard – Include this Location

7. Browse to the folder containing the extracted Bis4D_com drive files and then click OK.



Figure 25 - Found New Hardware Wizard - Browse for Folder

8. Click Next to install the USB driver.

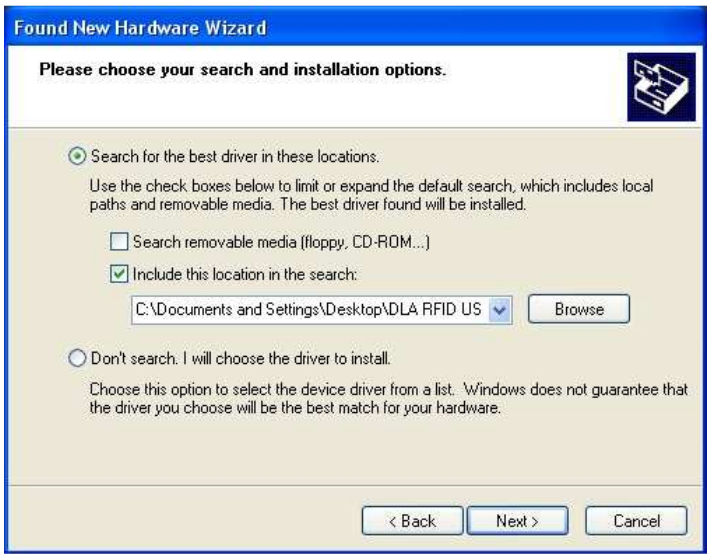


Figure 26 - Found New Hardware Wizard – Ready to Install

Please wait while the *Found New Hardware Wizard* installs the USB driver.



Figure 27 - Installing the USB Driver

9. After the USB driver has been installed, click Finish.



Figure 28 - USB Driver Installation Complete

NOTE

Immediately after you click Finish, the Found New Hardware Wizard will close and then automatically restart, prompting you to repeat the USB driver installation. This occurs because Windows requires a second trip through the installation routine to install and configure a virtual COM port for use by the RFID USB device.

10. Repeat Steps 5-9 to install the virtual COM port and complete the installation of the Bis4D_com drive.



Figure 29 - Virtual COM Port USB Driver Installation Complete

5 ETHERNET/IP INTERFACE



NOTE

For BIS Z-GW-001-IND models.

- Users of the *Balluff Dashboard™ Configuration Tool* should exit the application before attempting communications between the Industrial Gateway and an EtherNet/IP host Programmable Logic Controller (PLC).
- When installing the Gateway for communication over EtherNet/IP, the ODVA Guidelines for EtherNet/IP Media System installation should be followed (refer to www.odva.org, ODVA **PUB00148R0** (Pub 148), EtherNet/IP Media Planning and Installation Manual, 2006 ODVA).
- Follow ODVA recommendations for switching and wiring Ethernet/IP.
- If the Ethernet/IP network enables I/O Messaging for remote I/O, etc., or if other UDP traffic is present, then the Gateway must be protected by a switch that incorporates IGMP Snooping or a VLAN.

The *BIS Z-GW-001-IND* is designed to support many common Industrial Ethernet protocols and can be implemented in a wide variety of existing host / PLC applications. One such popular Ethernet protocol is ***Ethernet/IP*** (**EIP**).

This chapter focuses on the process of setting up the Subnet16™ Industrial Gateway to communicate (via Ethernet/IP) with a ControlLogix Programmable Logic Controller (PLC).

Also in this chapter are descriptions of the Balluff ***HTTP Server*** and ***OnDemand Utilities***, as well as systematic instructions to help configure the Industrial Gateway for Ethernet/IP environments.



NOTE

This manual assumes that users are already familiar with Ethernet/IP, industrial Ethernet communications protocols and programmable logic controller technologies. For specific information regarding the protocol used by your particular RFID application, please refer to the appropriate documentation from your host / PLC program provider.

5.1 ETHERNET/IP CONFIGURATION OVERVIEW

Based upon on the standard TCP/IP protocol suite, EtherNet/IP is a high-level application layer protocol for industrial automation applications that uses traditional Ethernet hardware and software to define an application layer protocol that structures the task of configuring, accessing and controlling industrial automation devices.

Ethernet/IP classifies Ethernet nodes as predefined device types with specific behaviors. The set of device types and the EIP application layer protocol is based on the Common Industrial Protocol (CIP) layer used in ControlNet. Building on these two widely used protocol suites, Ethernet/IP provides a seamlessly integrated system from the RFID Subnet network to the Host and enterprise networks.

The Gateway is designed to communicate as an EtherNet/IP client device, which will receive and distribute RFID commands issued by the host / PLC (acting as EtherNet/IP Server).

Paragraphs 5.3 through 5.7 contain instructions that will help you accomplish the following:

- Assign the Gateway an IP address via *HTTP Server*
- Configure the Subnet Nodes via *OnDemand Utilities*
- Create "*Controller Tags*" in the PLC
- Verify PLC and Cobalt Subnet Node connectivity

5.2 HTTP SERVER & ONDEMAND PLC SUPPORT

Below is a partial list of the programmable logic controllers that are supported by the Balluff *HTTP Server* and *OnDemand Utilities*:

- ControlLogix – OnDemand supports all current versions
- RA's PLC5E releases:
- Series C, Revision N.1
- Series D, Revision E.1
- Series E, Revision D.1
- PLC5 "Sidecar" Module Series B, Revision A with EIP support
- SLC5/05 releases:
- Series A with firmware revision OS501, FRN5
- All Series B and Series C PLC Controllers

5.3 HTTP SERVER AND ONDEMAND UTILITIES

Embedded in the *BIS Z-GW-001-IND* is an ***HTTP Server,*** which provides a Website-like interface and a suite of configuration tools.

Through the use of the *HTTP Server*, users can access, modify and save changes to the unit's Industrial Ethernet configuration, IP address, and *OnDemand* mode settings.

The *OnDemand Utilities* will be used later in this chapter to link the BIS Z-GW-001-IND to specific *Controller Tags* as defined in Rockwell Automation's (*RA*) ControlLogix PLC.



CAUTION

Disable any firewall services affecting or running locally on the host computer. Firewalls can potentially block communications between the Cobalt and the host and/or PLC.



NOTE

*In ControlLogix, a "**Controller Tag**" is a small block of internal memory that is used to hold outgoing (command) and incoming (response) data. Within each controller tag, information is stored in two-byte segments, known as registers or "words."*

OnDemand is the Balluff approach to adding *Change of State* messaging to ControlLogix and legacy support for *RA PLC5E* and *RA SCL5/05* programmable logic controllers.

5.4 IP CONFIGURATION VIA HTTP SERVER

To configure the Gateway for Ethernet communications, begin by assigning it a locally compatible IP address.

Through a standard Web browser, you can utilize the BIS Z-GW-001-IND *HTTP Server* to access an embedded suite of configuration tools, called the "***OnDemand Utilities***." Among its features is the ability to modify and save changes to the Gateway's IP address, which is stored internally on the BIS Z-GW-001-IND.

Subnet16™ Industrial Ethernet Gateway - Default IP Address:
192.168.253.110

Setting the Gateway IP Address

To set the Gateway's IP address using the *HTTP Server,* follow the steps below:

1. Open a Web browser on the PC.
2. In the URL address field, enter the Gateway's IP address (*192.168.253.110 = factory default*).
3. Press ***ENTER***.

The *HTTP Server - Main Page* will be displayed.

HTTP Server – Main Page

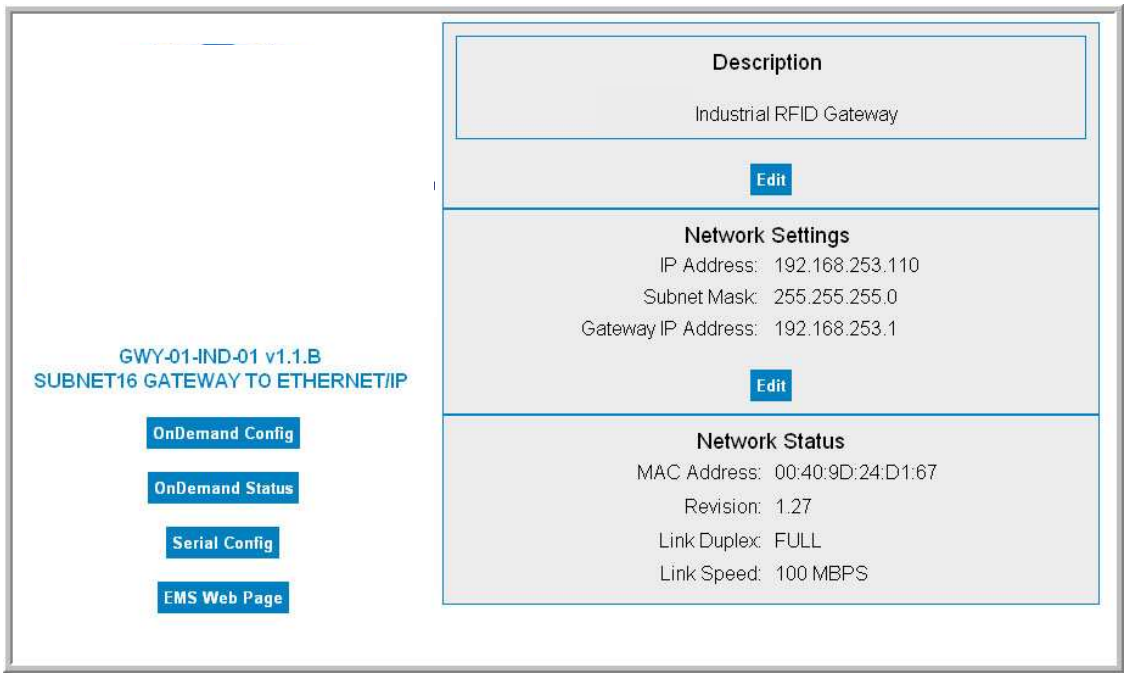
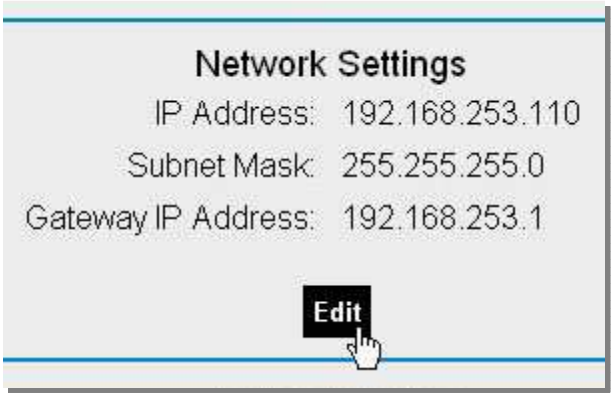


Figure 30 - The HTTP Server - Main Page

The ***HTTP Server - Main Page*** lists the IP address and network settings currently stored on the Gateway BIS Z-GW-001-IND.

4. Click the button labeled "***EDIT***", located below "***Network Settings.***"



The IP Configuration Page will be displayed.

IP Configuration Page

The *IP Configuration Page* is used to modify and save changes to the IP Address, Subnet Mask and (Network) Gateway IP Address.



Figure 31 - The IP Configuration Page

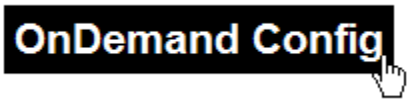
5. In the fields provided, enter your new IP configuration values for the Gateway.
6. Click the **"*Save Settings***" button to store the new IP configuration. The Gateway will completely reset and your IP changes will be implemented.
7. After the Gateway has restarted, verify the new IP configuration by opening a Web browser and manually entering the Gateway's new IP address in the URL field. If successful, you should arrive back at the *HTTP Server – Main Page*.

5.5 ONDEMAND CONFIGURATION FOR ETHERNET/IP

Now that you have configured the Gateway's IP address, use the *HTTP Server* to access the Gateway's ***OnDemand Configuration Page***, which is allows configuration of each Subnet Node. The Gateway supports the connection of up to 16 individual processor units. Through the use of the *OnDemand Utilities,* the Gateway's Subnet Nodes will be linked to specific "Controller Tags" as defined in the ControlLogix PLC.

To configure the Gateway's Subnet Nodes, follow the steps below:

1. Open a Web browser on the host and enter the Gateway's new IP address in the URL field. The *HTTP Server – Main Page* will be displayed.
2. At the *HTTP Server – Main Page*, click the button labeled "***OnDemand Config***."



The *OnDemand Configuration Page* will be displayed.

OnDemand Configuration Page

The ***OnDemand Configuration Page*** allows you to modify the settings for each Subnet Node.

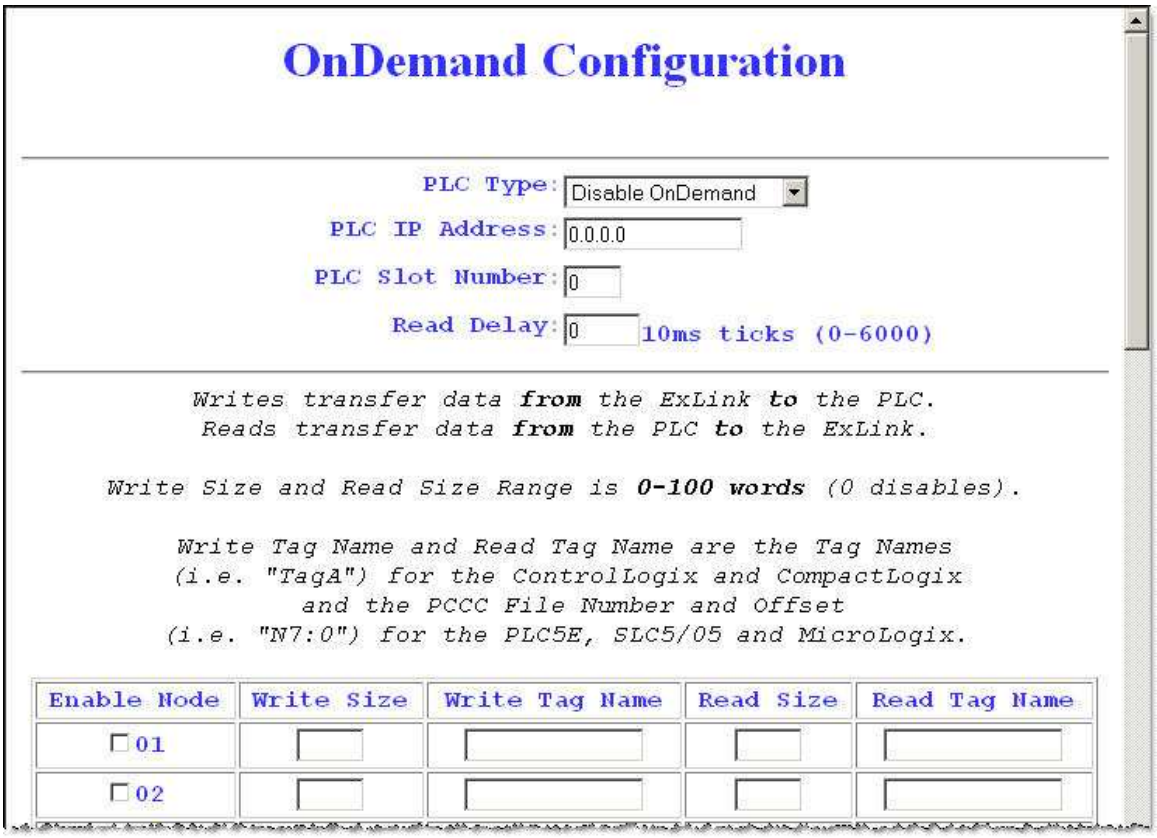


Figure 32 - The OnDemand Configuration Page

3. In the upper portion of the *OnDemand Configuration Page*, select a ***PLC Type*** from the drop-down menu.

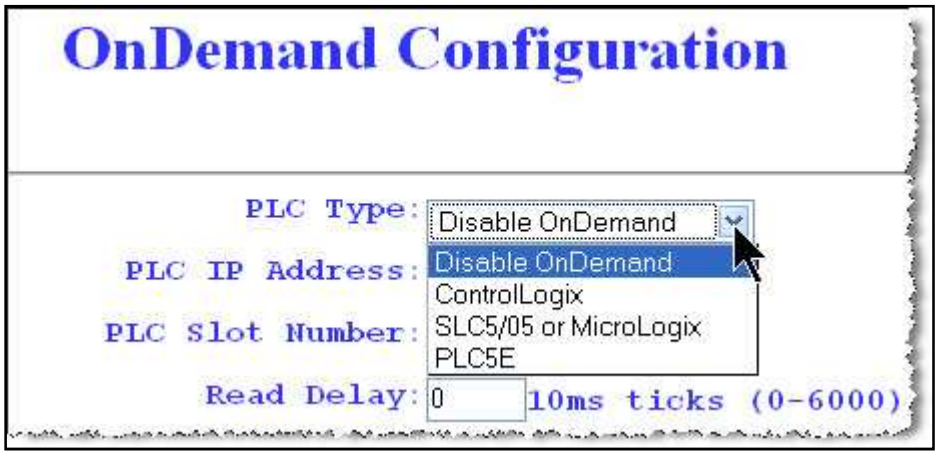


Figure 33 - The OnDemand Configuration Page

4. Enter the PLC's IP address.
5. For the ***PLC Slot Number,*** enter a value between 0 and 255. The PLC Slot Number indicates the location in your PLC rack where the controller module is installed (normally slot 0 for ControlLogix).

6. In the ***Read Delay*** field, enter a value between 0 and 6000. This number specifies (in 10 ms "ticks") how frequently the Gateway will poll the PLC for the presence of new data. (*Note: a value of 6000 = 60 seconds; zero = disable*).

7. In the column labeled "***Enable Node,***" place a check in the box for **Node 01**.



8. **Write Size:** Enter a value between 1 and 100 (or 0 to disable) for the ***Write Size.*** The Write Size represents the maximum number of 2-byte "words" that the Gateway will attempt to write to PLC memory during a single write cycle. *(Note: to accommodate message handshaking overhead, the actual data size required by the PLC is three words larger than the value specified in this field).*

9. **Write Tag Name:** For *ControlLogix* systems, specify a ***Write Tag Name*** that is 40 characters or less (for example *EMS_WRITE1,* for Node 01). The Write Tag Name is a user defined description or title for the area of memory in the PLC where host-bound data will be written for a particular Subnet Node. (*Note: the Write Tag Name is not to be confused with writing to a data carrier, which is often referred to as "writing to a tag").*

   OR

   Write Tag Name: For *PLC5E, SLC5/05 and MicroLogix* systems, enter the ***PCCC File Number and Offset*** (for example *N7:0*) in the *Write Tag Name* field. Together

these values identify the location in the PLC's Status File where host-bound data will be written for the Subnet Node.

10. **Read Size:** Enter a value between 1 and 100 (or 0 to disable) for the ***Read Size.*** The Read Size represents the maximum number of 2-byte "words" that the Gateway will attempt to retrieve from PLC memory during a single read cycle. *(Note: to accommodate message handshaking overhead, the actual data size required by the PLC is three words larger than the value specified in this field).*

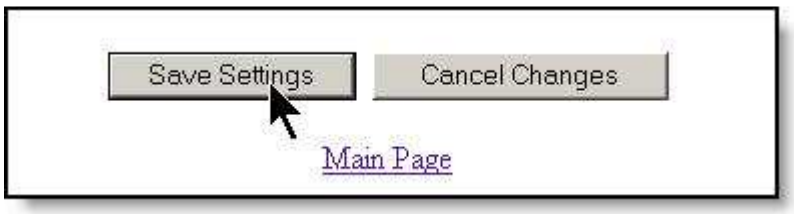
11. **Read Tag Name:** For *ControlLogix* systems, specify a ***Read Tag Name*** that is 40 characters or less (for example *EMS_READ1,* for Node 01). The Read Tag Name is a user defined description or title for the area of memory in the PLC from which the Gateway will retrieve data bound for a particular Subnet Node.

    OR

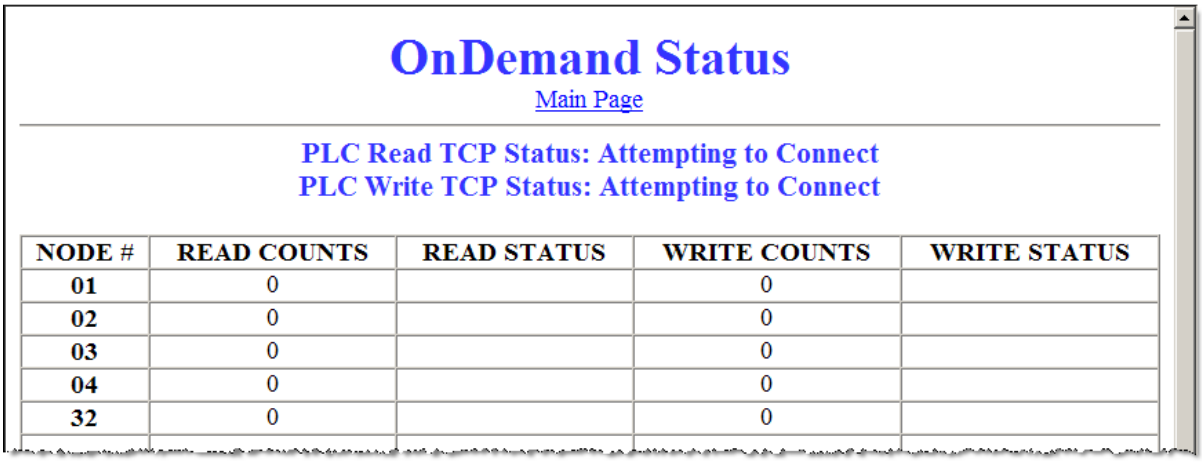
    Read Tag Name: For PLC5E, SLC5/05 and MicroLogix systems enter the ***PCCC File Number and Offset*** in the Read Tag Name field. Together these values indicate the location in the PLC's Status File where the Gateway will retrieve data destined for the Subnet Node.

12. After entering the proper information for Node 01, repeat steps 7 through 11 and configure Nodes 02 through 16 and 32. Node 32 (the "*Gateway Node*") is used to hold data specifically intended for the Gateway.

13. After entering the proper information for Nodes 01 - 16 and 32, click the ***Save Settings*** button located at the bottom of the page.



The *OnDemand Status Page* will be displayed.



OnDemand Status

Main Page

PLC Read TCP Status: Attempting to Connect
PLC Write TCP Status: Attempting to Connect

NODE #	READ COUNTS	READ STATUS	WRITE COUNTS	WRITE STATUS
01	0		0	
02	0		0	
03	0		0	
04	0		0	
32	0		0	

14. At the *OnDemand Status Page*, click the link labeled "***Main Page***" to return to the *HTTP Server – Main Page*.

5.6 CONFIGURING PLC CONTROLLER TAGS

After you have configured each Subnet Node via *OnDemand Node Configuration*, open your PLC program (i.e. RSLogix 5000) and, if you have not already done so, define two ***Controller Tags*** (a ***Write Tag*** and a ***Read Tag***) for each Subnet Node that has been configured.

Controller Tag Naming

All Controller Tags need to be assigned a name and size. Be sure to use the same ***Write Tag Name*** and ***Read Tag Name*** that you specified in the *OnDemand Node Configuration* (i.e., EMS_WRITE1 and EMS_READ1).

Controller Tag Size

Due to handshaking overhead, *Controller Tags* must have the size capacity to store an integer array equal to your previously specified ***Write/Read Size + three words***.
So for example, if the *Read Size* you specified earlier was 100 words, the corresponding *Read Tag* in the PLC must be able to store an array of 103 integers.



- The **Write Tag** holds messages and response data bound for the host or PLC. This data is generated either by the Gateway or is passed through the Gateway from an RFID controller. (Note: the RFID controller is linked to the proper Write Tag via *OnDemand Node Configuration)*.
- The **Read Tag** holds RFID commands and instructions intended for the Gateway or a specified Subnet Node. Instructions will be routed to the Node for which the Read Tag has been linked via *OnDemand Node Configuration*.

NOTE

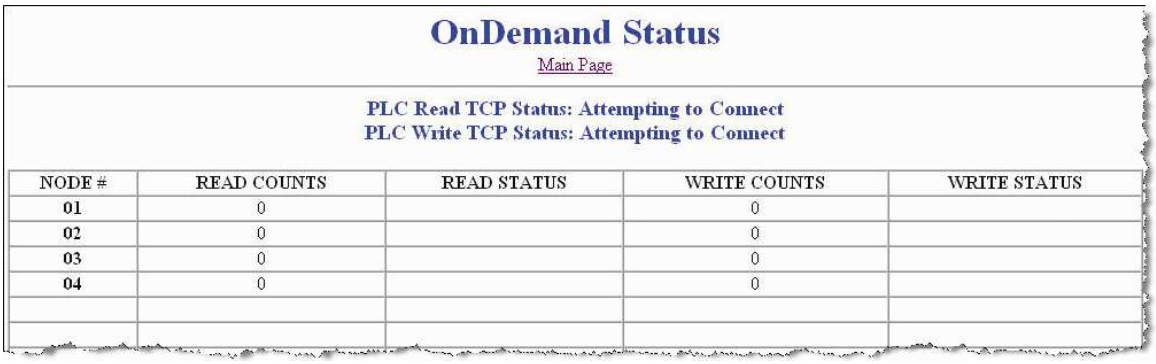
The Cobalt should already be linked to the proper Write Tag and Read Tag via the OnDemand Utilities - OnDemand Configuration Page).

After creating and defining a *Write Tags* and a *Read Tags* for each Subnet Node, return to the Gateway's *HTTP Server – Main Page* to continue.

5.7 CHECKING ONDEMAND STATUS

Now that you have configured the Gateway's Subnet Nodes and defined corresponding *Write* and *Read Tags* for each in the PLC, the last step is to check the communication status between the Gateway's Subnet Nodes and the PLC.

Return to the Gateway's *HTTP Server - Main Page* and click the link labeled "***OnDemand Status***." The *OnDemand Status Page* will be displayed.



OnDemand Status

Main Page

PLC Read TCP Status: Attempting to Connect
PLC Write TCP Status: Attempting to Connect

NODE #	READ COUNTS	READ STATUS	WRITE COUNTS	WRITE STATUS
01	0		0	
02	0		0	
03	0		0	
04	0		0	

Figure 34 - The OnDemand Status Page

The OnDemand Status Page provides statistical information regarding the connection status between the PLC and each configured Subnet Node. This information can be used to verify that read and write connections between each Subnet Node and the PLC have been established successfully.

- ***Read Counts***: this value indicates the number of times the Subnet Node has checked the PLC for new data.
- ***Write Counts***: this value indicates the number of times the Subnet Node has provided data to the PLC.

NOTE

Under Ethernet/IP, the host (and/or PLC) acts as the server. However, additional messaging instructions are not required on the part of the host because the Gateway will automatically poll the Read Tag in the PLC at the interval specified by the ***Read Delay*** *value set via the OnDemand Configuration Utility.*

There is no delay parameter when writing data to the PLC, as the Gateway delivers all PLC-bound data immediately after it is generated.

If you configured a low Read Delay value, the Read Counts on the OnDemand Status Page will accumulate rapidly. This occurs because a low Read Delay value instructs the Gateway to poll the PLC for new data more frequently.

CAUTION

If the Gateway BIS Z-GW-001-IND *and PLC do not successfully establish a connection, cycle power to the Gateway and verify that Ethernet/IP services are running properly on the PLC. If that does not resolve the issue, restart Ethernet/IP services on the PLC and the 1756-ENBT module.*

5.8 VERIFYING DATA EXCHANGE WITH RSLOGIX 5000

At this point, communication between the PLC and the Gateway should be properly configured and a connection established. You can verify the exchange of information between devices using RSLogix 5000.

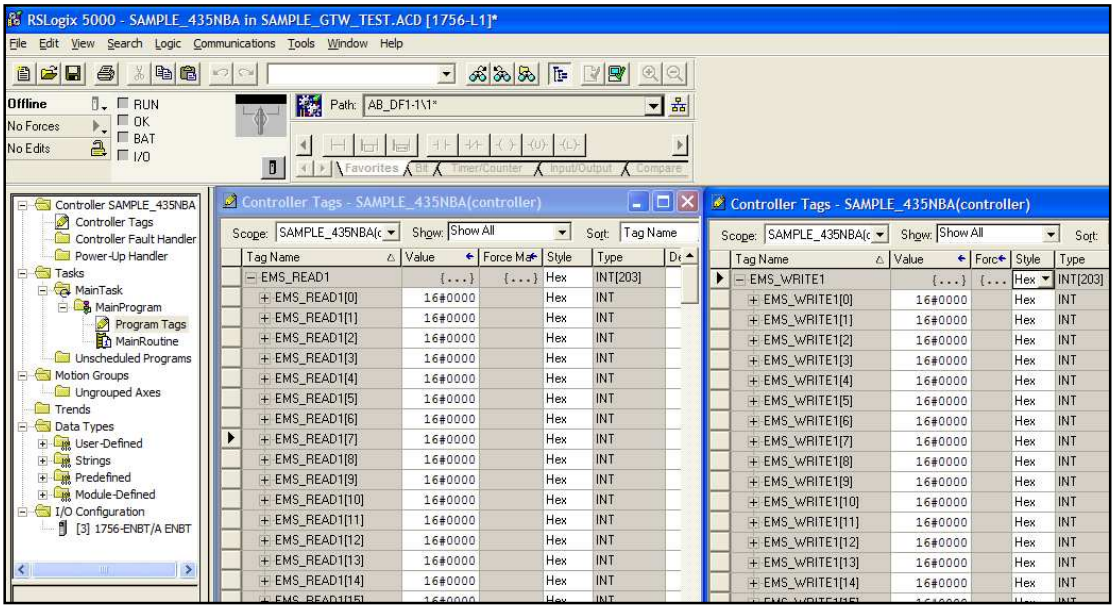


Figure 35 - RSLogix 5000

5.8.1 Ethernet/IP Handshaking

To ensure that messages to and from the BIS Z-GW-001-IND and Subnet16™ processor units are properly delivered and received, a handshaking mechanism has been implemented that uses a pair of dedicated words in the exchange. The first two words in each *Controller Tag* are dedicated to handshaking.

When new information is generated, the producing device (*Data Producer*) will increments a counter in one of the *Controller Tags*. After identifying the new data, the consuming device (*Data Consumer*) will copy that same counter value to a different Controller Tag location, which lets the Data Producer know that the information has been processed by the Data Consumer.

WRITE TAG (where responses are written by the Gateway)
EMS_Write1 [0] = (2) the Gateway copies counter here to ACK
EMS_Write1 [1] = (3) the Gateway increments this counter to signal response available
EMS_Write1 [2] = Data Size
EMS_Write1 [3-102] = Data
READ TAG (where commands are retrieved by the Gateway)
EMS_Read1 [0] = (4) PLC copies the counter here to ACK the response
EMS_Read1 [1] = (1) PLC increments this counter after writing a command
EMS_Read1 [2] = Data Size
EMS_Read1 [3-102] = Data

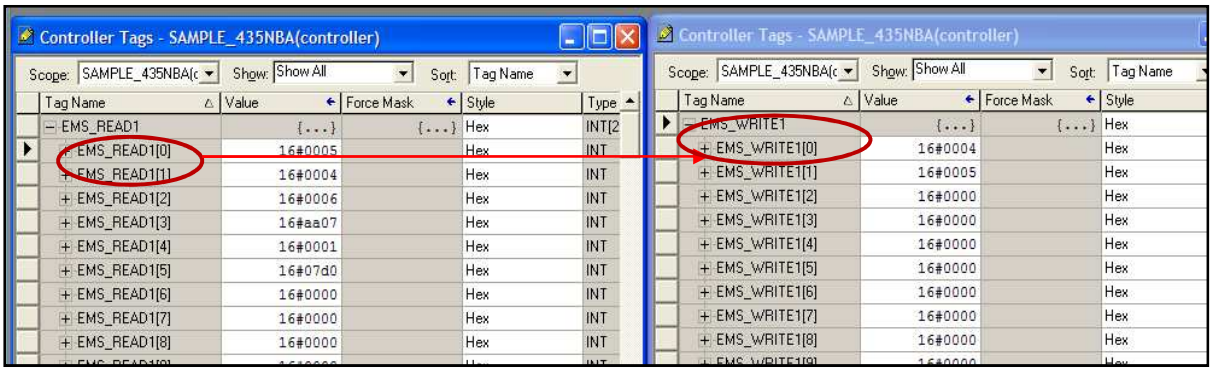
5.8.2 Ethernet/IP Handshaking Example

In the example below, **EMS_READ1** is the name of the Read Tag for Node 1 and **EMS_WRITE1** is the name of the Write Tag for Node 1.

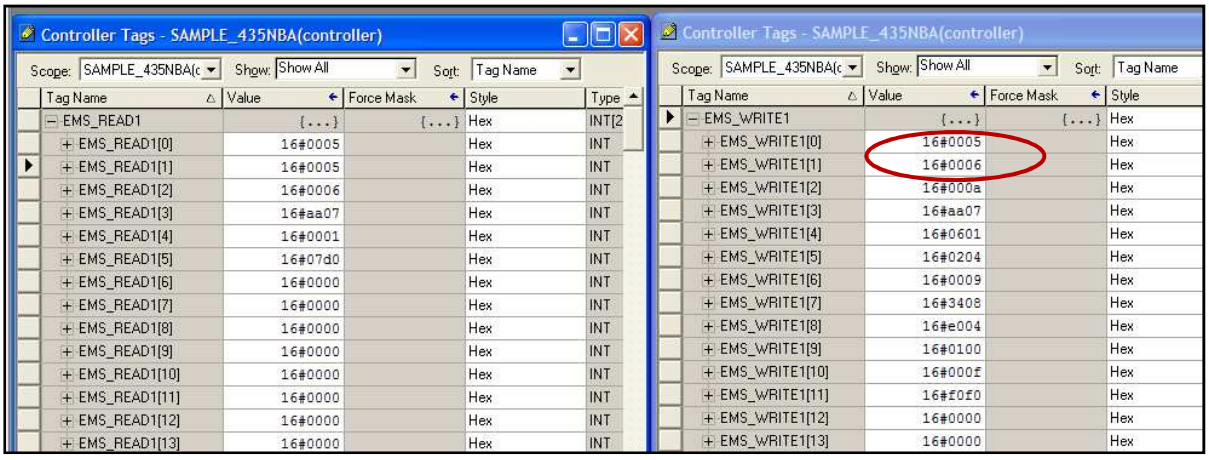
NOTE

***[0]** indicates the first word, **[1]** indicates the second word in a Controller Tag.*

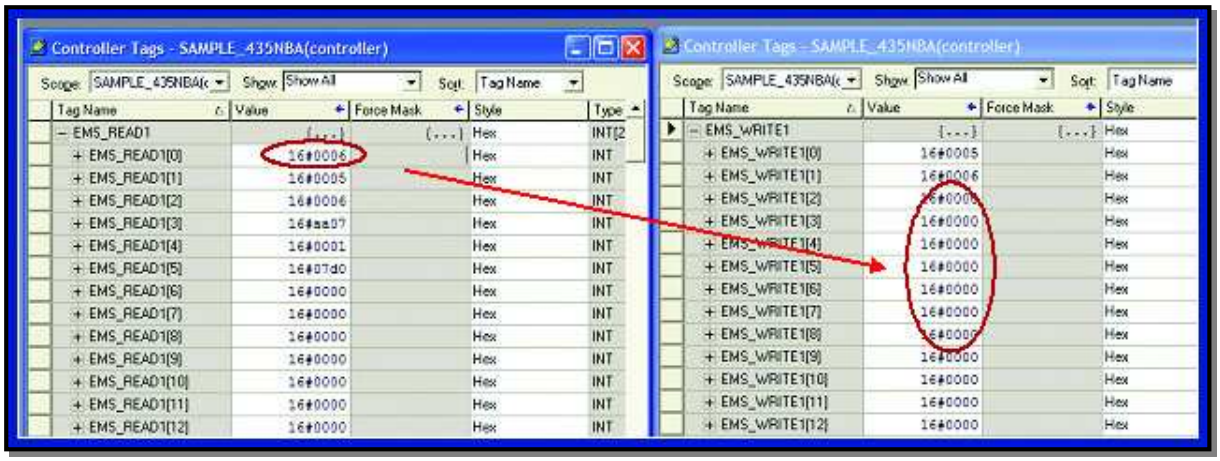
1. The PLC writes the command to the Read Tag (EMS_READ1) and then increments the counter in EMS_READ1 [1]
2. The counter in EMS_READ1 [1] is copied by the Gateway to EMS_WRITE1 [0] which acknowledges that the command has been received.



3. Following execution of the command, the Gateway copies the response from Node 01 to EMS_WRITE1 (the Write Tag for Node 01) and increments the counter in EMS_WRITE1 [1]. This signals that there is new data for the PLC (i.e. the RFID controller generated response).



4. After the PLC has processed the response information, it copies the counter from EMS_WRITE1 [1] to EMS_READ1 [0] which signals to the Gateway that the PLC has retrieved the response data.



5. The Gateway will then clear (set to 0) holding register 40001 of the Write Tag, data from EMS_WRITE1. After which it will be ready to receive another command.

5.9 ETHERNET/IP: OBJECT MODEL

The ***Object Model*** is the logical organization of attributes (parameters) within classes (objects) and services supported by each device.

Objects are broken down into three categories: ***Required Objects***, ***Vendor Specific Objects*** and ***Application Objects***.

- **Required Objects** are classes that must be supported by all devices on EtherNet/IP. The Gateway has six Required Objects.
- **Vendor Specific Objects** are classes that add attributes and services that do not fit into the Required Objects or Application Objects categories. The Gateway has two Vendor Specific Objects.
- **Application Objects** are classes that must be supported by all devices using the same profile. An example of a profile is a Discrete I/O device or an AC Drive. This ensures that all devices with the same profile have a common look on the network.

Data Type Definition Table

EtherNet/IP was designed by the *Open Device Vendors Association* (**ODVA**) as an open protocol. The following table contains a description of the data types used by ODVA that are also found in this chapter.

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

5.9.1 Ethernet/IP Required Objects

Under Ethernet/IP, there are **six** *Required Objects*:

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

Identity Object (0x01 - 1 Instance)

Class Attributes

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

Instance Attributes

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Vendor Number	UINT	50 DEC	Get
2	Device Type	UINT	0x0C	Get
3	Product Code Number	UINT	6102 DEC	Get
4	Product Major Revision Product Minor Revision	USINT USINT	01 25	Get
5	Status Word (see below for definition)	WORD	See Below	Get
6	Serial Number	UDINT	Unique 32 Bit Value	Get
7	Product Name: Product Name Size Product Name String	 USINT USINT[26]	GWY-01-IND-01 07 “Gateway”	Get

Status Word

Bit	Bit = 0	Bit = 1
0	No I/O Connection	I/O Connection Allocated
1 – 15	Unused	Unused

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x0E	Yes	Yes	Get Attribute Single
0x05	No	Yes	Reset

Message Router Object (0x02)

This object has no supported attributes.

Assembly Object (0x04 - 3 Instances)

Class Attributes

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Revision	UINT	1	Get
2	Max Instance	UINT	81	Get

Instance 0x64 Attributes (Input Instance)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
3	Status Information:			Get
	Bitmap of Consume Instances with Data	DINT	0	
	Bitmap of Produce Instances with Data	DINT	0	

User Datagram Protocol (UDP) I/O Sequence Number Handshaking

The data producing device increments the data sequence number by one with the transmission of each new serial data packet. Valid sequence numbers are 1-65535. After the consuming device has processed the data, it must echo the sequence number in the handshake to allow the producing device to remove the data from the queue. This is required for I/O communications because UDP is not guaranteed to arrive in order.

If the Node ID number is passed as part of the I/O message, the message is stored to the appropriate location in the Modbus RTU table. Because communications are asynchronous, the Node ID number is also stored as part of the output data. It is the responsibility of the PLC programmer to make sure the proper request lines up with the proper response if the Gateway is used as a request/response device.

Instance 0x65 Attributes (Input Instance 2)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
3	Serial Produce Data:			Get
	Consume Data Seq. Number Handshake	UINT	0	
	Produce Data Sequence Number	UINT	0	
	Node 1 Serial Produce Data Size	UINT	0	
	Node 1 Serial Produce Data	WORD[100]	All 0’s	

Instance 0x66 Attributes (Input Instance 3)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
3	Serial Produce Data:			Get
	Consume Data Seq. Number Handshake	UINT	0	
	Produce Data Sequence Number	UINT	0	
	Node ID (1-32)	UINT	1	
	Node Serial Produce Data Size	UINT	0	
	Node Serial Produce Data	WORD[100]	All 0's	

Instance 0x70 Attributes (Output Instance 1)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
3	Serial Consume Data:			Get / Set
	Produce Data Seq. Number Handshake	UINT	0	
	Consume Data Sequence Number	UINT	0	
	Node 1 Serial Consume Data Size	UINT	0	
	Node 1 Serial Consume Data	WORD[100]	All 0's	

Instance 0x71 Attributes (Output Instance 2)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
3	Serial Consume Data:			Get / Set
	Produce Data Seq. Number Handshake	UINT	0	
	Consume Data Sequence Number	UINT	0	
	Node ID (1-32)	UINT	1	
	Node Serial Consume Data Size	UINT	0	
	Node Serial Consume Data	WORD[100]	All 0's	

Instance 0x80 Attributes (Configuration Instance)

Most I/O clients include a configuration path when opening an I/O connection to a server. There is no configuration data needed.

Instance 0x81 Attributes (Heartbeat Instance – Input Only)

This instance allows clients to monitor input data without providing output data.

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x0E	Yes	Yes	Get Attribute Single
0x10	No	Yes	Set Attribute Single

Connection Manager Object (0x06)

This object has no attributes.

TCP Object (0xF5 - 1 Instance)

Class Attributes

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

Instance Attributes

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

*See section 5-3.2.2.1 – 5-3.2.2.6 of “Volume 2: EtherNet/IP Adaptation of CIP” from ODVA for more information regarding these attributes.

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x0E	Yes	Yes	Get Attribute Single

Ethernet Link Object (0xF6 - 1 Instance)

Class Attributes

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

Instance Attributes

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Interface Speed*	UDINT	100	Get
2	Interface Flags*	DWORD	3	Get
3	Physical Address*	USINT Array[6]	0	Get

*See section 5-4.2.2.1 – 5-4.2.2.3 of “Volume 2: EtherNet/IP Adaptation of CIP” from ODVA for more details on this attribute.

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x0E	Yes	Yes	Get Attribute Single

5.9.2 EtherNet/IP: Vendor Specific Objects

The Gateway has two Vendor Specific Objects:

Vendor Specific Objects:

Gateway Consume Data Object (0x64)
Gateway Produce Data Object (0x65)

Gateway Consume Data Object (0x64 - 32 Instances)

Class Attributes (Instance 0)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Revision	UINT	1	Get
2	Maximum Consume Data Buffer Size (in words)	UINT	32768	Get
3	Bitmap of Consume Instances with Data Bit 0: Instance 1 … Bit 31: Instance 32	DINT	0	Get

Instance Attributes (Instances 1-32)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Consume Data Size (in words)	UINT	0	Get / Set
2	Consume Data [0-249]	UINT	0	Get / Set
3	Consume Data [250-499]	UINT	0	Get / Set
4	Consume Data [500-749]	UINT	0	Get / Set
5	Consume Data [750-999]	UINT	0	Get / Set
6	Consume Data [1,000-1,249]	UINT	0	Get / Set
…	…	…	…	…
10	Consume Data [2,000-2,249]	UINT	0	Get / Set
…	…	…	…	…
34	Consume Data [8,000-8,249]	UINT	0	Get / Set
…	…	…	…	…
38	Consume Data [9,000-9,249]	UINT	0	Get / Set
…	…	…	…	…
42	Consume Data [10,000-10,249]	UINT	0	Get / Set
…	…	…	…	…
82	Consume Data [20,000-20,249]	UINT	0	Get / Set
…	…	…	…	…
122	Consume Data [30,000-30,249]	UINT	0	Get / Set
…	…	…	…	…
126	Consume Data [31,000-31,249]	UINT	0	Get / Set
…	…	…	…	…
130	Consume Data [32,000-32,249]	UINT	0	Get / Set
131	Consume Data [32,250-32,499]	UINT	0	Get / Set
132	Consume Data [32,500-32,749]	UINT	0	Get / Set
133	Consume Data [32,750-32,767]	UINT	0	Get / Set

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x05	No	Yes	Reset*
0x0E	Yes	Yes	Get Attribute Single
0x10	No	Yes	Set Attribute Single

*This Service Code is used to flush all attributes to zero.

Gateway Produce Data Object (0x65 - 32 Instances)

Class Attributes (Instance 0)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Revision	UINT	1	Get
2	Maximum Produce Data Buffer Size (in words)	UINT	32768	Get
3	Bitmap of Produce Instances with Data Bit 0: Instance 1 … Bit 31: Instance 32	DINT	0	Get

Instance Attributes (Instances 1-32)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	Produce Data Size (in words)	UINT	0	Get / Set
2	Produce Data [0-249]	UINT	0	Get
3	Produce Data [250-499]	UINT	0	Get
4	Produce Data [500-749]	UINT	0	Get
5	Produce Data [750-999]	UINT	0	Get
6	Produce Data [1,000-1,249]	UINT	0	Get
…	…	…	…	…
10	Produce Data [2,000-2,249]	UINT	0	Get
…	…	…	…	…
34	Produce Data [8,000-8,249]	UINT	0	Get
…	…	…	…	…
38	Produce Data [9,000-9,249]	UINT	0	Get
…	…	…	…	…
42	Produce Data [10,000-10,249]	UINT	0	Get
…	…	…	…	…
82	Produce Data [20,000-20,249]	UINT	0	Get
…	…	…	…	…
122	Produce Data [30,000-30,249]	UINT	0	Get
…	…	…	…	…
126	Produce Data [31,000-31,249]	UINT	0	Get
…	…	…	…	…
130	Produce Data [32,000-32,249]	UINT	0	Get
131	Produce Data [32,250-32,499]	UINT	0	Get
132	Produce Data [32,500-32,749]	UINT	0	Get
133	Produce Data [32,750-32,767]	UINT	0	Get

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x05	No	Yes	Reset*
0x0E	Yes	Yes	Get Attribute Single
0x10	No	Yes	Set Attribute Single

*This Service Code is used to flush all attributes to zero.

5.9.3 Application Object (0x67 _ 10 Instances)

Class Attributes (Instance 0)

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

Instance Attributes (Instances 1-32)

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
1	**Instance Type** (0-3): 0 - Disable 1 – ControlLogix 2 – SLC 5/05 3 – PLC5E	USINT	0	Get
2	**PLC IP Address**	UDINT	0	Get
3	**PLC Slot Location** (0-255)	USINT	0	Get
11	**Max Write Size** in Words: 0 – Disabled 1 – 100 Words	UINT	0	Get
12	**Write Tag Name** (ControlLogix Only)	SHORT STRING	0	Get
13	**Write File Number** (SLC/PLC Only) **NX:0** - where **"X"** is the File Number	UINT	7	Get
14	**Write File Offset** (SLC/PLC Only) **N7:Y** - where **"Y"** is the File Offset	UINT	0	Get
15	**Write "Heartbeat" Timeout** Measured in 10 ms "ticks" 0 = disabled Max value: 6000 ticks	UINT	100	Get
21	**Max Read Size** in Words 0 – Disable Max Value: 100	UINT	0	Get
22	**Read Tag Name** (ControlLogix Only)	SHORT STRING	0	Get
23	**Read File Number** (SLC/PLC Only) **NX:0 -** Where **"X"** is the File Number	UINT	7	Get

Attribute ID	Name / Description	Data Type	Default Data Value	Access Rule
24	**Read File Offset** (SLC/PLC Only) **N7:Y** - Where **"Y"** is the File Offset	UINT	0	Get
25	**Read Poll Rate** Measured in 10 ms "ticks" 0 = disabled 6000 ticks max	UINT	100	Get

Common Services

Service Code	Implementation		Service Name
	Class Level	Instance Level	
0x0E	Yes	Yes	Get Attribute Single

6 MODBUS TCP INTERFACE



NOTE *For BIS Z-GW-001-IND models.*

One of the most popular and well-proven industrial automation protocols in use today is Modbus. Modbus is an open client/server application protocol. Modbus TCP allows the Modbus protocol to be carried over standard Ethernet networks. Modbus TCP is managed by the Modbus-IDA User Organization.

6.1 MODBUS TCP OVERVIEW

Under the Modbus TCP protocol, the Gateway acts as a Modbus Server and the PLC acts as a Modbus Client. By utilizing **Produce** and **Consume** registers for mapping commands and responses, data produced by the Gateway is consumed by the Modbus Client and data produced by the Modbus Client is consumed by the Gateway.

- Modbus Client (Host or PLC) must connect to the Modbus Server (Gateway) on port **502**
- Maximum number of words transferred to/from an RFID tag per read/write cycle: **100 Words / 200 Bytes**
- Disable any firewall services running on the PC. Firewalls can potentially block communications between the processor unit and the host and/or PLC.

6.2 MODBUS TCP CONFIGURATION VIA HTTP SERVER

To configure the Gateway for Modbus TCP communications, begin by assigning it a locally compatible IP address.

Through a standard Web browser, you can utilize the Industrial Ethernet Gateway's *HTTP Server* to access an embedded suite of configuration tools, called the "***OnDemand Utilities***." Among its features is the ability to modify and save changes to the Gateway's IP address, which is stored internally on the Gateway.

Subnet16™ Industrial Ethernet Gateway - Default IP Address:
192.168.253.110

Setting the Gateway IP Address

To set the Gateway's IP address using the *HTTP Server,* follow the steps below:

1. Open a Web browser on the host.
2. In the URL address field, enter the Gateway's IP address (*192.168.253.110 = factory default)***.**
3. Press ***ENTER***.

The *HTTP Server - Main Page* will be displayed.

HTTP Server – Main Page

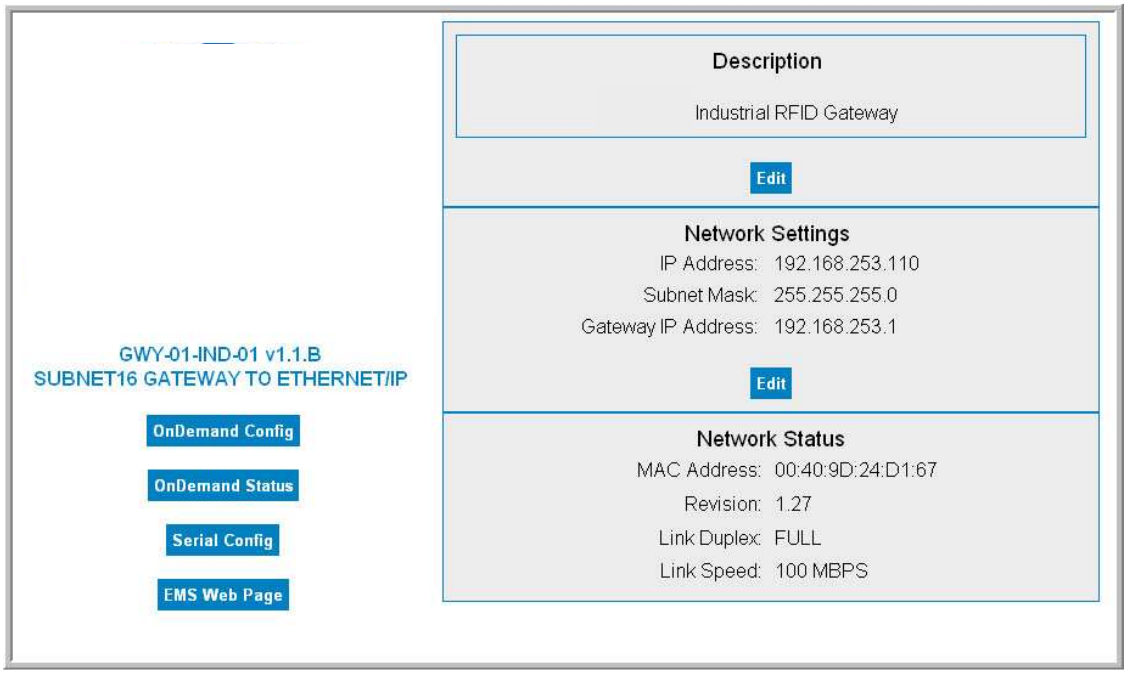
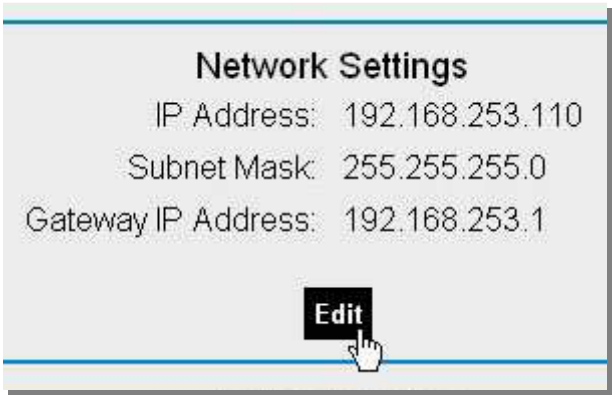


Figure 36 - The HTTP Server - Main Page

The ***HTTP Server - Main Page*** lists the IP address and network settings currently stored on the Gateway.

4. Click the button labeled "***EDIT***", located below "***Network Settings.***"



The IP Configuration Page will be displayed.

IP Configuration Page

The *IP Configuration Page* is used to modify and save changes to the IP Address, Subnet Mask and (Network) Gateway IP Address.



Figure 37 - The IP Configuration Page

5. In the fields provided, enter your new IP configuration values for the Gateway.
6. Click the **"*Save Settings*"** button to store the new IP configuration. The Gateway will completely reset and your IP changes will be implemented.
7. After the Gateway has restarted, verify the new IP configuration by opening a Web browser and manually entering the Gateway's new IP address in the URL field. If successful, you should arrive back at the *HTTP Server – Main Page*.

6.2.1 Modbus TCP - Command Packet Structure

Consume Registers hold data that is destined for the Gateway or Processor units. Modbus TCP commands must be placed in the holding registers, starting at address 40001, of Device ID 01 (Node Input Page 01). Commands utilize at least six registers (double-byte values or words).

Modbus Address (4xxxx / 3xxxx)	Read / Write Privilege	Register Description
(40001) 1	R/W	2-byte Gateway Consume Data Overall Length (> 0 indicates data is available; Gateway clears to 0 after data is processed)
2	R/W	MSB = Reader Type LSB = Command ID
3	R/W	MSB = 0x00 LSB = Node ID (1-16 or 32)
4	R/W	2-byte Timeout Value (0-65535) measured in milliseconds
5	R/W	2-byte Read/Write Start Address (0-65535)
6	R/W	2-byte Read/Write Block Size (0-65535 bytes)
7 – 32774	R/W	Gateway Consume Data (when applicable)
32775 – 65536	R/W	Reserved

6.2.2 Modbus TCP - Response Packet Structure

Produce Registers hold data that is destined for the host or PLC.

Modbus Address (4xxxx / 3xxxx)	Read / Write Privilege	Register Description
(40001) 1	R/W	2-byte Gateway Produce Data Overall Length (> 0 indicates data is available; Modbus Client clears to 0 after data is processed)
2	RO	MSB = Reader Type LSB = Command Echo
3	RO	Node ID Number (33-48, 64)
4	RO	Timeout Value (0-65535)
5	RO	Read/Write Start Address (0-65535)
6	RO	Read/Write Block Size (0-65535 bytes)
7 – 32774	RO	Gateway Produce Data (when applicable)
32775 – 65536	RO	Reserved

6.2.3 Modbus TCP - Mapping for Node 65

Modbus Address (4xxxx)	Read / Write Privilege	Register Description
1	R/W	IP Address 1 (MSB) Example: 192
2	R/W	IP Address 2 Example: 168
3	R/W	IP Address 3 Example: 000
4	R/W	IP Address 4 (LSB) Example: 100
5	R/W	Subnet Mask 1 (MSB) Example: 255
6	R/W	Subnet Mask 2 Example: 255
7	R/W	Subnet Mask 3 Example: 255
8	R/W	Subnet Mask 4 (LSB) Example: 000
9	R/W	Gateway Address 1 (MSB) Example: 192
10	R/W	Gateway Address 2 Example: 168
11	R/W	Gateway Address 3 Example: 000
12	R/W	Gateway Address 4 (LSB) Example: 001
13	RO	MAC Address 1 (MSB) Example: 0x00
14	RO	MAC Address 2 Example: 0x40
15	RO	MAC Address 3 Example: 0x9D
16	RO	MAC Address 4 Example: 0x12
17	RO	MAC Address 5 Example: 0x34
18	RO	MAC Address 6 (LSB) Example: 0x56
19	RO	Link Status: 0 = No Link 1 = Link is OK
20	RO	Ethernet Speed (10M or 100M bits)
21	RO	Link Duplex: 0 = Half Duplex 1 = Full Duplex
22	RO	Revision (Major/Minor)
23 – 1000	R/W	Reserved
1001	RO	(Input) Data Ready Mask - Nodes 1 - 16
1002	RO	(Input) Data Ready Mask - Nodes 17 - 32
1003	RO	(Output) Data Ready Mask - Nodes 33 - 48
1004	RO	(Output) Data Ready Mask - Nodes 49 - 64
1005-10099	R/W	Reserved
10100 – 10199	R/W	OnDemand Node 1 Configuration (See *OnDemand Settings Table* below)
10200 – 10299	R/W	OnDemand Node 2 Configuration
…	…	…
13100 – 13199	R/W	OnDemand Node 31 Configuration
13200 – 13299	R/W	OnDemand Node 32 Configuration
13300 – 65536	R/W	Reserved

Modbus Address (4xxxx)	Read / Write Privilege	Register Description
10100	R/W	Instance Type: 0 = Disable 1 = ControlLogix 2 = SLC 5/05 3 = PLC 5E
10101	R/W	PLC IP Address 1 MSB
10102	…	PLC IP Address 2
10103	R/W	PLC IP Address 3
10104	R/W	PLC IP Address 4 LSB
10105	R/W	PLC Slot Location (0-255)

6.3 MODBUS TCP - HANDSHAKING

Due to the process with which commands and responses are passed between the Gateway and the host, a handshaking procedure is used to notify the host that returning data is available for retrieval.

Overall Length

The handshaking process is governed by the changing of the "***Overall Length***" value within a data packet. The Overall Length value is typically the first word (2-bytes) of a command or response and indicates the total number of data words in the packet.

Node Input and Node Output Pages

Under the Modbus TCP protocol, host-generated data is written to a pre-defined region of the Gateway's own memory known as the ***Node Input Page***. Host-bound data generated by the Gateway, is written to a separate region of the Gateway's memory known as the ***Node Output Page*** (in Modbus TCP these regions of memory are called **Device IDs**). Each Subnet Node has its own Node Input and Node Output Page (Device ID) which is used to temporarily hold incoming (controller-bound) and outgoing (host-bound) data.

Output Data Ready Mask

To notify the host that new data is waiting to be retrieved from one of the Node Output Pages, the Gateway utilizes a separate 32-bit block of internal memory, called the ***Output Data Ready Mask***.
The lowest 16 bits of the 32-bit Output Data Ready Mask each represent the status of one Node Output Page (one for each of the 16 Subnet Nodes). For example, the first or lowest bit (*bit 01*) represents Node Output Page 33 - which holds output data from Node 01. Bit 02 represents Node Output Page 34 - which holds output data from Subnet Node 02, and so forth.
The Gateway, itself, is assigned Subnet Node 32 and thus, its corresponding Node Output Page is 64. Node Output Page 64 is represented by the final bit (*bit 32*) in the Output Data Ready Mask. *See section 4.1 for more information on Node Input and Node Output Pages.*

Holding Registers

When writing host-bound data to one of the Node Output Pages, the Gateway actually places each byte of the data packet into pre-defined "***holding registers***" within the Node Output Page of the Subnet Node that generated the data. Note that a single holding register stores 2-bytes or one word of data. The 2-byte *Overall Length* value, for example, is written to the first holding register (which is location **40001)** of the data producer's Node Output Page.
Then, as the Gateway finishes writing host-bound data (such as a controller's response) to the Node Output Page of the data producing Subnet Node, the Overall Length value (stored at holding register 40001) will change from its default value of *0x00* to reflect the number of data words within the newly written host-bound data packet. This change to the Overall Length value (i.e. register 40001) within a Node Output Page, triggers the Gateway to enable (change from zero to one) the corresponding bit in the Output Data Ready Mask. It is when a bit in the Output Data Ready Mask has become enabled, that the host will recognize the pending data.
Finally, after the host has retrieved its pending data, the enabled bit in the Output Data Ready Mask and the Overall Length value at holding register 40001 of the Node Output Page will be reset to zero (0x00), indicating that the host has received and processed its pending data.

6.3.1 Modbus TCP - Host/processor unit Handshaking

When the host issues a command, it must first write the entire command to the given Node Input Page, leaving the Overall Length value to be written last.
For example, for the host to issue the 6-word command "*Read Data from Node 3*," it must first write the last five words of the command to Node Input Page 03, beginning at register 40002. After which, the host will fill in the first word (at holding register 40001) with the Overall Length of the command packet.

Last Five Words of a Read Data Command

Word	MSB	LSB	Description
02	0xAA	0x05	**Command ID:** Read Data
03	0x00	0x03	**Node ID:** 3
04	0x03	0xE8	**Timeout Value:** 1 second
05	0x00	0x20	**Read Start Address**: 0x0020
06	0x00	0x04	**Block Size**: 4 Bytes

After writing the last five words of the command, the host will write the Overall Length value to holding register 40001 of Node Input Page 03.

First Word of a Read Data Command

Word	MSB	LSB	Description
01	0x00	0x06	**Overall Length** (*in words*)

The moment the Overall Length value (at holding register 40001) of Node Input Page 03 changes from 0x00 to a "non-zero" value, the Gateway will recognize the waiting data and will distribute the command to the Processor unit at Subnet Node 03.

6.3.2 Modbus TCP - Handshaking Example

1. The host or PLC issues an RFID command to Subnet Node 04, writing the command string to the holding registers for Device ID 04 (Node Input Page 04). An Overall Length value of 0x06 is written last to holding register 40001.
2. The Gateway recognizes that the Overall Length value at holding register 40001 has changed for Device ID 04 (Node Input Page 04), indicating that a command is waiting to be directed to the Processor unit at Subnet Node 04. The Gateway distributes the command accordingly.
3. The Gateway now having passed the command to the specified Processor unit, clears the Overall Length holding register of Device ID 04 (Node Input Page 04), setting it back the default value of zero (0x00).



NOTE

When the Node Input Page's value at register 40001 is returned to 0x00, the host can assume that the command was at least received and execution was attempted. The host can also assume that it is OK to clear the remaining holding registers and write another command to the Device ID (Node Input Page).

4. Meanwhile, the Processor unit at Subnet Node 04 executes its given command instructions and generates a controller command response.
5. The Gateway retrieves the command response data from the Processor unit at Subnet Node 04 and writes the host-bound content to the holding registers for Device ID 36 (Node Output Page 36). Again, the Overall Length value is written last to holding register 40001.



NOTE

Host-bound data is always written to a Device ID (Node Output Page) that is 32 greater in number than the Node ID of the data producing device.

6. With holding register 40001 of Device ID 36 (Node Output Page 36) now containing a non-zero length value, the Gateway will enable (change from zero to 1) the fourth bit in the *Output Data Ready Mask*. (The fourth bit is allocated to Node Output Page 36, just as the fifth bit is allocated to Node Output Page 37, and so on).
7. Once bit 4 in the Output Data Ready Mask becomes enabled, the host retrieves the data string stored in the holding register area for Device ID 36 (Node Output Page 36).
8. After importing the data from Device ID 36 (Node Output Page 36), the host clears (sets back to 0x00) the Overall Length value at holding register 40001 of Device ID 36 (Node Output Page 36). In doing so, bit 4 in the Output Data Ready Mask is also cleared.



NOTE

The clearing of bit 4 in the Output Data Ready Mask indicates to the Gateway that the host has received the response and that it is now OK to write another response to Node Output Page 36.

This completes the Modbus TCP handshaking cycle.

7 STANDARD TCP/IP INTERFACE



NOTE

For BIS Z-GW-001-IND and BIS Z-GW-001-TCP models.

7.1 STANDARD TCP/IP OVERVIEW

Another means of communicating with the Gateway is through the standard TCP/IP protocol. For this manual, the protocol is referred to as ***Standard TCP/IP*** to distinguish it from other industrial protocols.

In this environment, the Gateway acts as the server and the host or PLC acts as client. Standard TCP/IP sessions are established between the host computer and the Gateway via TCP/IP client software. A TCP/IP session generally consists of three stages: *connection setup, data transactions* and *connection termination*.

All connections to the Gateway are initiated by client side software only. If, for example, an existing connection terminates unexpectedly, the Gateway will not attempt to contact the client software or re-establish a connection. The client is responsible for opening, maintaining, and closing all TCP/IP sessions.

After establishing a successful connection, communications between the host and the Gateway can proceed. When communication is no longer necessary, it is the responsibility of the client side application to terminate the connection.

- The TCP/IP client software (running on the host or PLC) must connect to the TCP/IP server (Gateway) on port **2101**.
- Maximum number of words transferred to/from an RFID tag per read/write cycle: **100 Words / 200 Bytes**.
- Disable any firewall services running on the PC. Firewalls can potentially block communications between the processor unit and the host and/or PLC.

7.2 IP CONFIGURATION VIA HTTP SERVER



NOTE

For BIS Z-GW-001-IND models only.

To configure the Gateway for standard TCP/IP communications, begin by assigning it a locally compatible IP address.

Through a standard Web browser, you can utilize the Industrial Ethernet Gateway's *HTTP Server* to access an embedded suite of configuration tools, called the "***OnDemand Utilities***." Among its features is the ability to modify and save changes to the Gateway's IP address, which is stored internally on the Gateway.

Subnet16™ Industrial Ethernet Gateway - Default IP Address:
192.168.253.110

Setting the Gateway IP Address

To set the Gateway's IP address using the *HTTP Server,* follow the steps below:

1. Open a Web browser on the PC.
2. In the URL address field, enter the Gateway's IP address (*192.168.253.110 = factory default*).
3. Press ***ENTER***.

The *HTTP Server - Main Page* will be displayed.

HTTP Server – Main Page

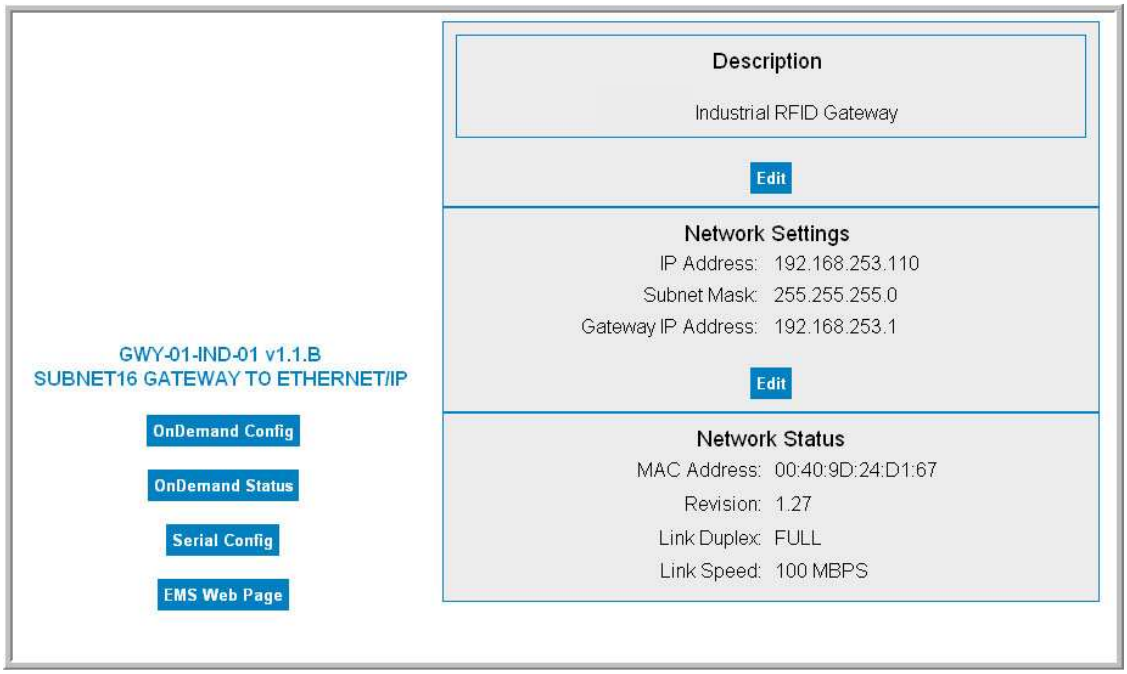
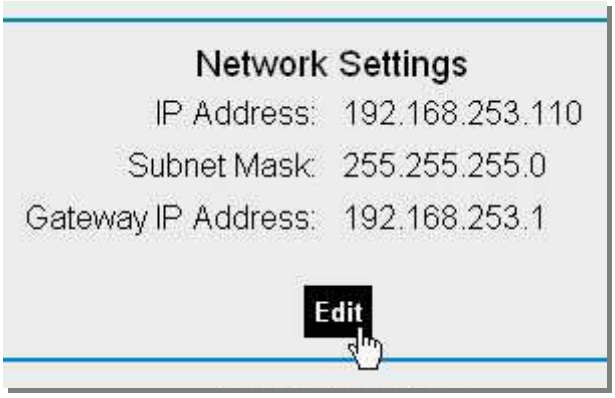


Figure 38 - The HTTP Server - Main Page

The ***HTTP Server - Main Page*** lists the IP address and network settings currently stored on the Gateway.

4. Click the button labeled "***EDIT***", located below "***Network Settings.***"



The IP Configuration Page will be displayed.

IP Configuration Page

The *IP Configuration Page* is used to modify and save changes to the IP Address, Subnet Mask and (Network) Gateway IP Address.



Figure 39 - The IP Configuration Page

5. In the fields provided, enter your new IP configuration values for the Gateway.
6. Click the **"*Save Settings***" button to store the new IP configuration. The Gateway will completely reset and your IP changes will be implemented.
7. After the Gateway has restarted, verify the new IP configuration by opening a Web browser and manually entering the Gateway's new IP address in the URL field. If successful, you should arrive back at the *HTTP Server – Main Page*.

7.3 IP CONFIGURATION VIA DIGI DISCOVERY



NOTE

For BIS Z-GW-001-TCP models only.

The BIS Z-GW-001-TCP interface module is designed to receive its IP address via DHCP. After your DHCP server has assigned an IP address to a TCP/IP Gateway, the IP address can be identified through the use of the "***Digi Device Discovery***" software tool.

Once the IP address is identified, you can connect the Gateway to the Balluff Dashboard™ Configuration Tool for system configuration and/or monitoring.

Digi Device Discovery is a separate utility available at www.balluff.com that identifies the IP address of the Subnet16™ TCP7IP Gateway Interface Modules.

7.4 STANDARD TCP/IP - COMMAND & RESPONSE EXAMPLES



NOTE

For BIS Z-GW-001-IND and BIS Z-GW-001-TCP models.

In standard TCP/IP, RFID commands issued by the host resemble Modbus TCP commands. The Gateway handles all handshaking tasks.

Moreover, the command & response packets need an additional word at the beginning of the string:

Protocol Header 0xFF in MSB, **<Node ID>** in LSB.

Please notice that these two bytes are not considered part of the CBx command packet and should not be counted in the Overall Length.

Below is the structure of the additional word required, named as **Word # 00**:

Word #	Command Packet Element	MSB	LSB
00	**CBx Protocol Header** in MSB: *0xFF* **Node ID** in LSB	0xFF	<Node ID>

And similarly for the response:

Word #	Response PACKET ELEMENT	MSB	LSB
00	**CBx Protocol Header** in MSB: *0xFF* **Node ID Echo** in LSB	0xFF	<Node ID Echo>



NOTE

These first two bytes will not be returned in the response packet for commands executed by Node 01.

Therefore, the command packet structure for Standard TCP/IP applications is:

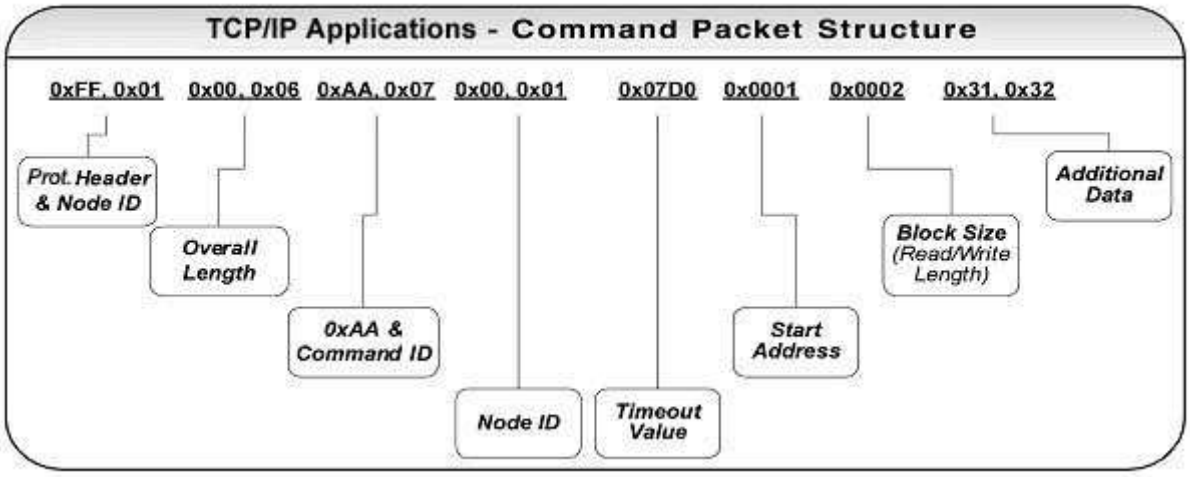


Figure 40 - Standard TCP/IP Protocol Command Packet Structure

7.4.1 Standard TCP/IP - Command Structure Example

In the following example, a 12-byte command has been issued to an RFID controller at Node 01 in a Subnet16™ network, instructing the controller to read six bytes from a tag within RF range. A *Timeout Value* of five seconds has been set for the completion of the command.

Word	Description	MSB	LSB
00	**CBx Protocol Header** in MSB = 0xFF **Node ID** in LSB = Node 01 first controller in the Subnet16™ network (0x01)	0xFF	0x01
01	**Overall Length:** 2-byte integer indicating number of "*words*" in the command packet	0x00	0x06
02	**MSB** = 0xAA **LSB** = Command ID: (example: *0x05 – Read Data*)	0xAA	0x05
03	**MSB** = 0x00 **LSB** = Node ID: Node 01 first controller in the Subnet16™ network (0x01)	0x00	0x01
04	**Timeout Value:** 2-byte integer measured in .10 second increments. (*0x0032 = 50 x .10 or 5 seconds*)	0x00	0x32
05	**Start Address:** 2-byte integer identifies tag address where read will begin	0x00	0x01
06	**Block Size:** 2-byte integer indicates number of bytes to retrieve	0x00	0x06

7.4.2 Standard TCP/IP - Response Structure Example

The following resembles a typical response to the command issued in the previous example:

Word	Description	MSB	LSB
00	**CBx Protocol Header** in MSB = **0xFF** **Node ID** in LSB = Node 01 first controller in the Subnet16™ network (0x01)	0xFF	0x01
01	**Overall Length:** 2-byte integer indicating number of "*words*" in the response packet	0x00	0x09
02	**MSB** = 0xAA **LSB** = Command Echo: (*0x05 - Read Data)*	0xAA	0x05
03	**MSB** = Instance Counter **LSB** = Node ID: *0x01*	<IC>	0x01
04	**Time Stamp:** Month / Day (*March 19th*)	0x03	0x13
05	**Time Stamp:** Hour / Minute (*8:14 a.m.)*	0x08	0x0E
06	**MSB** = Time Stamp: Seconds **LSB** = Number of Additional Bytes Retrieved: *6*	0x00	0x06
07	**Retrieved Bytes** 1 & 2	0x61	0x62
08	**Retrieved Bytes** 3 & 4	0x63	0x64
09	**Retrieved Bytes** 5 & 6	0x65	0x66

See the CBx Command Protocol Reference Manual for complete information on command structure and contents.

8 DEVICENET INTERFACE



NOTE

For BIS Z-GW-001-DNT models.

8.1 DEVICENET OVERVIEW

DeviceNet is a digital, multi-drop network based on the CAN (Controller Area Network) specification, which permits easy connectivity between industrial controllers and I/O devices.

When the Gateway is connected to a DeviceNet network, it is considered an individual node for which a unique Node Address number between 1 and 63 is assigned (this is not to be confused with a Subnet Node ID number, for which the Gateway has 16). The DeviceNet Gateway conforms to the standards set by the Open DeviceNet Vendor Association (ODVA).

8.2 DEVICENET CONFIGURATION

8.2.1 Importing the Controller.EDS File

After making all necessary hardware connections, the next step in configuring the *BIS Z-GW-001-DNT* for DeviceNet is to import the .EDS file.



NOTE

Electronic Data Sheets (.EDS) are basic text files that are utilized by network configuration tools to identify and configure hardware devices for DeviceNet networks. A typical .EDS file contains a description of the product, its device type, hardware version and configurable parameters.*

*The .EDS file (filename: "**DeviceNet EDS.zip**") for the BIS Z-GW-001-DNT is available from the technical support area of the Balluff website.*

1. Download the .EDS file to the computer running your network's Rockwell Automation software (i.e. the host computer).
2. Using the *EDS Hardware Installation Tool,* located in the *RSLinx™ Tools* program group, import the .EDS file into your RSNetWorx/DeviceNet system. Refer to Rockwell Automation's documentation for specific instructions.
3. After you have imported the .EDS file, close and restart all Rockwell Automation programs. If you are uncertain which programs to close, cycle power to the host computer after importing the .EDS file.

8.2.2 Configuring Gateway and PLC DeviceNet Communications

After importing the .EDS file and rebooting the host computer (or after restarting your Rockwell Automation software), follow the steps below to continue configuring DeviceNet network communications between the Gateway and a *ControlLogix* PLC.

1. On the host computer, start RSNetWorx for DeviceNet.
2. Go online (click NETWORK and select ONLINE).

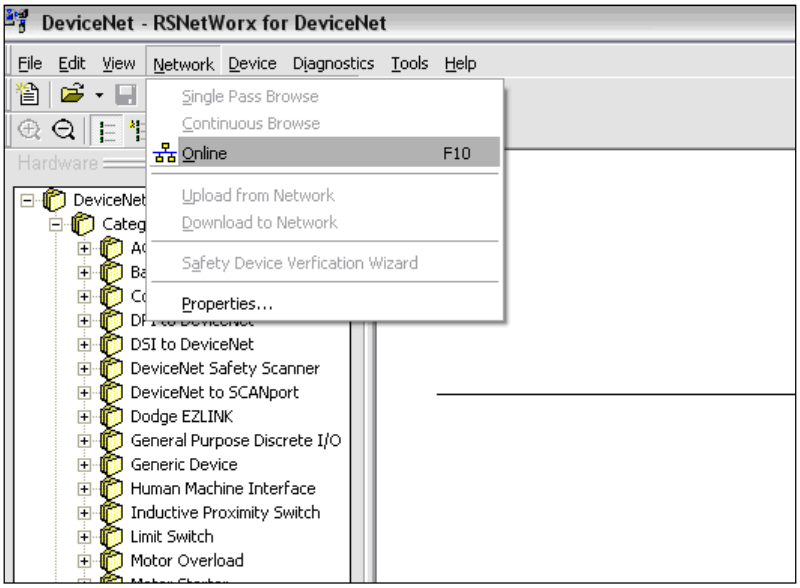
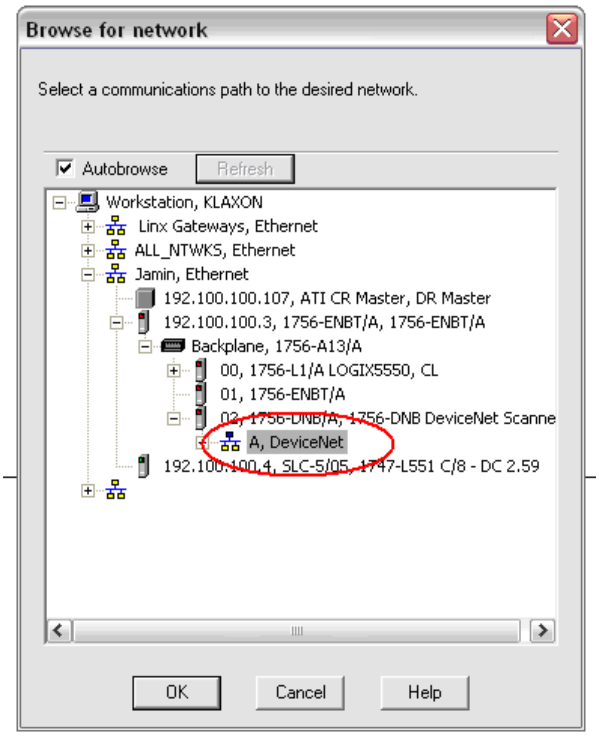


Figure 41 - Configuring Gateway for DeviceNet - Going Online



3. Select the appropriate DeviceNet network and then click "***OK***."

The *Scanner Configuration Applet* in *RSNetWorx* will begin scanning the specified network. This procedure may take some time depending on the speed of the bus and the number of devices connected. Node addresses are scanned from zero to 63. The default node address for the Gateway is 63.

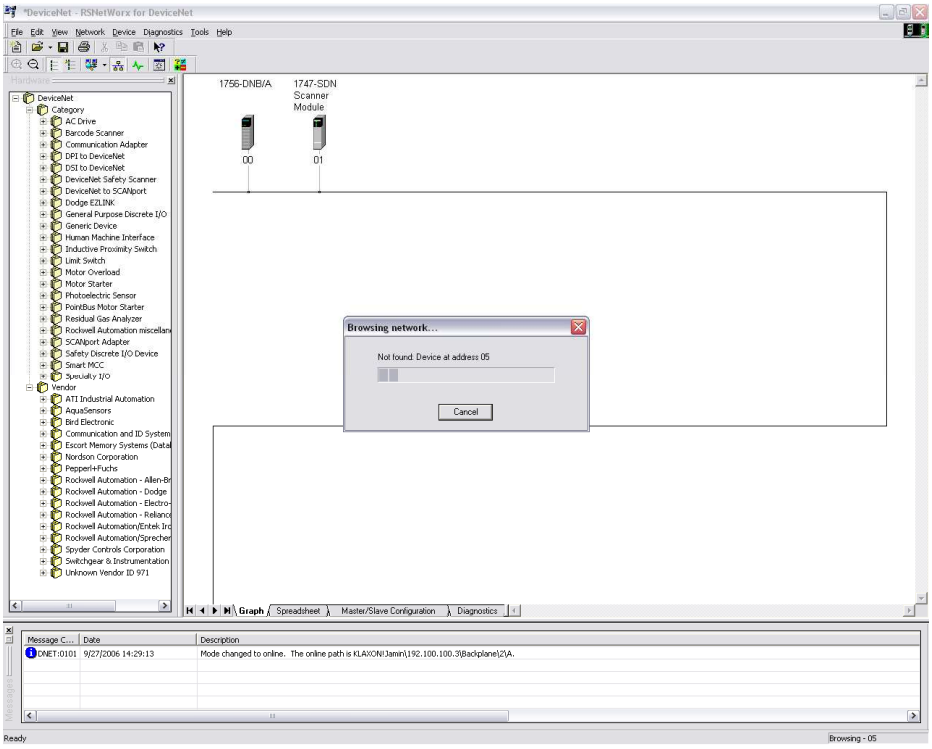


Figure 42 - Scanning Node Addresses on a DeviceNet Network

4. When the scan operation has completed, click "***UPLOAD***", in the *Scanner Configuration Applet* dialog box, to update the configuration of the *RSNetWorx* software.

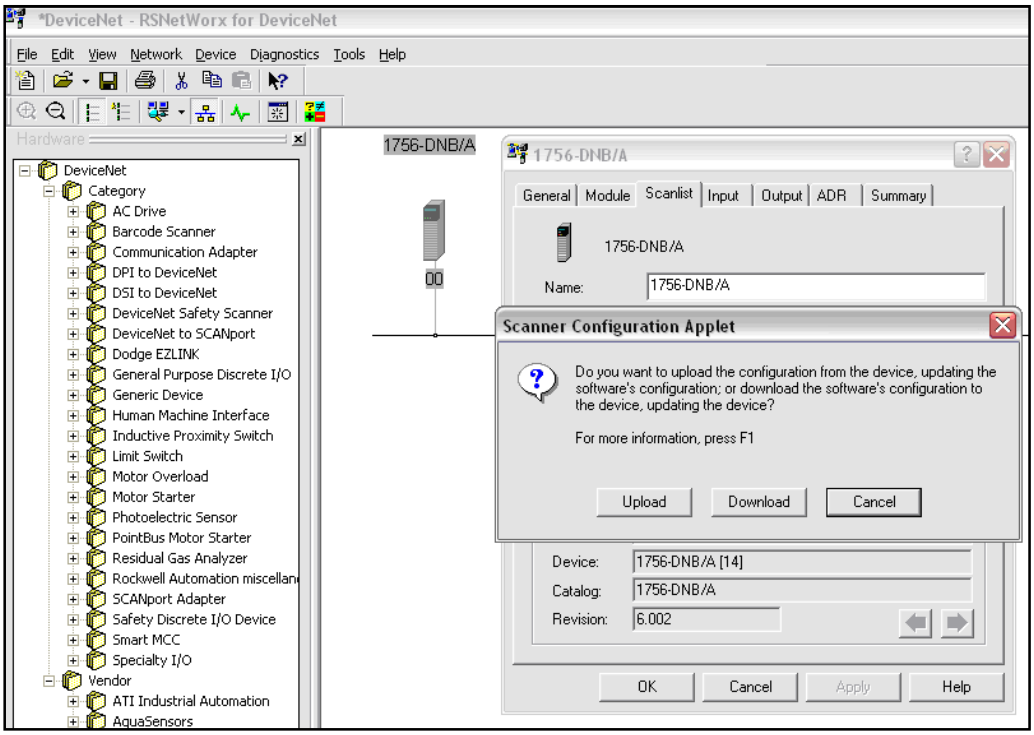
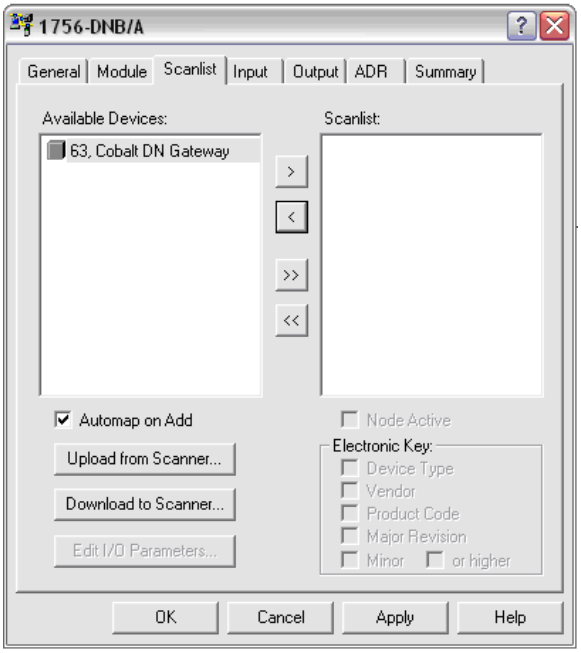


Figure 43 - Updating Configuration in RSNetWorx



NOTE

*The 1756-DNB/A is a Series A DeviceNet Bridge / Scanner Module. After updating the software, the Gateway should be recognized on the network and the device name, "**63**, **DN Gateway**", should be displayed under "Available Devices."*

5. Highlight the *Gateway* in the *Available Devices* list, and add it to the *Scanlist* field on the right hand side of the dialogue box. Click "***Apply***" and then "***OK***."

   The Gateway will be added to the list of DeviceNet hardware in RSNetWorx.

6. Next, select the *Gateway* from the list of DeviceNet hardware and edit its *I/O Parameters.* Set the *Input Size* and *Output Size* parameters according to your application requirements, then click "***OK.***" In the example below, 30 input bytes and 30 output bytes will be scanned per polling cycle.

NOTE

Strobed mode is not supported by the BIS Z-GW-001-DNT.

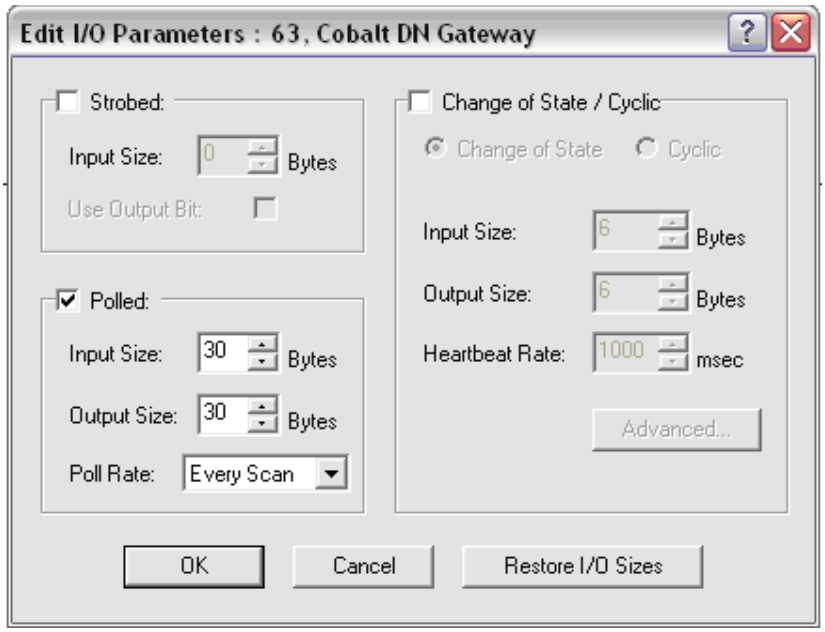


Figure 44 - Editing the Gateway's DeviceNet I/O Parameters

The following images display the **Input** and **Output** properties tabs (in *RSNetWorx for DeviceNet)* for the *1756-DNB/A DeviceNet Bridge / Scanner Module* after running the *Scanner Configuration Applet* for a second time. The scanner module, in this case, only identified one node, the Gateway at node address 63. The tabs are used to identify where input and output data is mapped for each identified node. In the image below, input data is mapped to start at **1:I.Data(0).0** on the PLC.

7. Run the *Scanner Configuration Applet* and verify the mapping of the address where the PLC will write input data for the Gateway.

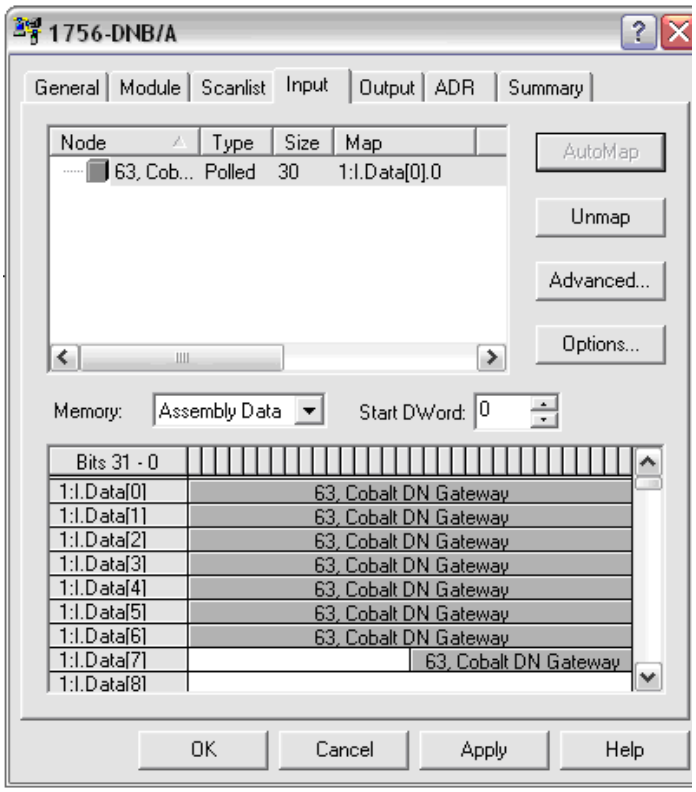


Figure 45 - 1756-DNB/A Input Properties Tab

8. Next, verify the mapping of the address where the PLC will retrieve output data from the Gateway. In the image below, output data is mapped to start at **1:O.Data(0).0** on the PLC.

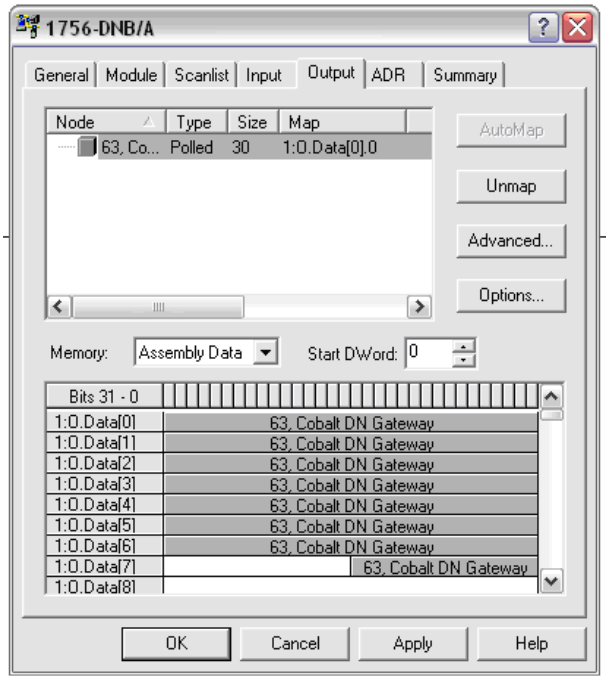


Figure 46 - 1756-DNB/A Output Properties Tab

9. Lastly, click "***Apply***" and select "***YES***" to download the configuration and mapping settings from *RSNetWorx* to the PLC.

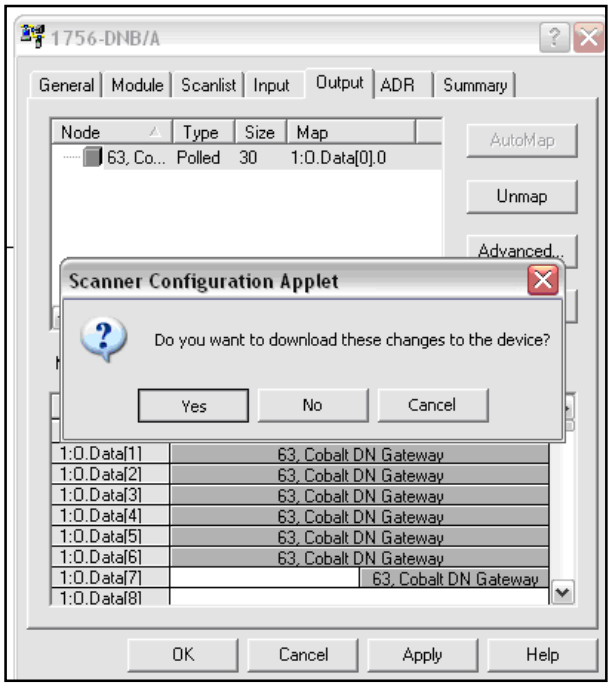


Figure 47 - 1756-DNB/A Output Properties Tab

8.2.3 Configuring Data Rate and Node Address

As noted, each device, computer and controller on a DeviceNet network is considered an individual node for which a unique *Node Address* number (between 0 and 63) is assigned. The node address provides a means of numerically identifying each device on a DeviceNet network.

Prior to operating the *BIS Z-GW-001-DNT*, you must verify that it has been configured for the same Data Rate as your network and that it has been assigned a suitable node address value. The Gateway supports data rates of 125Kb (default), 250Kb and 500Kb and supports node addresses 1 – 63 (default: 63).

To change the data rate or node address, use either the "*Node Commissioning*" tool in *RSNetWorx for DeviceNet* or the Balluff Dashboard™ Configuration Tool running on a host computer that is connected to the USB port on the Gateway. The *Balluff Dashboard™ Configuration Tool* is available on the Balluff Web site (www.balluff.com).



NOTE

When using node commissioning in RSNetWorx for DeviceNet, modify only one parameter at a time (either data rate or node address). After changing the data rate, you must manually cycle power to your DeviceNet network for the change to take effect.

Factory Default Configuration**:**

Data Rate = 125Kb

Node Address = 63

8.2.4 DeviceNet - Exchanging Data and Handshaking

After the Gateway has been properly configured for your DeviceNet network, it will be possible to send the Controller commands using the Balluff **CBx Command Protocol**. For , the CBx Command Protocol Manual is available on the Balluff Web site (www.balluff.com).

However, to ensure that messages to and from the Gateway are properly delivered and received, a handshaking mechanism has been implemented that uses a pair of dedicated words in the exchange.

The first two words in the *Input Controller Tag* and *Output Controller Tag* are dedicated to handshaking. When new information is generated, the data-producing device increments the counter value stored in the second word of a controller tag (either Input or Output, depending on the device). The data-consuming device, copies that same value to the counter in the first word of the reciprocal (or opposite) controller tag. This handshaking scheme signals to the data producer that the information has been received.

The image below displays an example of the data contained in the two I/O Controller Tags for the Gateway.

Figure 48 - Gateway I/O Controller Tags (in RSLogix 5000)

8.2.5 DeviceNet - Handshaking Example

This example describes the sequence of events for a simple command and response. All data is written in 2-byte *WORD* format and stored in 2-byte "*registers.*"
The ***Output Controller Tag*** holds command data written by the PLC. The ***Input Controller Tag*** holds response data generated by the Gateway. Handshaking is implemented using the first two words (*Words 0 and 1*) in both *Input Controller Tag* and *Output Controller Tags*.

1. The PLC writes a command to the *Output Controller Tag*, starting with the 2-byte *Consume Data Size* value at **Local:2:O.Data [2]** (which is the third register of the *Output Controller Tag*). The remainder of the command packet is then written, 2-byte per register, to the *Output Controller Tag,* starting at the fourth register, **Local:2:O.Data [3].** After writing the command packet data to the appropriate registers, the PLC increments the counter value stored at **Local:2:O.Data [1]** (the second register in the *Output Controller Tag*).

2. The counter at **Local:2:O.Data [1]** is copied by the Gateway to **Local:2:I.Data [0]** (the first register of the *Input Controller Tag*) which signals the PLC that the command has been received by the Gateway.

3. Following execution of the command, the Gateway writes its response to the *Input Controller Tag,* starting with the 2-byte *Produce Data Size,* at **Local:2:I.Data [2]** and the actual data beginning at **Local:2:I.Data [3]**. It then increments the counter value at **Local:2:I.Data [1]**. This alerts the PLC to the new data available (the Gateway generated response, in this case).

4. After processing the response information, the PLC copies the counter from **Local:2:I.Data [1]** to **Local:2:O.Data [0]**, which signals to the Gateway that the PLC has retrieved the response data.

OUTPUT CONTROLLER TAG

Controller Tag Location and Data	Description
Local:2:O.Data [0]	(4) The PLC copies the value at **2:I:Data[1]** here to acknowledge receipt of the response
Local:2:O.Data [1]	(1) The PLC increments this counter value after copying a command in Consume Data
Local:2:O.Data [2]	Consume Data Size
Local:2:O.Data [3]	First WORD of Consume Data (*Command from PLC*)
Local:2:O.Data [xxx]	xxx WORD of Consume Data

INPUT CONTROLLER TAG

Controller Tag Location and Data	Description
Local:2:I.Data [0]	(2) The value at **2:O:Data[1]** is copied here by the Gateway to acknowledge receipt of a command
Local:2:I.Data [1]	(3) The Gateway increments this counter to signal that a response is available
Local:2:I.Data [2]	Produce Data Size
Local:2:I.Data [3]	First WORD of Produce Data (*Response from Gateway*)
Local:2:I.Data [xxx]	xxx WORD of Produce Data



NOTE

A ladder logic example illustrating the implementation of this handshaking strategy can be downloaded from the technical support area of the Balluff website.

9 PROFIBUS INTERFACE



NOTE

For BIS Z-GW-001-PBS models.

9.1 PROFIBUS OVERVIEW

Profibus was created under German Government leadership in co-operation with automation manufacturers (Siemens) in 1989. Today it is commonly found in Process Control, large assembly and material handling machines. Just a single-cable which is able to wire multi-input sensor blocks, pneumatic valves, complex intelligent devices, smaller sub-networks, operator interfaces and many other devices.

9.2 PROFIBUS-DP

Basically Profibus is available in three different versions:

Profibus-DP (Decentralized Periphery)
Multiple masters are possible with Profibus-DP, in which case each slave device is assigned to one master. This means that multiple masters can read inputs from the device but only one master can write outputs to that device.

Profibus-FMS
It is a peer to peer messaging format, which allows masters to communicate with one another. Just as in Profibus-DP, up to 126 nodes are available and all can be masters if desired. FMS messages consume more overhead than DP messages.

Profibus-PA
PA protocol is the same as the latest Profibus-DP except that voltage and current levels are reduced to meet the requirements of intrinsic safety (Class I div. II) for the process industry.

The Profibus Gateway supports **Profibus-DP only**, since this version has been specifically designed for factory automation.

MAIN FEATURES:

Maximum Number of Nodes: 126

Distance: 100 m to 24 Km (with repeaters and fibre optic transmission)

Baud rate: 9600 to 12M bps

9.3 DATA EXCHANGE

The Master Profibus is usually a PLC (Siemens S7 or others) but it could be a PC based device as well. The Profibus Gateway is always Slave in the Profibus network.

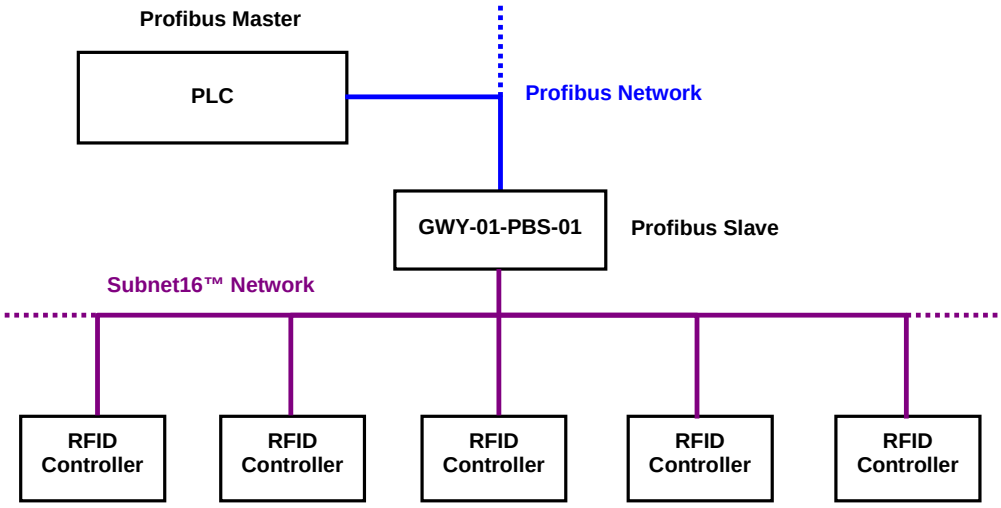


Figure 49 - Profibus-DP Network Diagram

Basically two shared memory areas (Exchange Areas) are used to exchange information between the SLAVE and the MASTER, both devices provide information to each other.

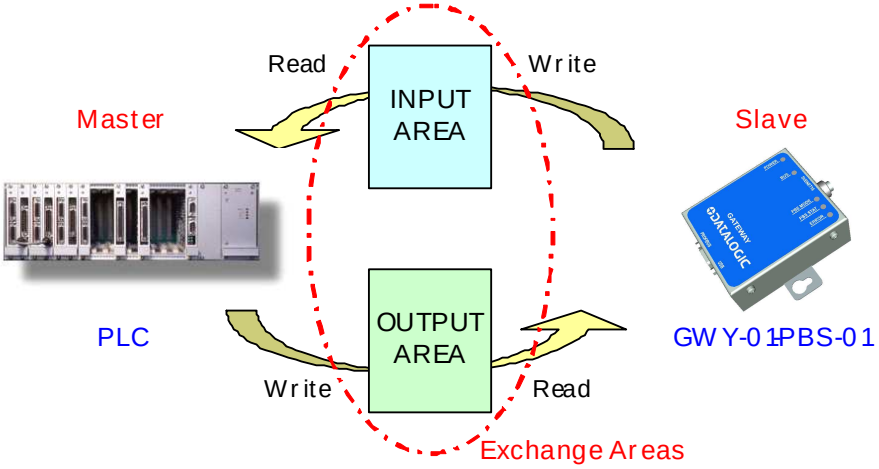


Figure 50 - Profibus Communication – Data Exchange Areas Diagram

Input and Output areas always refer to the Master: this means that the Gateway writes to the Input buffer and the PLC writes to the Output buffer.

The dimension of the exchange areas can be set to different values by the PLC through the GSD file: the Profibus Gateway allows up to **152 bytes as a combined total of the Input and Output Areas**.

NOTE

For further information regarding Fieldbus interfacing including downloadable support files, go to the HMS website at http://www.anybus.com, choose the link to the support page, select the Anybus-CompactCom product type *and then your* network type.

9.4 PROTOCOL IMPLEMENTATION

9.4.1 Definitions

In the protocol description we'll use the following definitions:

- Input field: is the set of master inputs that can be modified by the specific slave
- Output field: is the set of master outputs that can be read by the specific slave
- MaxInBytes: is the number of input bytes shared by the master and the specific slave
- MaxOutBytes: is the number of output bytes shared by the master and the specific slave
- IN[Nin] represent the input byte number Nin, where numbering starts from 0 to MaxInBytes-1
- OUT[Nout] represent the output byte number Nout, where numbering starts from 0 to MaxOutBytes-1

Obviously, MaxInBytes and MaxOutBytes are respectively the configured **INPUT** and **OUTPUT AREA** sizes.

The I/O Exchange Areas are actually updated and read every 30 ms at the Profibus Gateway side. So after an RFID tag is read, the worst delivery time from the Profibus Gateway to the Master station is about 30 ms plus the intrinsic PROFIBUS DP delay and the Master delay.

This product implements the Balluff AnyBus Driver which is a layer that is built upon the intrinsic fieldbus data exchange mechanism. The Driver is needed to add features such as flow control and fragmentation.

In order to implement the flow controlled version of the driver, I/O Exchange Areas must be congruently compiled in both directions. **INPUT** Area is the Exchange buffer from Profibus Gateway to the Master while **OUTPUT** Area is the exchange buffer from the Master to the Profibus Gateway. Only the first three bytes are used by the Balluff AnyBus Driver layer in both buffers for the extended protocol.

These are:

byte 0: **Control Field**, used to issue and control the Balluff AnyBus Driver primitives such as flowcontrol, fragmentation and resynchronization;

byte 1: **Service Access Point Field**, used to distinguish among different services but also to provide future expandability. Since this SAP definition is introduced by the Balluff AnyBus Driver, it must not be confused with the AnyBus SAP that is defined by the international standard.

byte 2: **Length Field**, that contains the number of bytes used by the application layer. This number must always be less than or equal to MaxInBytes-3 for the IN[ ] buffer and less than or equal to MaxOutBytes-3 for the OUT[ ] buffer.

Figure 51 - Exchange Area Buffer Structure

9.4.2 Control Field

The Input field structure reserves IN[0] for handshake purposes: bit 0 and bit 1 are used for this. Bit 6 is set to 1 in order to specify the messaging protocol number 1 is in use. The Output field structure is symmetrical, and reserves bit 0 and 1 for handshake purposes. Bit 6 is set to 1 in order to specify the messaging protocol number 1 is in use. Bit 2 of the Output buffer is used to request a clear of the synchronization numbers (bit 0 and bit 1 of both Input and Output buffers).

This is called a resynchronization request and it is always initiated by the Master Station. The Slave must acknowledge the request, using bit 2 of the Input buffer. Bit 3 is used to control a fragmentation sequence in both directions.

More precisely,

function of the IN[0] byte:

IN[0].bit0 = TxBufferFull, toggles when new data is available on IN[1] .. IN[Nin] input area

IN[0].bit1 = RxBufferEmpty, toggles when rx block has been read on OUT[1] .. OUT[Nout]

IN[0].bit2 = Resync Acknowledge, set to 1 as an acknowledge to a resync request.

IN[0].bit3 = More Bit, it must be set to 1 when this is not the last piece of a fragmentation sequence. It must be set to 0 when this is the last piece of a fragmentation sequence.

IN[0].bit4,5,7 = set to 0,0,0 when this messaging protocol is used.

IN[0].bit6 = set to 1 when this messaging protocol is used.

function of the OUT[0] byte:

OUT[0].bit0 = TxBufferEmpty, toggles when transmitted data block has been read from master.

OUT[0].bit1 = RxBufferFull, toggles when new data block is available from master.

OUT[0].bit2 = Resync Request, set to 1 for 1 second to resynchronize a slave. After

resynchronization, all 4 handshake bits are set to 0 and next toggle brings them to 1.

OUT[0].bit3 = More Bit, it must be set to 1 when this is not the last piece of a fragmentation sequence. It must be set to 0 when this is the last piece of a fragmentation sequence.

OUT[0].bit4,5,7 = set to 0,0,0 when this messaging protocol is used.

OUT[0].bit6 = set to 1 when this messaging protocol is used.

The following figure shows how it is possible to exchange messages with flow control using bit 0 and bit 1 in the IN/OUT buffers.

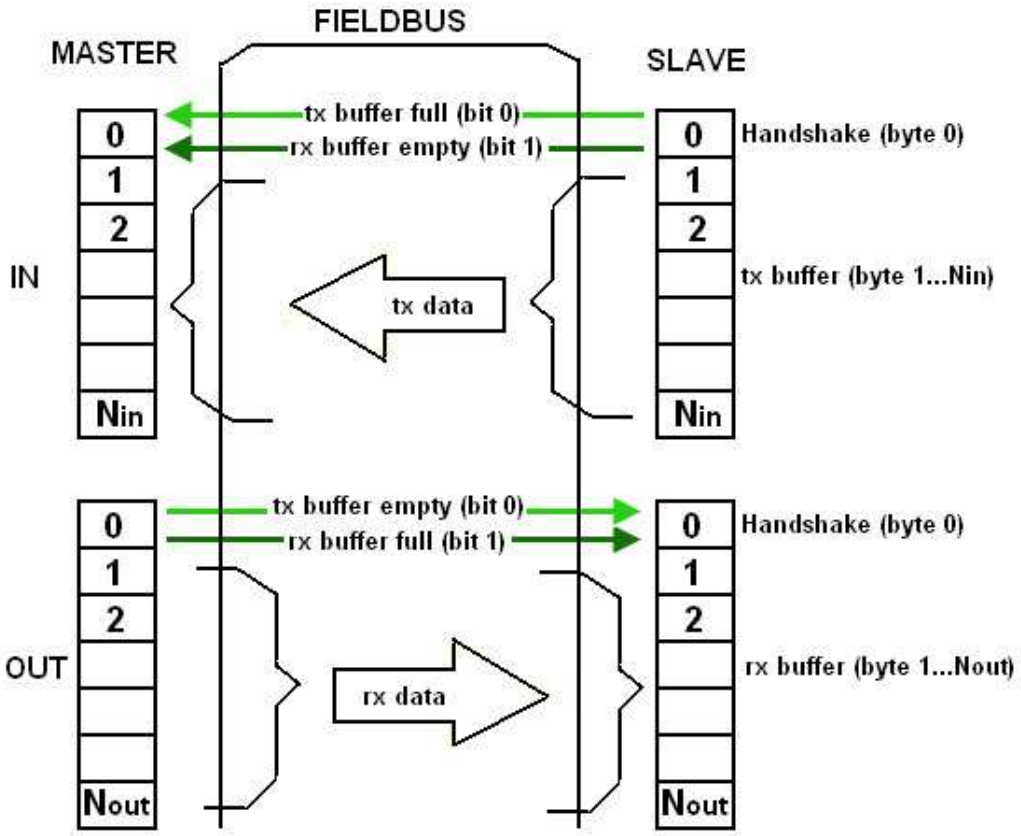


Figure 52 - Message Exchange with Flow Control

Data Transmission Slave → Master

The transmission state machine is shown to understand how a single block is transmitted and received. This protocol guarantees a basic flow control mechanism from slave to master.

Figure 53 - Slave to Master Transmission State Machine

Data Transmission Master → Slave

The receive state machine is shown to understand how a single block is transmitted by the master and received by a slave. This protocol guarantees a basic flow control mechanism from master to slave.

Figure 54 - Master to Slave Reception State Machine

Resynchronization Protocol

Resynchronization may be used at the master startup, both to detect if a slave is on line or not, or to restart the messaging protocol from a predefined state. It is also used during normal operations in case of errors requiring a protocol reset procedure to be started.

Figure 55 - Resynchronization State Machine

9.4.3 SAP Field

SAP (Service Access Point) is an identifier that is used to share the same communication channel between processes of two remote stations. This allows splitting the single service into different services.

SAP = 0 is actually used by the slave to transfer acquisition information; it should also be used to transfer application data from Master to Slave.

SAP = 2 is currently reserved.

SAP = 255 is currently reserved.

Only SAP 255 and 2 are reserved. All other SAPs are free and may be used by new application programs.

9.4.4 Length Field

The Application layer uses all or a part of the remaining bytes of the Exchange Area buffers that are not used by the Balluff AnyBus Driver. The Length Field is introduced to keep the information of how many bytes are really used by the Application Layer. A fragment that is not the last one of a fragmentation sequence must fill this field with Max(In/Out)Bytes-3, depending on whether it is an INPUT/OUTPUT fragment. Otherwise this field is filled with a number that is less than or equal to Max(In/Out)Bytes-3.

9.4.5 Application Data Buffer

The Application data buffer holds the CBx commands described in the CBx Command Protocol Reference Manual.

9.5 EXAMPLES OF PROFIBUS COMMAND/RESPONSE MECHANISM

As seen in par. 9.3, there are two buffers – an OUTPUT Buffer that is controlled by the MASTER, and an INPUT Buffer that is controlled by the slave (the Gateway).

The **OUTPUT Buffer** is mapped the following way:

Output Buffer	
Byte #	
00:	OUTPUT BUFFER CONTROL BYTE (OBCB)
01:	(Always 0)
02:	Packet Length in Bytes
03:	Packet Bytes (Command)
04:	" "
05:	" "
06:	" "
07:	" "
08:	" "
09:	" "
10:	" "
-	" "
-	" "
N-2:	" "
N-1:	Data Consistency Byte (OBDCB)

Byte 0 is the **Output Buffer Control Byte**. The Master uses the lowest two bits of this byte for handshaking: to signal that a command is ready for the slave (**Bit 1**), and to acknowledge receiving a response from the slave (**Bit 0**).

OUTPUT BUFFER CONTROL BYTE							
7	6	5	4	3	2	1	0
[1]	[0]	[0]	[0]	[0]	**[0]**	**[0]**	**[0]**

Bit 0 is toggled by the Master to acknowledge a packet (response) from the Gateway.

Bit 1 is toggled by the Master when it has a packet (command) ready for the Gateway.

Bit 2 is set when the Master wishes to initiate a "Resynchronization" with the Slave, and then cleared when it sees the corresponding handshake from the Slave, (indicating that the resynchronization is complete).

Bit 3 is set by the Slave when the total CBx response being returned to the Master is larger than the space available in the Input Buffer (or that the packet being returned is a fragment, and that there are more fragments to follow). This bit is cleared for the final fragment of a fragmented response – and so the Master can know when all the fragments of a response have been returned from the Slave.

Bit 7 is always 1, to conform to Balluff's proprietary Protocol.

Byte 1: is always 0.

Byte 2: contains the length of the packet in bytes (CBx Command or Command Fragment) to be sent to the Gateway. This can be the length of an entire CBx command, or the length of a fragment of a command, if the CBx command is larger than the space allowed to send it in a single fragment.

Byte 3 through Byte N-2 are used for the actual CBx Command or Command Fragment to be sent.

Byte N-1: the final byte of the Output Buffer is the **Data Consistency Byte.** It is a copy of the **Ouptut Buffer Control Byte.** When changes to the Control Byte are made, the same changes must also be made in the Data Consistency Byte, before the changes "take effect". This is to guarantee the validity of the data between the two bytes.

The **INPUT Buffer** is controlled by the Slave (Gateway) and is mapped the same way, except for the packet bytes containing a response (or response fragment) from the Gateway.

Input Buffer	
Byte #	
00:	INPUT BUFFER CONTROL BYTE (IBCB)
01:	(Always 0)
02:	Packet Length in Bytes
03:	Packet Bytes (Response)
04:	" "
05:	" "
06:	" "
07:	" "
08:	" "
09:	" "
10:	" "
-	" "
-	" "
N-2:	" "
N-1:	Data Consistency Byte (IBDCB)

Byte 0 is the **Input Buffer Control Byte.** The Slave uses the lowest four bits of this byte for handshaking: to acknowledge receiving a command from the master (Bit 1), and to signal that a response is ready for the master (Bit 0).

INPUT BUFFER CONTROL BYTE							
7	6	5	4	3	2	1	0
[1]	[0]	[0]	[0]	[0]	[0]	[0]	[0]

Bit 0 is toggled by the Slave when it has a new packet (response or response fragment) ready for the Master.

Bit 1 is toggled by the Slave to acknowledge a packet (command or command fragment) from the Master.

Bit 2 is set by the Slave after it completes resynchronization, and then cleared once the Master has acknowledged that resynchronization is complete.

Bit 3 is set by the Slave when the total CBx response being returned to the Master is larger than the space available in the Input Buffer (or that the packet being returned is a fragment, and that there are more fragments to follow). This bit is cleared for the final fragment of a fragmented response – and so the Master can know when all the fragments of a response have been returned from the Slave.

Bit 7 is set to 1 as soon as the Slave has been successfully initialized at power-up, and remains at 1, to conform to Balluff's proprietary Protocol.

Byte 1: is always 0.

Byte 2: contains the length of the packet in bytes (CBx response or response fragment) to be sent back to the Master.

Byte 3 through Byte N-2 are used for the actual CBx response or response fragment to be sent.

Byte N-1: The final byte of the Input Buffer is the Data Consistency Byte for the Input Buffer. It is a copy of the Input Buffer Control Byte. The Master should check that these two bytes are the same, before considering the Input Buffer's data to be valid.



NOTE

The input and output buffers can exceed 64 bytes. The combined total of the input and output buffers cannot exceed 152 bytes.

9.5.1 Example 1: Normal Command/Response Sequence

For this example, the Master will send a CBx "Read Tag ID" command to the Slave (the RFID Controller at Node 01 of the Subnet16™ Profibus Gateway network) to read an 8-byte tag ID from an RFID Tag. First we will see a "Tag Not Found" error (assuming that the tag is not read) and then we will see a successful read of the Tag ID.

We will assume for this example that both the Input and Output Buffers have been configured to 32 bytes each. This means that the controllers response (for this command) can fit entirely in the input buffer, and no fragmentation is required.

Sending the command:

In **Byte 2** of the output buffer the Master places the length (in bytes) of the data packet (CBx Command) we are sending. In this case the CBx command we are sending is 12 bytes. This length is the length of the command bytes we are interested in sending, not the full size of the buffer. The length also does not include the "Data Consistency Byte" at the end of the buffer. That is just a mirror of the Control Byte.

In Byte 3 through **Byte 14** the Master places the 12 bytes of this particular CBx command. Some CBx commands are larger, but all will be at least 12 bytes, even if some of those 12 bytes are not actually used.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	80	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	**0C**	**(Packet length in bytes)**	02:	00	
03: 04: 05: 06: 07: 08: 09: 10: 11: 12: 13: 14: 15: 16: 17: 18: 19: 20: .. 30:	**00** **06** **AA** **07** **00** **01** **03** **E8** **00** **00** **00** **00** 00 00 00 00 00 00 .. 00	**(CBx Command word length MSB)** **(CBx Command word length LSB)** Minimum of 6 words **(CBx Command Type)** Always AA **(CBx Command Opcode)** 0x07 = Read Tag ID **(CBx Command, byte not used)** **(CBx Command "Node ID")** **(CBx Command Timeout MSB)** **(CBx Command Timeout LSB)** 0xE8 = 1000 ms timeout **(CBx Command Not Used)** **(CBx Command Not Used)** **(CBx Command Not Used)** **(CBx Command Not Used)**	03: 04: 05: 06: 07: 08: 09: 10: 11: 12: 13: 14: 15: 16: 17: 18: 19: 20: .. 30:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .. 00	
31:	80	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

Now that the command is in the Output Buffer, The Master alerts the Slave that the command is ready. It does this by toggling **Bit 1** of the **Output Buffer Control Byte** (the **OBCB**) and then also toggling the same bit in the **Output Buffer Data Consistence Byte** (the **OBDCB**).



NOTE

This bit is a toggle. So if it is 0, it is toggled to 1 to indicate a new command. If it is 1, it is toggled to 0 to indicate a new command. If the bit is 1, setting it to 0, and then back to 1 will cause the command to be issued twice.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	**82**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1]** [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	0C	(Packet length in bytes)	02:	00	
03:	00	(CBx Command word length MSB)	03:	00	
04:	06	(CBx Command word length LSB)	04:	00	
05:	AA	(CBx Command Type)	05:	00	
06:	07	(CBx Command Opcode)	06:	00	
07:	00	(CBx Command, byte not used)	07:	00	
08:	01	(CBx Command “Node ID”)	08:	00	
09:	03	(CBx Command Timeout MSB)	09:	00	
10:	E8	(CBx Command Timeout LSB)	10:	00	
11:	00	(CBx Command Not Used)	11:	00	
12:	00	(CBx Command Not Used)	12:	00	
13:	00	(CBx Command Not Used)	13:	00	
14:	00	(CBx Command Not Used)	14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..	..		..	..	
30:	00		30:	00	
31:	**82**	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

When the Slave sees Bit 1 of the **OBCB & OBDBC** toggle, it grabs the command from the **Output Buffer.** The Slave then acknowledges the command by toggling **Bit 1** of the **Input Buffer Control Byte** (the **IBCB**) and also the same bit of the **Input Buffer Data Consistency Byte** (the **IBDCB**).

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	82	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	**82**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1]** [0]
01:	00	(Always 0)	01:	00	
02:	0C	(Packet length in bytes)	02:	00	
03:	00	(CBx Command word length MSB)	03:	00	
04:	06	(CBx Command word length LSB)	04:	00	
05:	AA	(CBx Command Type)	05:	00	
06:	07	(CBx Command Opcode)	06:	00	
07:	00	(CBx Command, byte not used)	07:	00	
08:	01	(CBx Command “Node ID”)	08:	00	
09:	03	(CBx Command Timeout MSB)	09:	00	
10:	E8	(CBx Command Timeout LSB)	10:	00	
11:	00	(CBx Command Not Used)	11:	00	
12:	00	(CBx Command Not Used)	12:	00	
13:	00	(CBx Command Not Used)	13:	00	
14:	00	(CBx Command Not Used)	14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..	..		..	..	
30:	00		30:	00	
31:	82	**Data Consistency Byte (OBDCB)**	31:	**82**	**Data Consistency Byte (IBDCB)**

The Slave writes the response into the **Input Buffe**r, and toggles **Bit 0** of the **IBCB** to indicate that there is a response fragment ready for the master. Since the entire response fits in the buffer, it does not need to use fragmentation. The Slave also simultaneously makes the same changes to the **IBDCB**.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	82	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	**83**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1] [1]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	**0E**	**(Packet length in bytes)**
03:	00	(CBx Command word length MSB)	03:	**00**	**(CBx Response word length MSB)**
04:	06	(CBx Command word length LSB)	04:	**07**	**(CBx Response word length LSB)**
05:	AA	(CBx Command Type)			Minimum of 6 words
06:	07	(CBx Command Opcode)	05:	**FF**	**(CBx Response Type) FF=Error**
07:	00	(CBx Command, byte not used)	06:	**FF**	**(CBx Response Opcode) FF=Error**
08:	01	(CBx Command “Node ID”)	07:	**00**	**(CBx Response Instance Counter)**
09:	03	(CBx Command Timeout MSB)	08:	**01**	**(CBx Response “Node ID”)**
10:	E8	(CBx Command Timeout LSB)	09:	**01**	**(CBx Response Timestamp Month)**
11:	00	(CBx Command Not Used)	10:	**01**	**(CBx Response Timestamp Day)**
12:	00	(CBx Command Not Used)	11:	**00**	**(CBx Response Timestamp Hour)**
13:	00	(CBx Command Not Used)	12:	**13**	**(CBx Response Timestamp Minute)**
14:	00	(CBx Command Not Used)	13:	**22**	**(CBx Response Timestamp Second)**
15:	00		14:	**01**	**(CBx Response “Data length”)**
16:	00				1 byte (the Error Code)
17:	00		15:	**07**	**(CBx Response Data Byte 1)**
18:	00				Error Code 7 = Tag Not Found
19:	00		16:	**00**	**(CBx Response byte not used)**
20:	00		17:	00	
..	..		18:	00	
30:	00		19:	00	
			20:	00	
			..	..	
			30:	00	
31:	82	**Data Consistency Byte (OBDCB)**	31:	**83**	**Data Consistency Byte (IBDCB)**

In this case, the response is a "Tag Not Found" error.

The Master can see that **Bit 0** of the **IBCB & IBDCB** has been toggled, so it knows that the response in the Input Buffer is ready. Since **Bit 2** of the **IBCB & IBDCB** is **not** set to 1, it knows that the response is complete (not a fragment).

The Master now toggles **Bit 0** of the **OBCB & OBDCB** to acknowledge that it has received the response.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	**83**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] **[1]**	00	83	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [1]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	0E	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	00	(CBx Response word length MSB)
04:	06	(CBx Command word length LSB)	04:	07	(CBx Response word length LSB) Minimum of 6 words
05:	AA	(CBx Command Type)	05:	FF	(CBx Response Type) FF=Error
06:	07	(CBx Command Opcode)	06:	FF	(CBx Response Opcode) FF=Error
07:	00	(CBx Command, byte not used)	07:	00	(CBx Response Instance Counter)
08:	01	(CBx Command “Node ID”)	08:	01	(CBx Response “Node ID”)
09:	03	(CBx Command Timeout MSB)	09:	01	(CBx Response Timestamp Month)
10:	E8	(CBx Command Timeout LSB)	10:	01	(CBx Response Timestamp Day)
11:	00	(CBx Command Not Used)	11:	00	(CBx Response Timestamp Hour)
12:	00	(CBx Command Not Used)	12:	13	(CBx Response Timestamp Minute)
13:	00	(CBx Command Not Used)	13:	22	(CBx Response Timestamp Second)
14:	00	(CBx Command Not Used)	14:	01	(CBx Response “Data length”) 1 byte (the Error Code)
15:	00		15:	07	(CBx Response Data Byte 1) Error Code 7 = Tag Not Found
16:	00		16:	00	(CBx Response byte not used)
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..	..		..	..	
30:	00		30:	00	
31:	**83**	**Data Consistency Byte (OBDCB)**	31:	83	**Data Consistency Byte (IBDCB)**

The command/response sequence has completed. A command has been issued and the response received (in this case, a “Tag Not Found” error) and the response has been acknowledged.

If we now place a tag on the controller's antenna (Node 01), we can reissue the same command by toggling **Bit 1** of the **OBCB & OBDCB** again.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**81**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[0]** [1]	00	83	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [1]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	0E	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	00	(CBx Response word length MSB)
04:	06	(CBx Command word length LSB)	04:	07	(CBx Response word length LSB) Minimum of 6 words
05:	AA	(CBx Command Type)	05:	FF	(CBx Response Type) FF=Error
06:	07	(CBx Command Opcode)	06:	FF	(CBx Response Opcode) FF=Error
07:	00	(CBx Command, byte not used)	07:	00	(CBx Response Instance Counter)
08:	01	(CBx Command "Node ID")	08:	01	(CBx Response "Node ID")
09:	03	(CBx Command Timeout MSB)	09:	01	(CBx Response Timestamp Month)
10:	E8	(CBx Command Timeout LSB)	10:	01	(CBx Response Timestamp Day)
11:	00	(CBx Command Not Used)	11:	00	(CBx Response Timestamp Hour)
12:	00	(CBx Command Not Used)	12:	13	(CBx Response Timestamp Minute)
13:	00	(CBx Command Not Used)	13:	22	(CBx Response Timestamp Second)
14:	00	(CBx Command Not Used)	14:	01	(CBx Response "Data length") 1 byte (the Error Code)
15:	00		15:	07	(CBx Response Data Byte 1) Error Code 7 = Tag Not Found
16:	00		16:	00	(CBx Response byte not used)
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..	..		..	..	
30:	00		30:	00	
31:	**81**	**Data Consistency Byte (OBDCB)**	31:	83	**Data Consistency Byte (IBDCB)**

The Gateway will toggle **Bit 1** of the **IBCB & IBDCB** to indicate it has received the command.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	81	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [1]	00	**81**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[0]** [1]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	0E	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	00	(CBx Response word length MSB)
04:	06	(CBx Command word length LSB)	04:	07	(CBx Response word length LSB) Minimum of 6 words
05:	AA	(CBx Command Type)	05:	FF	(CBx Response Type) FF=Error
06:	07	(CBx Command Opcode)	06:	FF	(CBx Response Opcode) FF=Error
07:	00	(CBx Command, byte not used)	07:	00	(CBx Response Instance Counter)
08:	01	(CBx Command “Node ID”)	08:	01	(CBx Response “Node ID”)
09:	03	(CBx Command Timeout MSB)	09:	01	(CBx Response Timestamp Month)
10:	E8	(CBx Command Timeout LSB)	10:	01	(CBx Response Timestamp Day)
11:	00	(CBx Command Not Used)	11:	00	(CBx Response Timestamp Hour)
12:	00	(CBx Command Not Used)	12:	13	(CBx Response Timestamp Minute)
13:	00	(CBx Command Not Used)	13:	22	(CBx Response Timestamp Second)
14:	00	(CBx Command Not Used)	14:	01	(CBx Response “Data length”) 1 byte (the Error Code)
15:	00		15:	07	(CBx Response Data Byte 1) Error Code 7 = (Tag Not Found)
16:	00		16:	00	(CBx Response byte not used)
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..	..		..	..	
30:	00		30:	00	
31:	81	**Data Consistency Byte (OBDCB)**	31:	**81**	**Data Consistency Byte (IBDCB)**

We will assume that the controller successfully reads the RFID tag.

The Slave (Gateway) writes the response into the Input Buffer, and toggles **Bit 0** of the **IBCB & IBDCB** to indicate that the response is ready.



NOTE

If the master has not acknowledged receiving the previous response, the Gateway will not be able to place the response in the Input Buffer.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	81	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [1]	00	**80**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] **[0]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	**14**	**(Packet length in bytes)**
03:	00	(CBx Command word length MSB)	03:	**00**	**(CBx Response word length MSB)**
04:	06	(CBx Command word length LSB)	04:	**0A**	**(CBx Response word length LSB)**
05:	AA	(CBx Command Type)			Minimum of 6 words
06:	07	(CBx Command Opcode)	05:	**AA**	**(CBx Response Type)**
07:	00	(CBx Command, byte not used)			AA=Normal Response
08:	01	(CBx Command "Node ID")	06:	**07**	**(CBx Response Opcode)**
09:	03	(CBx Command Timeout MSB)			07=Command Echo of Tag Read ID
10:	E8	(CBx Command Timeout LSB)	07:	**01**	**(CBx Response Instance Counter)**
11:	00	(CBx Command Not Used)	08:	**01**	**(CBx Response "Node ID")**
12:	00	(CBx Command Not Used)	09:	**01**	**(CBx Response Timestamp Month)**
13:	00	(CBx Command Not Used)	10:	**01**	**(CBx Response Timestamp Day)**
14:	00	(CBx Command Not Used)	11:	**01**	**(CBx Response Timestamp Hour)**
15:	00		12:	**17**	**(CBx Response Timestamp Minute)**
16:	00		13:	**58**	**(CBx Response Timestamp Second)**
17:	00		14:	**08**	**(CBx Response "Data length")**
18:	00				8 bytes (the Tag ID)
19:	00		15:	**E0**	**(CBx Response Data Byte 1)**
20:	00				Tag ID Byte 1
..	..		16:	**04**	**(CBx Response Data Byte 2)**
30:	00				Tag ID Byte 2
			17:	**01**	**(CBx Response Data Byte 3)**
					Tag ID Byte 3
			18:	**00**	**(CBx Response Data Byte 4)**
					Tag ID Byte 4
			19:	**0E**	**(CBx Response Data Byte 5)**
					Tag ID Byte 5
			20:	**20**	**(CBx Response Data Byte 6)**
					Tag ID Byte 6
			21:	**DD**	**(CBx Response Data Byte 7)**
					Tag ID Byte 7
			22:	**AF**	**(CBx Response Data Byte 8)**
					Tag ID Byte 8
			..	..	
			30:	00	
31:	81	**Data Consistency Byte (OBDCB)**	31:	**80**	**Data Consistency Byte (IBDCB)**

You can see the Tag ID in the data portion of the CBx response, Tag ID **E00401000E20DDAF**.

The Master can see that **Bit 0** of the **IBCB & IBDCB** has been toggled, so it knows that the response in the Input Buffer is ready. Since **Bit 2** is not set to 1, it knows that the response is complete (not a fragment).

The Master now toggles **Bit 0** of the **OBCB** & **OBDCB** to acknowledge that it has received the response.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	80	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] **[0]**	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	14	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	00	(CBx Response word length MSB)
04:	06	(CBx Command word length LSB)	04:	0A	(CBx Response word length LSB) Minimum of 6 words
05:	AA	(CBx Command Type)	05:	AA	(CBx Response Type) AA=Normal Response
06:	07	(CBx Command Opcode)	06:	07	(CBx Response Opcode) 07=Command Echo of Tag Read ID
07:	00	(CBx Command, byte not used)	07:	01	(CBx Response Instance Counter)
08:	01	(CBx Command “Node ID”)	08:	01	(CBx Response “Node ID”)
09:	03	(CBx Command Timeout MSB)	09:	01	(CBx Response Timestamp Month)
10:	E8	(CBx Command Timeout LSB)	10:	01	(CBx Response Timestamp Day)
11:	00	(CBx Command Not Used)	11:	01	(CBx Response Timestamp Hour)
12:	00	(CBx Command Not Used)	12:	17	(CBx Response Timestamp Minute)
13:	00	(CBx Command Not Used)	13:	58	(CBx Response Timestamp Second)
14:	00	(CBx Command Not Used)	14:	08	(CBx Response “Data length”) 8 bytes (the Tag ID)
15:	00		15:	E0	(CBx Response Data Byte 1) Tag ID Byte 1
16:	00		16:	04	(CBx Response Data Byte 2) Tag ID Byte 2
17:	00		17:	01	(CBx Response Data Byte 3) Tag ID Byte 3
18:	00		18:	00	(CBx Response Data Byte 4) Tag ID Byte 4
19:	00		19:	0E	(CBx Response Data Byte 5) Tag ID Byte 5
20:	00		20:	20	(CBx Response Data Byte 6) Tag ID Byte 6
..	..		21:	DD	(CBx Response Data Byte 7) Tag ID Byte 7
30:	00		22:	AF	(CBx Response Data Byte 8) Tag ID Byte 8
			..	..	
			30:	00	
31:	80	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

The command/response sequence has completed. A command has been issued and the response received (in this case, a successful read of the RFID Tag ID) and the response has been acknowledged.

9.5.2 Example 2: Unsolicited Responses (Continuous Read Mode)

In some modes (such as Continuous Read Mode) the slave can generate unsolicited responses. If the Slave generates an unsolicited response, it will place the response in the Input Buffer, as long as the Master has acknowledged receiving the previous response. If the Master does not perform the handshake to acknowledge the previous response, the responses will accumulate in the internal memory buffer of the Slave (The RFID controller has an internal **2K buffer** for responses) and the responses will remain until the handshakes are performed for each response.

For this example, the controller automatically reads a tag (6 bytes of data), and places the "response" in the Input Buffer, and toggles **Bit 0** to indicate that a response is waiting.

Although no command was issued by the Master, we will still call this a "response".

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
		Output Buffer Control Byte (OBCB)			**Input Buffer Control Byte (IBCB)**
00:	80	7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]	00	**81**	7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] **[1]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	00	(Packet length in bytes)	02:	**12**	**(Packet length in bytes)**
03:	00		03:	**00**	**(CBx Response word length MSB)**
04:	00		04:	**09**	**(CBx Response word length LSB)** Minimum of 6 words
05:	00		05:	**AA**	**(CBx Response Type)** AA=Normal Response
06:	00		06:	**0D**	**(CBx Response Opcode)** 0D=Continuous Read Response
07:	00		07:	**01**	**(CBx Response Instance Counter)**
08:	00		08:	**01**	**(CBx Response "Node ID")**
09:	00		09:	**01**	**(CBx Response Timestamp Month)**
10:	00		10:	**01**	**(CBx Response Timestamp Day)**
11:	00		11:	**02**	**(CBx Response Timestamp Hour)**
12:	00		12:	**12**	**(CBx Response Timestamp Minute)**
13:	00		13:	**34**	**(CBx Response Timestamp Second)**
14:	00		14:	**06**	**(CBx Response "Data length")** 6 bytes (the Tag ID)
15:	00		15:	**11**	**(CBx Response Data Byte 1)** Tag ID Byte 1
16:	00		16:	**22**	**(CBx Response Data Byte 2)** Tag ID Byte 2
17:	00		17:	**33**	**(CBx Response Data Byte 3)** Tag ID Byte 3
18:	00		18:	**44**	**(CBx Response Data Byte 4)** Tag ID Byte 4
19:	00		19:	**55**	**(CBx Response Data Byte 5)** Tag ID Byte 5
20:	00		20:	**66**	**(CBx Response Data Byte 6)** Tag ID Byte 6
..	..		..	..	
30:	00		30:	00	
31:	80	**Data Consistency Byte (OBDCB)**	31:	**81**	**Data Consistency Byte (IBDCB)**

The Master can see that **Bit 0** of the **IBCB & IBDCB** has been toggled, so it knows that a new response in the Input Buffer is ready (even though it hasn't issued a command).

Since **Bit 2** is not set to 1, it knows that the response is complete (not a fragment).

The Master now toggles **Bit 0** of the **OBCB** & **OBDCB** to acknowledge that it has received the response.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**81**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] **[1]**	00	81	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [1]
01:	00	(Always 0)	01:	00	(Always 0)
02:	00	(Packet length in bytes)	02:	12	(Packet length in bytes)
03:	00		03:	00	(CBx Response word length MSB)
04:	00		04:	09	(CBx Response word length LSB) Minimum of 6 words
05:	00		05:	AA	(CBx Response Type) AA=Normal Response
06:	00		06:	0D	(CBx Response Opcode) 0D=Continuous Read Response
07:	00		07:	01	(CBx Response Instance Counter)
08:	00		08:	01	(CBx Response "Node ID")
09:	00		09:	01	(CBx Response Timestamp Month)
10:	00		10:	01	(CBx Response Timestamp Day)
11:	00		11:	02	(CBx Response Timestamp Hour)
12:	00		12:	12	(CBx Response Timestamp Minute)
13:	00		13:	34	(CBx Response Timestamp Second)
14:	00		14:	06	(CBx Response "Data length") 6 bytes (the Tag ID)
15:	00		15:	11	(CBx Response Data Byte 1) Tag ID Byte 1
16:	00		16:	22	(CBx Response Data Byte 2) Tag ID Byte 2
17:	00		17:	33	(CBx Response Data Byte 3) Tag ID Byte 3
18:	00		18:	44	(CBx Response Data Byte 4) Tag ID Byte 4
19:	00		19:	55	(CBx Response Data Byte 5) Tag ID Byte 5
20:	00		20:	66	(CBx Response Data Byte 6) Tag ID Byte 6
..	..		..	..	
30:	00		30:	00	
31:	**81**	**Data Consistency Byte (OBDCB)**	31:	81	**Data Consistency Byte (IBDCB)**

The response has been acknowledged (a read of 6 bytes: 11 22 33 44 55 66).

The reader then reads another tag, puts another response in the Input Buffer, and toggles **Bit 0** again in the **IBCB & IBDCB**.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	81	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [1]	00	**80**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] **[0]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	00	(Packet length in bytes)	02:	**12**	**(Packet length in bytes)**
03:	00		03:	**00**	**(CBx Response word length MSB)**
04:	00		04:	**09**	**(CBx Response word length LSB)**
05:	00				Minimum of 6 words
06:	00		05:	**AA**	**(CBx Response Type)**
07:	00				AA=Normal Response
08:	00		06:	**0D**	**(CBx Response Opcode)**
09:	00				0D=Continuous Read Response
10:	00		07:	**02**	**(CBx Response Instance Counter)**
11:	00		08:	**01**	**(CBx Response "Node ID")**
12:	00		09:	**01**	**(CBx Response Timestamp Month)**
13:	00		10:	**01**	**(CBx Response Timestamp Day)**
14:	00		11:	**02**	**(CBx Response Timestamp Hour)**
15:	00		12:	**13**	**(CBx Response Timestamp Minute)**
16:	00		13:	**34**	**(CBx Response Timestamp Second)**
17:	00		14:	**06**	**(CBx Response "Data length")**
18:	00				6 bytes (the Tag ID)
19:	00		15:	**77**	**(CBx Response Data Byte 1)**
20:	00				Tag ID Byte 1
..	..		16:	**88**	**(CBx Response Data Byte 2)**
30:	00				Tag ID Byte 2
			17:	**99**	**(CBx Response Data Byte 3)**
					Tag ID Byte 3
			18:	**AA**	**(CBx Response Data Byte 4)**
					Tag ID Byte 4
			19:	**BB**	**(CBx Response Data Byte 5)**
					Tag ID Byte 5
			20:	**CC**	**(CBx Response Data Byte 6)**
					Tag ID Byte 6
			..	..	
			30:	00	
31:	81	**Data Consistency Byte (OBDCB)**	31:	**80**	**Data Consistency Byte (IBDCB)**

This response contains a timestamp that is 60 seconds after the previous response, and tag has different data.

Note that the "Instance Counter" in the CBx response increments for each response.

The Master can see that **Bit 0** of the **IBCB & IBDCB** has been toggled, so it knows that a new response in the Input Buffer is ready.

The Master now toggles **Bit 0** of the **OBCB** & **OBDCB** to acknowledge that it has received the response.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**80**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] **[0]**	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	(Always 0)
02:	00	(Packet length in bytes)	02:	12	(Packet length in bytes)
03:	00		03:	00	(CBx Response word length MSB)
04:	00		04:	09	(CBx Response word length LSB) Minimum of 6 words
05:	00		05:	AA	(CBx Response Type) AA=Normal Response
06:	00		06:	0D	(CBx Response Opcode) 0D=Continuous Read Response
07:	00		07:	02	(CBx Response Instance Counter)
08:	00		08:	01	(CBx Response “Node ID”)
09:	00		09:	01	(CBx Response Timestamp Month)
10:	00		10:	01	(CBx Response Timestamp Day)
11:	00		11:	02	(CBx Response Timestamp Hour)
12:	00		12:	13	(CBx Response Timestamp Minute)
13:	00		13:	34	(CBx Response Timestamp Second)
14:	00		14:	06	(CBx Response “Data length”) 6 bytes (the Tag ID)
15:	00		15:	77	(CBx Response Data Byte 1) Tag ID Byte 1
16:	00		16:	88	(CBx Response Data Byte 2) Tag ID Byte 2
17:	00		17:	99	(CBx Response Data Byte 3) Tag ID Byte 3
18:	00		18:	AA	(CBx Response Data Byte 4) Tag ID Byte 4
19:	00		19:	BB	(CBx Response Data Byte 5) Tag ID Byte 5
20:	00		20:	CC	(CBx Response Data Byte 6) Tag ID Byte 6
..	..		..	..	
30:	00		30:	00	
31:	**80**	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

No new responses will come from the reader until the Master has acknowledged the previous response by toggling **Bit 0** of the **OBCB & OBDCB**.

9.5.3 Example 3: Fragmentation of Responses

For this example, the Master will send a CBx “Read Tag Data” command to the Slave (the RFID Controller at Node 01 of the Subnet16™ Profibus Gateway network) to read 50 bytes from a tag.

We will assume for this example that both the input and output buffers have been configured to 32 bytes each. This means that the controller's response to the tag read command cannot completely fit in the input buffer, and the response will be “**fragmented**” or sent in multiple fragments.

Sending the command:

In **Byte 2** of the output buffer, the Master places the length (in bytes) of the data packet (CBx Command) we are sending. In this case the CBx command we are sending is 12 bytes. This length is the length of the command bytes we are interested in sending, not the full size of the buffer. The length also does not include the “Data Consistency Byte” at the end of the buffer. That is just a mirror of the Control Byte.

In Byte 3 through **Byte 14** the Master places the 12 bytes of this particular CBx command. Some CBx commands are larger, but all will be at least 12 bytes, even if some of those 12 bytes are not actually used.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	80	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	**0C**	**(Packet length in bytes)**	02:	00	
03:	**00**	**(CBx Command word length MSB)**	03:	00	
04:	**06**	**(CBx Command word length LSB)** Minimum of 6 words	04: 05:	00 00	
05:	**AA**	**(CBx Command Type)** Always AA	06:	00	
06:	**05**	**(CBx Command Opcode)** 0x05 = Read Tag Data	07: 08:	00 00	
07:	**00**	**(CBx Command, byte not used)**	09:	00	
08:	**01**	**(CBx Command “Node ID”)**	10:	00	
09:	**03**	**(CBx Command Timeout MSB)**	11:	00	
10:	**E8**	**(CBx Command Timeout LSB)** 0xE8 = 1000 ms timeout	12: 13:	00 00	
11:	**00**	**(CBx Command Start Address MSB)**	14:	00	
12:	**00**	**(CBx Command Start Address LSB)** address 0	15: 16:	00 00	
13:	**00**	**(CBx Command Length MSB)**	17:	00	
14:	**32**	**(CBx Command Length LSB)** 50 bytes	18: 19:	00 00	
15:	00		20:	00	
16:	00		..	..	
17:	00		30:	00	
18:	00				
19:	00				
20:	00				
..	..				
30:	00				
31:	80	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

Now that the command is in the Output Buffer, The Master alerts the Slave that the command is ready. It does this by toggling **Bit 1** of the **Output Buffer Control Byte** (the **OBCB**) and then also toggling the same bit in the **Output Buffer Data Consistence Byte** (the **OBDCB**)



NOTE

This bit is a toggle. So if it is 0, it is toggled to 1 to indicate a new command. If it is 1, it is toggled to 0 to indicate a new command. If the bit is 1, setting it to 0, and then back to 1 will cause the command to be issued twice.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**82**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1]** [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	0C	(Packet length in bytes)	02:	00	
03:	00	(CBx Command word length MSB)	03:	00	
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	00	
			05:	00	
05:	AA	(CBx Command Type) Always AA	06:	00	
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	07:	00	
			08:	00	
07:	00	(CBx Command, byte not used)	09:	00	
08:	01	(CBx Command “Node ID”)	10:	00	
09:	03	(CBx Command Timeout MSB)	11:	00	
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	12:	00	
			13:	00	
11:	00	(CBx Command Start Address MSB)	14:	00	
12:	00	(CBx Command Start Address LSB) address 0	15:	00	
			16:	00	
13:	00	(CBx Command Length MSB)	17:	00	
14:	32	(CBx Command Length LSB) 50 bytes	18:	00	
			19:	00	
15:	00		20:	00	
16:	00		..	..	
17:	00		30:	00	
18:	00				
19:	00				
20:	00				
..	..				
30:	00				
31:	**82**	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

When the Slave sees Bit 1 of the **OBCB & OBDBC** toggle, it grabs the command from the **Output Buffer.** The Slave then acknowledges the command by toggling **Bit 1** of the **Input Buffer Control Byte** (the **IBCB**) and also the same bit of the **Input Buffer Data Consistency Byte** (the **IBDCB**).

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	82	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	**82**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1]** [0]
01:	00	(Always 0)	01:	00	
02:	0C	(Packet length in bytes)	02:	00	
03:	00	(CBx Command word length MSB)	03:	00	
04:	06	(CBx Command word length LSB) Minimum of 6 words	04: 05:	00 00	
05:	AA	(CBx Command Type) Always AA	06:	00	
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	07: 08:	00 00	
07:	00	(CBx Command, byte not used)	09:	00	
08:	01	(CBx Command “Node ID”)	10:	00	
09:	03	(CBx Command Timeout MSB)	11:	00	
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	12: 13:	00 00	
11:	00	(CBx Command Start Address MSB)	14:	00	
12:	00	(CBx Command Start Address LSB) address 0	15: 16:	00 00	
13:	00	(CBx Command Length MSB)	17:	00	
14:	32	(CBx Command Length LSB) 50 bytes	18: 19:	00 00	
15:	00		20:	00	
16:	00		..	..	
17:	00		30:	00	
18:	00				
19:	00				
20:	00				
..	..				
30:	00				
31:	82	**Data Consistency Byte (OBDCB)**	31:	**82**	**Data Consistency Byte (IBDCB)**

The Slave writes the first fragment of the response into the **Input Buffe**r, and toggles **Bit 0** of the **IBCB** to indicate that there is a response fragment ready for the master, and sets **Bit 3** of the **IBCB to 1** to indicate that this is a fragment of a longer response (i.e. there is more data remaining) The Slave also simultaneously makes the same changes to the **IBDCB**.



NOTE

Bit 3 is not a toggle – If it is 1, then there are more fragments to follow. If it is 0, it is either a complete response, or the final fragment of a response.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
		Output Buffer Control Byte (OBCB)			**Input Buffer Control Byte (IBCB)**
00:	82	7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	8B	7 6 5 4 3 2 1 0 [1] [0] [0] [0] **[1]** [0] [1] **[1]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	1C	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	00	**(CBx Response word length MSB)**
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	1F	**(CBx Response word length LSB)** Minimum of 6 words
05:	AA	(CBx Command Type) Always AA	05:	AA	**(CBx Response Type)** AA=Normal Response
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	06:	05	**(CBx Response Opcode)** 05=Continuous Read Response
07:	00	(CBx Command, byte not used)	07:	00	**(CBx Response Instance Counter)**
08:	01	(CBx Command “Node ID”)	08:	01	**(CBx Response “Node ID”)**
09:	03	(CBx Command Timeout MSB)	09:	01	**(CBx Response Timestamp Month)**
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	10:	01	**(CBx Response Timestamp Day)**
11:	00	(CBx Command Start Address MSB)	11:	00	**(CBx Response Timestamp Hour)**
12:	00	(CBx Command Start Address LSB) address 0	12:	01	**(CBx Response Timestamp Minute)**
13:	00	(CBx Command Length MSB)	13:	1D	**(CBx Response Timestamp Second)**
14:	32	(CBx Command Length LSB) 50 bytes	14:	32	**(CBx Response “Data length”)** 50 bytes (total Tag Data)
15:	00		15:	2F	**(CBx Response Data Byte 1)**
16:	00		16:	13	**(CBx Response Data Byte 2)**
17:	00		17:	19	**(CBx Response Data Byte 3)**
18:	00		18:	45	**(CBx Response Data Byte 4)**
19:	00		19:	94	**(CBx Response Data Byte 5)**
20:	00		20:	D1	**(CBx Response Data Byte 6)**
..	..		21:	B5	**(CBx Response Data Byte 7)**
30:	00		22:	FA	**(CBx Response Data Byte 8)**
			23:	C7	**(CBx Response Data Byte 9)**
			24:	42	**(CBx Response Data Byte 10)**
			25:	33	**(CBx Response Data Byte 11)**
			26:	58	**(CBx Response Data Byte 12)**
			27:	A3	**(CBx Response Data Byte 13)**
			28:	55	**(CBx Response Data Byte 14)**
			29:	88	**(CBx Response Data Byte 15)**
			30:	49	**(CBx Response Data Byte 16)**
31:	82	**Data Consistency Byte (OBDCB)**	31:	8B	**Data Consistency Byte (IBDCB)**

The Master can see that **Bit 3** of the **IBCB** & **IBDCB** has been set to 1, so it knows that the response in the **Input Buffer** is just a fragment of a longer response, and not a complete response, and that there are more fragments to follow.

The Master now toggles **Bit 0** of the **OBCB** & **OBDCB** to acknowledge that it has received the response fragment.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	83	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] **[1]**	00	8B	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [1] [1]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	1C	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	00	(CBx Response word length MSB)
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	1F	(CBx Response word length LSB) Minimum of 6 words
05:	AA	(CBx Command Type) Always AA	05:	AA	(CBx Response Type) AA=Normal Response
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	06:	05	(CBx Response Opcode) 05=Continuous Read Response
07:	00	(CBx Command, byte not used)	07:	00	(CBx Response Instance Counter)
08:	01	(CBx Command “Node ID”)	08:	01	(CBx Response “Node ID”)
09:	03	(CBx Command Timeout MSB)	09:	01	(CBx Response Timestamp Month)
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	10:	01	(CBx Response Timestamp Day)
11:	00	(CBx Command Start Address MSB)	11:	00	(CBx Response Timestamp Hour)
12:	00	(CBx Command Start Address LSB) address 0	12:	01	(CBx Response Timestamp Minute)
13:	00	(CBx Command Length MSB)	13:	1D	(CBx Response Timestamp Second)
14:	32	(CBx Command Length LSB) 50 bytes	14:	32	(CBx Response “Data length”) 50 bytes (total Tag Data)
15:	00		15:	2F	(CBx Response Data Byte 1)
16:	00		16:	13	(CBx Response Data Byte 2)
17:	00		17:	19	(CBx Response Data Byte 3)
18:	00		18:	45	(CBx Response Data Byte 4)
19:	00		19:	94	(CBx Response Data Byte 5)
20:	00		20:	D1	(CBx Response Data Byte 6)
..	..		21:	B5	(CBx Response Data Byte 7)
30:	00		22:	FA	(CBx Response Data Byte 8)
			23:	C7	(CBx Response Data Byte 9)
			24:	42	(CBx Response Data Byte 10)
			25:	33	(CBx Response Data Byte 11)
			26:	58	(CBx Response Data Byte 12)
			27:	A3	(CBx Response Data Byte 13)
			28:	55	(CBx Response Data Byte 14)
			29:	88	(CBx Response Data Byte 15)
			30:	49	(CBx Response Data Byte 16)
31:	83	**Data Consistency Byte (OBDCB)**	31:	8B	**Data Consistency Byte (IBDCB)**

After the Master acknowledges that it has received the fragment, the Slave places the next fragment in the Input Buffer and toggles **Bit 0** of the **IBCB** & **IBDCB**.

Since this is still not the last fragment, the Save leaves **Bit 3** set to 1 in the **IBCB** & **IBDCB**

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	83	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [1]	00	**8A**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [1] **[0]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	**1C**	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	**02**	**(CBx Response Data Byte 17)**
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	**02**	**(CBx Response Data Byte 18)**
			05:	**02**	**(CBx Response Data Byte 19)**
05:	AA	(CBx Command Type) Always AA	06:	**02**	**(CBx Response Data Byte 20)**
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	07:	**02**	**(CBx Response Data Byte 21)**
			08:	**02**	**(CBx Response Data Byte 22)**
07:	00	(CBx Command, byte not used)	09:	**02**	**(CBx Response Data Byte 23)**
08:	01	(CBx Command “Node ID”)	10:	**02**	**(CBx Response Data Byte 24)**
09:	03	(CBx Command Timeout MSB)	11:	**02**	**(CBx Response Data Byte 25)**
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	12:	**02**	**(CBx Response Data Byte 26)**
			13:	**02**	**(CBx Response Data Byte 27)**
11:	00	(CBx Command Start Address MSB)	14:	**02**	**(CBx Response Data Byte 28)**
12:	00	(CBx Command Start Address LSB) address 0	15:	**02**	**(CBx Response Data Byte 29)**
			16:	**02**	**(CBx Response Data Byte 30)**
13:	00	(CBx Command Length MSB)	17:	**02**	**(CBx Response Data Byte 31)**
14:	32	(CBx Command Length LSB) 50 bytes	18:	**02**	**(CBx Response Data Byte 32)**
			19:	**02**	**(CBx Response Data Byte 33)**
15:	00		20:	**02**	**(CBx Response Data Byte 34)**
16:	00		21:	**02**	**(CBx Response Data Byte 35)**
17:	00		22:	**02**	**(CBx Response Data Byte 36)**
18:	00		23:	**02**	**(CBx Response Data Byte 37)**
19:	00		24:	**02**	**(CBx Response Data Byte 38)**
20:	00		25:	**02**	**(CBx Response Data Byte 39)**
..	..		26:	**02**	**(CBx Response Data Byte 40)**
30:	00		27:	**02**	**(CBx Response Data Byte 41)**
			28:	**02**	**(CBx Response Data Byte 42)**
			29:	**02**	**(CBx Response Data Byte 43)**
			30:	**02**	**(CBx Response Data Byte 44)**
31:	83	**Data Consistency Byte (OBDCB)**	31:	**8A**	**Data Consistency Byte (IBDCB)**

Now the Master acknowledges this fragment by toggling **Bit 0** of the **OBCB** & **OBDCB**. It knows that this is still not the last fragment of the response, since **Bit 3** of the **IBCB** & **IBDCB** is still set to 1.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**82**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] **[0]**	00	8A	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [1] [0]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	1C	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	02	(CBx Response Data Byte 17)
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	02	(CBx Response Data Byte 18)
			05:	02	(CBx Response Data Byte 19)
05:	AA	(CBx Command Type) Always AA	06:	02	(CBx Response Data Byte 20)
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	07:	02	(CBx Response Data Byte 21)
			08:	02	(CBx Response Data Byte 22)
07:	00	(CBx Command, byte not used)	09:	02	(CBx Response Data Byte 23)
08:	01	(CBx Command “Node ID”)	10:	02	(CBx Response Data Byte 24)
09:	03	(CBx Command Timeout MSB)	11:	02	(CBx Response Data Byte 25)
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	12:	02	(CBx Response Data Byte 26)
			13:	02	(CBx Response Data Byte 27)
11:	00	(CBx Command Start Address MSB)	14:	02	(CBx Response Data Byte 28)
12:	00	(CBx Command Start Address LSB) address 0	15:	02	(CBx Response Data Byte 29)
			16:	02	(CBx Response Data Byte 30)
13:	00	(CBx Command Length MSB)	17:	02	(CBx Response Data Byte 31)
14:	32	(CBx Command Length LSB) 50 bytes	18:	02	(CBx Response Data Byte 32)
			19:	02	(CBx Response Data Byte 33)
15:	00		20:	02	(CBx Response Data Byte 34)
16:	00		21:	02	(CBx Response Data Byte 35)
17:	00		22:	02	(CBx Response Data Byte 36)
18:	00		23:	02	(CBx Response Data Byte 37)
19:	00		24:	02	(CBx Response Data Byte 38)
20:	00		25:	02	(CBx Response Data Byte 39)
..	..		26:	02	(CBx Response Data Byte 40)
30:	00		27:	02	(CBx Response Data Byte 41)
			28:	02	(CBx Response Data Byte 42)
			29:	02	(CBx Response Data Byte 43)
			30:	02	(CBx Response Data Byte 44)
31:	**82**	**Data Consistency Byte (OBDCB)**	31:	8A	**Data Consistency Byte (IBDCB)**

Now the Slave places the final fragment into the Input Buffer and toggles **Bit 0** of the **IBCB** & **IBDCB** to indicate the new fragment is ready.

Since it is the final fragment, the Slave also now clears **Bit 3** of the **IBCB** & **IBDCB**:

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	82	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	**83**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] **[0]** [0] [1] **[1]**
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	**06**	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	**02**	**(CBx Response Data Byte 45)**
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	**02**	**(CBx Response Data Byte 46)**
			05:	**02**	**(CBx Response Data Byte 47)**
05:	AA	(CBx Command Type) Always AA	06:	**02**	**(CBx Response Data Byte 48)**
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	07:	**02**	**(CBx Response Data Byte 49)**
			08:	**02**	**(CBx Response Data Byte 50)**
07:	00	(CBx Command, byte not used)	09:	00	
08:	01	(CBx Command “Node ID”)	10:	00	
09:	03	(CBx Command Timeout MSB)	11:	00	
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	12:	00	
			13:	00	
11:	00	(CBx Command Start Address MSB)	14:	00	
12:	00	(CBx Command Start Address LSB) address 0	15:	00	
			16:	00	
13:	00	(CBx Command Length MSB)	17:	00	
14:	32	(CBx Command Length LSB) 50 bytes	18:	00	
			19:	00	
15:	00		20:	00	
16:	00		21:	00	
17:	00		22:	00	
18:	00		23:	00	
19:	00		24:	00	
20:	00		25:	00	
..	..		26:	00	
30:	00		27:	00	
			28:	00	
			29:	00	
			30:	00	
31:	82	**Data Consistency Byte (OBDCB)**	31:	**83**	**Data Consistency Byte (IBDCB)**

And lastly, the Master acknowledges receipt of the final fragment by toggling **Bit 0** of its **OBCB** & **OBDCB**:

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
		Output Buffer Control Byte (OBCB)			**Input Buffer Control Byte (IBCB)**
00:	**83**	7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] **[1]**	00	83	7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [1]
01:	00	(Always 0)	01:	00	(Always 0)
02:	0C	(Packet length in bytes)	02:	06	(Packet length in bytes)
03:	00	(CBx Command word length MSB)	03:	02	(CBx Response Data Byte 45)
04:	06	(CBx Command word length LSB) Minimum of 6 words	04:	02	(CBx Response Data Byte 46)
			05:	02	(CBx Response Data Byte 47)
05:	AA	(CBx Command Type) Always AA	06:	02	(CBx Response Data Byte 48)
06:	05	(CBx Command Opcode) 0x05 = Read Tag Data	07:	02	(CBx Response Data Byte 49)
			08:	02	(CBx Response Data Byte 50)
07:	00	(CBx Command, byte not used)	09:	00	
08:	01	(CBx Command “Node ID”)	10:	00	
09:	03	(CBx Command Timeout MSB)	11:	00	
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	12:	00	
			13:	00	
11:	00	(CBx Command Start Address MSB)	14:	00	
12:	00	(CBx Command Start Address LSB) address 0	15:	00	
			16:	00	
13:	00	(CBx Command Length MSB)	17:	00	
14:	32	(CBx Command Length LSB) 50 bytes	18:	00	
			19:	00	
15:	00		20:	00	
16:	00		21:	00	
17:	00		22:	00	
18:	00		23:	00	
19:	00		24:	00	
20:	00		25:	00	
..	..		26:	00	
30:	00		27:	00	
			28:	00	
			29:	00	
			30:	00	
31:	**83**	**Data Consistency Byte (OBDCB)**	31:	83	**Data Consistency Byte (IBDCB)**

The command/response sequence has completed. A command has been issued and the response received and all fragments of a response have been retrieved and acknowledged.

9.5.4 Example 4: Fragmentation of Commands

For this example, the Master will send a CBx “Write Tag Data” command to the Slave (the RFID Controller at Node 01 of the Subnet16™ Profibus Gateway network) to write 50 bytes to a tag.

We will assume for this example that the both the input and output buffers have been configured to 32 bytes each. This means that the command itself cannot completely fit in the output buffer, and therefore needs to be sent in fragments. Long Tag writes that exceed the buffer size can be separated into multiple writes, with each write addressed to a different location of the tag, but if it is desirable to send one long CBx command, it can be accomplished using this method of fragmentation:

Sending the command:

In **Byte 2** of the output buffer the Master places the length (in bytes) of the data packet (first Fragment of the CBx Command) we are sending – in this case the first fragment will be 28 bytes – the maximum size of a packet when the output buffer is 32 bytes.

(The entire CBx command we are planning to send, over 3 fragments, is 62 bytes).

In **Byte 3** through **Byte 30** the Master places the first 28 bytes of this CBx command.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	80	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	**1C**	**(Packet length in bytes)**	02:	00	
03:	**00**	**(CBx Command word length MSB)**	03:	00	
04:	**1F**	**(CBx Command word length LSB)**	04:	00	
05:	**AA**	**(CBx Command Type)** Always AA	05:	00	
06:	**06**	**(CBx Command Opcode)**	06:	00	
		0x06 = Write Tag Data	07:	00	
07:	**00**	**(CBx Command, byte not used)**	08:	00	
08:	**01**	**(CBx Command “Node ID”)**	09:	00	
09:	**03**	**(CBx Command Timeout MSB)**	10:	00	
10:	**E8**	**(CBx Command Timeout LSB)**	11:	00	
		0xE8 = 1000 ms timeout	12:	00	
11:	**00**	**(CBx Command Start Address MSB)**	13:	00	
12:	**00**	**(CBx Command Start Address LSB)**	14:	00	
		address 0	15:	00	
13:	**00**	**(CBx Command Length MSB)**	16:	00	
14:	**32**	**(CBx Command Length LSB)**	17:	00	
		50 bytes	18:	00	
15:	**41**	**(CBx Command Data Byte 1)**	19:	00	
16:	**42**	**(CBx Command Data Byte 2)**	20:	00	
17:	**43**	**(CBx Command Data Byte 3)**	..	..	
18:	**44**	**(CBx Command Data Byte 4)**	30:	00	
19:	**45**	**(CBx Command Data Byte 5)**			
20:	**46**	**(CBx Command Data Byte 6)**			
21:	**47**	**(CBx Command Data Byte 7)**			
22:	**48**	**(CBx Command Data Byte 8)**			
23:	**49**	**(CBx Command Data Byte 9)**			
24:	**50**	**(CBx Command Data Byte 10)**			
25:	**51**	**(CBx Command Data Byte 11)**			
26:	**52**	**(CBx Command Data Byte 12)**			
27:	**53**	**(CBx Command Data Byte 13)**			
28:	**54**	**(CBx Command Data Byte 14)**			
29:	**55**	**(CBx Command Data Byte 15)**			
30:	**56**	**(CBx Command Data Byte 16)**			
31:	80	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

Now that the first command fragment is in the Output Buffer, the Master alerts the Slave that the command fragment is ready. It does this by toggling **Bit 1** of the **OBCB** & **OBDCB**.

Since there are more command fragments to follow to complete the command, the Master also sets **Bit 3** of the **OBCB** & **OBDCB** to 1. This bit is what tells the Slave to wait for further fragments before processing the command.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
		Output Buffer Control Byte (OBCB)			**Input Buffer Control Byte (IBCB)**
00:	**8A**	7 6 5 4 3 2 1 0 [1] [0] [0] [0] **[1]** [0] **[1]** [0]	00	80	7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	1C	(Packet length in bytes)	02:	00	
03:	00	(CBx Command word length MSB)	03:	00	
04:	1F	(CBx Command word length LSB)	04:	00	
05:	AA	(CBx Command Type) Always AA	05:	00	
06:	06	(CBx Command Opcode)	06:	00	
		0x06 = Write Tag Data	07:	00	
07:	00	(CBx Command, byte not used)	08:	00	
08:	01	(CBx Command “Node ID”)	09:	00	
09:	03	(CBx Command Timeout MSB)	10:	00	
10:	E8	(CBx Command Timeout LSB)	11:	00	
		0xE8 = 1000 ms timeout	12:	00	
11:	00	(CBx Command Start Address MSB)	13:	00	
12:	00	(CBx Command Start Address LSB)	14:	00	
		address 0	15:	00	
13:	00	(CBx Command Length MSB)	16:	00	
14:	32	(CBx Command Length LSB)	17:	00	
		50 bytes	18:	00	
15:	41	(CBx Command Data Byte 1)	19:	00	
16:	42	(CBx Command Data Byte 2)	20:	00	
17:	43	(CBx Command Data Byte 3)	..	..	
18:	44	(CBx Command Data Byte 4)	30:	00	
19:	45	(CBx Command Data Byte 5)			
20:	46	(CBx Command Data Byte 6)			
21:	47	(CBx Command Data Byte 7)			
22:	48	(CBx Command Data Byte 8)			
23:	49	(CBx Command Data Byte 9)			
24:	50	(CBx Command Data Byte 10)			
25:	51	(CBx Command Data Byte 11)			
26:	52	(CBx Command Data Byte 12)			
27:	53	(CBx Command Data Byte 13)			
28:	54	(CBx Command Data Byte 14)			
29:	55	(CBx Command Data Byte 15)			
30:	56	(CBx Command Data Byte 16)			
31:	**8A**	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

When the Slave sees Bit 1 of the **OBCB** & **OBDBC** toggle, it grabs the command fragment from the **Output Buffer.** The Slave then acknowledges the command fragment by toggling **Bit 1** of the **IBCB** & **IBDCB**.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	8A	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [1] [0]	00	**82**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1]** [0]
01:	00	(Always 0)	01:	00	
02:	1C	(Packet length in bytes)	02:	00	
03:	00	(CBx Command word length MSB)	03:	00	
04:	1F	(CBx Command word length LSB)	04:	00	
05:	AA	(CBx Command Type) Always AA	05:	00	
06:	06	(CBx Command Opcode) 0x06 = Write Tag Data	06:	00	
			07:	00	
07:	00	(CBx Command, byte not used)	08:	00	
08:	01	(CBx Command “Node ID”)	09:	00	
09:	03	(CBx Command Timeout MSB)	10:	00	
10:	E8	(CBx Command Timeout LSB) 0xE8 = 1000 ms timeout	11:	00	
			12:	00	
11:	00	(CBx Command Start Address MSB)	13:	00	
12:	00	(CBx Command Start Address LSB) address 0	14:	00	
			15:	00	
13:	00	(CBx Command Length MSB)	16:	00	
14:	32	(CBx Command Length LSB) 50 bytes	17:	00	
			18:	00	
15:	41	(CBx Command Data Byte 1)	19:	00	
16:	42	(CBx Command Data Byte 2)	20:	00	
17:	43	(CBx Command Data Byte 3)	..	..	
18:	44	(CBx Command Data Byte 4)	30:	00	
19:	45	(CBx Command Data Byte 5)			
20:	46	(CBx Command Data Byte 6)			
21:	47	(CBx Command Data Byte 7)			
22:	48	(CBx Command Data Byte 8)			
23:	49	(CBx Command Data Byte 9)			
24:	50	(CBx Command Data Byte 10)			
25:	51	(CBx Command Data Byte 11)			
26:	52	(CBx Command Data Byte 12)			
27:	53	(CBx Command Data Byte 13)			
28:	54	(CBx Command Data Byte 14)			
29:	55	(CBx Command Data Byte 15)			
30:	56	(CBx Command Data Byte 16)			
31:	8A	**Data Consistency Byte (OBDCB)**	31:	**82**	**Data Consistency Byte (IBDCB)**

Now that the Slave has acknowledged receiving the command fragment, the Master writes the next command fragment into the Output Buffer:

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	8A	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [1] [0]	00	82	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]
01:	00	(Always 0)	01:	00	
02:	**1C**	**(Packet length in bytes)**	02:	00	
03:	**57**	**(CBx Command Data Byte 17)**	03:	00	
04:	**58**	**(CBx Command Data Byte 18)**	04:	00	
05:	**59**	**(CBx Command Data Byte 19)**	05:	00	
06:	**60**	**(CBx Command Data Byte 20)**	06:	00	
07:	**61**	**(CBx Command Data Byte 21)**	07:	00	
08:	**62**	**(CBx Command Data Byte 22)**	08:	00	
09:	**63**	**(CBx Command Data Byte 23)**	09:	00	
10:	**64**	**(CBx Command Data Byte 24)**	10:	00	
11:	**65**	**(CBx Command Data Byte 25)**	11:	00	
12:	**66**	**(CBx Command Data Byte 26)**	12:	00	
13:	**67**	**(CBx Command Data Byte 27)**	13:	00	
14:	**68**	**(CBx Command Data Byte 28)**	14:	00	
15:	**69**	**(CBx Command Data Byte 29)**	15:	00	
16:	**70**	**(CBx Command Data Byte 30)**	16:	00	
17:	**71**	**(CBx Command Data Byte 31)**	17:	00	
18:	**72**	**(CBx Command Data Byte 32)**	18:	00	
19:	**73**	**(CBx Command Data Byte 33)**	19:	00	
20:	**74**	**(CBx Command Data Byte 34)**	20:	00	
21:	**75**	**(CBx Command Data Byte 35)**	..	..	
22:	**76**	**(CBx Command Data Byte 36)**	30:	00	
23:	**77**	**(CBx Command Data Byte 37)**			
24:	**78**	**(CBx Command Data Byte 38)**			
25:	**79**	**(CBx Command Data Byte 39)**			
26:	**80**	**(CBx Command Data Byte 40)**			
27:	**81**	**(CBx Command Data Byte 41)**			
28:	**82**	**(CBx Command Data Byte 42)**			
29:	**83**	**(CBx Command Data Byte 43)**			
30:	**84**	**(CBx Command Data Byte 44)**			
31:	8A	**Data Consistency Byte (OBDCB)**	31:	82	**Data Consistency Byte (IBDCB)**

Next, the Master signals that this fragment is ready, by toggling **Bit 1** of the **OBCB** & **OBDCB**. Since this is still not the final fragment, the Master leaves **Bit 3** set to 1.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**88**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] **[0]** [0]	00	82	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]
01:	00	(Always 0)	01:	00	
02:	1C	(Packet length in bytes)	02:	00	
03:	57	(CBx Command Data Byte 17)	03:	00	
04:	58	(CBx Command Data Byte 18)	04:	00	
05:	59	(CBx Command Data Byte 19)	05:	00	
06:	60	(CBx Command Data Byte 20)	06:	00	
07:	61	(CBx Command Data Byte 21)	07:	00	
08:	62	(CBx Command Data Byte 22)	08:	00	
09:	63	(CBx Command Data Byte 23)	09:	00	
10:	64	(CBx Command Data Byte 24)	10:	00	
11:	65	(CBx Command Data Byte 25)	11:	00	
12:	66	(CBx Command Data Byte 26)	12:	00	
13:	67	(CBx Command Data Byte 27)	13:	00	
14:	68	(CBx Command Data Byte 28)	14:	00	
15:	69	(CBx Command Data Byte 29)	15:	00	
16:	70	(CBx Command Data Byte 30)	16:	00	
17:	71	(CBx Command Data Byte 31)	17:	00	
18:	72	(CBx Command Data Byte 32)	18:	00	
19:	73	(CBx Command Data Byte 33)	19:	00	
20:	74	(CBx Command Data Byte 34)	20:	00	
21:	75	(CBx Command Data Byte 35)	..	..	
22:	76	(CBx Command Data Byte 36)	30:	00	
23:	77	(CBx Command Data Byte 37)			
24:	78	(CBx Command Data Byte 38)			
25:	79	(CBx Command Data Byte 39)			
26:	80	(CBx Command Data Byte 40)			
27:	81	(CBx Command Data Byte 41)			
28:	82	(CBx Command Data Byte 42)			
29:	83	(CBx Command Data Byte 43)			
30:	84	(CBx Command Data Byte 44)			
31:	**88**	**Data Consistency Byte (OBDCB)**	31:	82	**Data Consistency Byte (IBDCB)**

When the Slave sees Bit 1 of the **OBCB** & **OBDBC** toggle, it grabs this command fragment from the Output Buffer. The Slave then acknowledges the command fragment by toggling **Bit 1** of the **IBCB** & **IBDCB**.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	88	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [0] [0]	00	**80**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[0]** [0]
01:	00	(Always 0)	01:	00	
02:	1C	(Packet length in bytes)	02:	00	
03:	57	(CBx Command Data Byte 17)	03:	00	
04:	58	(CBx Command Data Byte 18)	04:	00	
05:	59	(CBx Command Data Byte 19)	05:	00	
06:	60	(CBx Command Data Byte 20)	06:	00	
07:	61	(CBx Command Data Byte 21)	07:	00	
08:	62	(CBx Command Data Byte 22)	08:	00	
09:	63	(CBx Command Data Byte 23)	09:	00	
10:	64	(CBx Command Data Byte 24)	10:	00	
11:	65	(CBx Command Data Byte 25)	11:	00	
12:	66	(CBx Command Data Byte 26)	12:	00	
13:	67	(CBx Command Data Byte 27)	13:	00	
14:	68	(CBx Command Data Byte 28)	14:	00	
15:	69	(CBx Command Data Byte 29)	15:	00	
16:	70	(CBx Command Data Byte 30)	16:	00	
17:	71	(CBx Command Data Byte 31)	17:	00	
18:	72	(CBx Command Data Byte 32)	18:	00	
19:	73	(CBx Command Data Byte 33)	19:	00	
20:	74	(CBx Command Data Byte 34)	20:	00	
21:	75	(CBx Command Data Byte 35)	..	..	
22:	76	(CBx Command Data Byte 36)	30:	00	
23:	77	(CBx Command Data Byte 37)			
24:	78	(CBx Command Data Byte 38)			
25:	79	(CBx Command Data Byte 39)			
26:	80	(CBx Command Data Byte 40)			
27:	81	(CBx Command Data Byte 41)			
28:	82	(CBx Command Data Byte 42)			
29:	83	(CBx Command Data Byte 43)			
30:	84	(CBx Command Data Byte 44)			
31:	88	**Data Consistency Byte (OBDCB)**	31:	**80**	**Data Consistency Byte (IBDCB)**

Now that the Slave has acknowledged receiving the command fragment, the Master writes the next (and final) command fragment into the Output Buffer:

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	88	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [1] [0] [0] [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	**06**	**(Packet length in bytes)**	02:	00	
03:	**85**	**(CBx Command Data Byte 45)**	03:	00	
04:	**86**	**(CBx Command Data Byte 46)**	04:	00	
05:	**87**	**(CBx Command Data Byte 47)**	05:	00	
06:	**88**	**(CBx Command Data Byte 48)**	06:	00	
07:	**89**	**(CBx Command Data Byte 49)**	07:	00	
08:	**90**	**(CBx Command Data Byte 50)**	08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
21:	00		..	..	
22:	00		30:	00	
23:	00				
24:	00				
25:	00				
26:	00				
27:	00				
28:	00				
29:	00				
30:	00				
31:	88	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

Next, the Master signals that this fragment is ready, by toggling **Bit 1** of the **OBCB** & **OBDCB**. Since this **is** the final fragment, the Master clears **Bit 3** to 0.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**82**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] **[0]** [0] **[1]** [0]	00	80	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	85	(CBx Command Data Byte 45)	03:	00	
04:	86	(CBx Command Data Byte 46)	04:	00	
05:	87	(CBx Command Data Byte 47)	05:	00	
06:	88	(CBx Command Data Byte 48)	06:	00	
07:	89	(CBx Command Data Byte 49)	07:	00	
08:	90	(CBx Command Data Byte 50)	08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
21:	00		..	..	
22:	00		30:	00	
23:	00				
24:	00				
25:	00				
26:	00				
27:	00				
28:	00				
29:	00				
30:	00				
31:	**82**	**Data Consistency Byte (OBDCB)**	31:	80	**Data Consistency Byte (IBDCB)**

When the Slave sees Bit 1 of the **OBCB** & **OBDBC** toggle, it grabs this command fragment from the Output Buffer. The Slave then acknowledges the command fragment by toggling **Bit 1** of the **IBCB** & **IBDCB**.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	82	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	**82**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1]** [0]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	85	(CBx Command Data Byte 45)	03:	00	
04:	86	(CBx Command Data Byte 46)	04:	00	
05:	87	(CBx Command Data Byte 47)	05:	00	
06:	88	(CBx Command Data Byte 48)	06:	00	
07:	89	(CBx Command Data Byte 49)	07:	00	
08:	90	(CBx Command Data Byte 50)	08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
21:	00		..	..	
22:	00		30:	00	
23:	00				
24:	00				
25:	00				
26:	00				
27:	00				
28:	00				
29:	00				
30:	00				
31:	82	**Data Consistency Byte (OBDCB)**	31:	**82**	**Data Consistency Byte (IBDCB)**

The Slave, at this point, after acknowledging the final fragment, knows it has the complete CBx command, so it processes the command.

Assuming the command is successful, the Slave will write the response (in this case a “Tag Write Successful” response) into the Input buffer, and then toggle **Bit 0** of the **IBCB** & **IBDCB**.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	82	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]	00	**83**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] **[1] [1]**
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	**0C**	**(Packet length in bytes)**
03:	85	(CBx Command Data Byte 45)	03:	**00**	**(CBx Response word length MSB)**
04:	86	(CBx Command Data Byte 46)	04:	**06**	**(CBx Response word length LSB)**
05:	87	(CBx Command Data Byte 47)			Minimum of 6 words
06:	88	(CBx Command Data Byte 48)	05:	**AA**	**(CBx Response Type)**
07:	89	(CBx Command Data Byte 49)			AA=Normal Response
08:	90	(CBx Command Data Byte 50)	06:	**06**	**(CBx Response Opcode)**
09:	00				06=echo of "Tag Write"
10:	00		07:	**01**	**(CBx Response Instance Counter)**
11:	00		08:	**01**	**(CBx Response “Node ID”)**
12:	00		09:	**01**	**(CBx Response Timestamp Month)**
13:	00		10:	**01**	**(CBx Response Timestamp Day)**
14:	00		11:	**00**	**(CBx Response Timestamp Hour)**
15:	00		12:	**01**	**(CBx Response Timestamp Minute)**
16:	00		13:	**20**	**(CBx Response Timestamp Second)**
17:	00		14:	**00**	**(CBx Response Not Used)**
18:	00		15:	00	
19:	00		16:	00	
20:	00		17:	00	
21:	00		18:	00	
22:	00		19:	00	
23:	00		20:	00	
24:	00		..	..	
25:	00		30:	00	
26:	00				
27:	00				
28:	00				
29:	00				
30:	00				
31:	82	**Data Consistency Byte (OBDCB)**	31:	**83**	**Data Consistency Byte (IBDCB)**

The Master now toggles **Bit 0** of the **OBCB** & **OBDCB** to acknowledge that it has received the response.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**83**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] **[1]**	00	83	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [1]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	0C	(Packet length in bytes)
03:	85	(CBx Command Data Byte 45)	03:	00	(CBx Response word length MSB)
04:	86	(CBx Command Data Byte 46)	04:	06	(CBx Response word length LSB) Minimum of 6 words
05:	87	(CBx Command Data Byte 47)			
06:	88	(CBx Command Data Byte 48)	05:	AA	(CBx Response Type) AA=Normal Response
07:	89	(CBx Command Data Byte 49)			
08:	90	(CBx Command Data Byte 50)	06:	06	(CBx Response Opcode) 06=echo of "Tag Write"
09:	00				
10:	00		07:	01	(CBx Response Instance Counter)
11:	00		08:	01	(CBx Response “Node ID”)
12:	00		09:	01	(CBx Response Timestamp Month)
13:	00		10:	01	(CBx Response Timestamp Day)
14:	00		11:	00	(CBx Response Timestamp Hour)
15:	00		12:	01	(CBx Response Timestamp Minute)
16:	00		13:	20	(CBx Response Timestamp Second)
17:	00		14:	00	(CBx Response Not Used)
18:	00		15:	00	
19:	00		16:	00	
20:	00		17:	00	
21:	00		18:	00	
22:	00		19:	00	
23:	00		20:	00	
24:	00		..	..	
25:	00		30:	00	
26:	00				
27:	00				
28:	00				
29:	00				
30:	00				
31:	**83**	**Data Consistency Byte (OBDCB)**	31:	83	**Data Consistency Byte (IBDCB)**

The command/response sequence has completed. A command has been issued over 3 fragments and processed, and the response received and the response has been acknowledged.

9.5.5 Example 5: Resynchronization

For this example we will assume the same conditions as the previous example, that the input buffer and output buffer are 32 bytes each.

It does not matter what data is currently in the two buffers, other than the control bytes and data consistency bytes – resynchronization only resets the handshaking to a known state.

For this example we will assume a starting state as follows:

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	80	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]	00	82	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	00		03:	00	
04:	00		04:	00	
05:	00		05:	00	
06:	00		06:	00	
07:	00		07:	00	
08:	00		08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..			..	..	
30:			30:	00	
31:	80	**Data Consistency Byte (OBDCB)**	31:	82	**Data Consistency Byte (IBDCB)**

If the Master believes that the handshaking has gotten out of synch, it can request a resynchronization, by setting **Bit 2** of the **Output Buffer Control Byte (**the **OBCB)** and then also setting the same bit in the **Output Buffer Data Consistency Byte** (the **OBDCB**).



NOTE

Bit 2 is not a toggle – It is always set to 1 to begin a resynchronization process, and cleared later to acknowledge that the process is complete.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	**84**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] **[1]** [0] [0]	00	82	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [1] [0]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	00		03:	00	
04:	00		04:	00	
05:	00		05:	00	
06:	00		06:	00	
07:	00		07:	00	
08:	00		08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..			..	..	
30:			30:	00	
31:	**84**	**Data Consistency Byte (OBDCB)**	31:	82	**Data Consistency Byte (IBDCB)**

When the slave sees **Bit 2** In the **OBCB** & **OBDCB** set, it knows it needs to resynchronize its handshaking bits in the **IBCB** & **IBDCB.**

So the Slave will acknowledge the resynchronization request by setting **Bit 2,** and will clear **Bit 1** and **Bit 0** in the **IBCB** & **IBDCB**.

Note that whatever values Bit 1 or Bit 0 had, they will be set to 0. This process forces the handshaking into a known state.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	84	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [1] [0] [0]	00	**84**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] **[1] [0] [0]**
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	00		03:	00	
04:	00		04:	00	
05:	00		05:	00	
06:	00		06:	00	
07:	00		07:	00	
08:	00		08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..			..	..	
30:			30:	00	
31:	84	**Data Consistency Byte (OBDCB)**	31:	**84**	**Data Consistency Byte (IBDCB)**

When the Master sees **Bit 2** of the **IBCB** & **IBDCB** set, it clears **Bit 2** of the **OBCB** & **OBDCB** to acknowledge that the Slave has resynchronized.

(See the **Green** changes below)

Output Buffer			Input Buffer		
Byte #	**Value**		**Byte #**	**Value**	
00:	**80**	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] **[0]** [0] [0]	00	84	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [1] [0] [0]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	00		03:	00	
04:	00		04:	00	
05:	00		05:	00	
06:	00		06:	00	
07:	00		07:	00	
08:	00		08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..			..	..	
30:			30:	00	
31:	**80**	**Data Consistency Byte (OBDCB)**	31:	84	**Data Consistency Byte (IBDCB)**

And lastly, when the Slave sees the Master clear **Bit 2** of the **OBCD** & **OBCDB**, it clears **Bit 2** of the **IBCB** & **IBDCB** to complete the resynchronization process.

(See the **Green** changes below)

Output Buffer			**Input Buffer**		
Byte #	**Value**		**Byte #**	**Value**	
00:	80	**Output Buffer Control Byte (OBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] [0] [0] [0]	00	**80**	**Input Buffer Control Byte (IBCB)** 7 6 5 4 3 2 1 0 [1] [0] [0] [0] [0] **[0]** [0] [0]
01:	00	(Always 0)	01:	00	
02:	06	(Packet length in bytes)	02:	00	
03:	00		03:	00	
04:	00		04:	00	
05:	00		05:	00	
06:	00		06:	00	
07:	00		07:	00	
08:	00		08:	00	
09:	00		09:	00	
10:	00		10:	00	
11:	00		11:	00	
12:	00		12:	00	
13:	00		13:	00	
14:	00		14:	00	
15:	00		15:	00	
16:	00		16:	00	
17:	00		17:	00	
18:	00		18:	00	
19:	00		19:	00	
20:	00		20:	00	
..			..	..	
30:			30:	00	
31:	80	**Data Consistency Byte (OBDCB)**	31:	**80**	**Data Consistency Byte (IBDCB)**

The Resynchronization process is complete. The Slave is now in a known state, with the handshake bits set to zero, and internally in a state of “waiting for a new command”.

10 TECHNICAL FEATURES

ELECTRICAL FEATURES	
Supply Voltage	12 to 30 Vdc
DC Input Current max.	200 – 100 mA
Communication Interfaces: Host (depending on model) RFID Multidrop Readers Configuration	 RS232, Ethernet/IP, Modbus TCP/IP, Standard TCP/IP, DeviceNet, Profibus Subnet16™ (uses RS485 physical layer) USB
Subnet16™ Baud Rate	9600 (default), 19.2k, 38.4k, 57.6k, 115.2k
ENVIRONMENTAL FEATURES	
Operating Temperature	-20°to 50 ℃ (-4 to 122 ℉)
Storage Temperature	-20°to 70 ℃ (-4 to 158 ℉)
Humidity max.	90% non condensing
Vibration Resistance EN 60068-2-6 2 hours on each axis	14 mm @ 2 to 10 Hz 1.5 mm @ 13 to 55 Hz 2 g @ 70 to 200 Hz
Shock Resistance EN 60068-2-27	30 g; 11 ms; 3 shocks on each axis
Protection Class EN 60529	IP30
PHYSICAL FEATURES	
Mechanical Dimensions	104 x 107 x 32 mm (4.1 x 4.2 x 1.3 in)
Weight	256 g. (9 oz.)
Enclosure	Stainless Steel 304 (18-8)
USER INTERFACE	
LED Indicators	Power On (green) SubNet16™ Bus (amber) Configuration Error (red)
	RS232 (amber) Ethernet (amber) DeviceNet (green/red) Profibus Mode (green/red) Profibus Stat (green/red)

The features given are typical at a 25 °C ambient temperature (if not otherwise indicated).