STB3574

Industrial Ethernet Cradle

Developer Guide

ZEBRA

MN-002694-01

STB3574 INDUSTRIAL ETHERNET CRADLE DEVELOPER GUIDE

MN-002694-01

Revision A

May 2016

Revision History

Changes to the original guide are listed below:

Change	Date	/2016Description
Rev. A	5/2016	Initial Release

TABLE OF CONTENTS

ABOUT THIS GUIDE

Introduction

This guide provides information on the use of the Industrial Ethernet interface on the STB3574 scanner cradle.

✓ ***NOTE*** Screens and windows pictured in this guide are samples and can differ from actual screens.

Chapter Descriptions

Topics covered in this guide are as follows:

- *Chapter 1, Initial Setup and Industrial Ethernet Software*, provides information about how to perform setup and configure the STB3574.
- *Chapter 2, Ethernet Interface*, provides an overview of the Industrial Ethernet (IE) interface.
- *Chapter 3, Bar Code Transfer*, provides the details of how this transfer is accomplished, independent of the protocol being used.
- *Chapter 4, PROFINET Interface*, provides information about PROFINET IO modules including the steps and screen shots to configure the Siemens S7 communications, and transferring bar code data from the cradle.
- *Chapter 5, Ethernet/IP Interface*, provides information about I/O assemblies, I/O connections, steps to configure Rockwell ControlLogix communication, and transferring bar code data from the cradle.
- *Chapter 6, Modbus TCP Interface*, provides information about using the Modbus TCP interface to transfer bar code data from the cradle.
- *Chapter 7, Restoring Settings to Factory Defaults* provides information and parameter bar codes that can be used to reset the cradle Ethernet settings back to factory defaults.
- *Appendix A, Troubleshooting* provides the symptoms and solutions to common issues.

Notational Conventions

The following conventions are used in this document:

- *Italics* are used to highlight the following:
  - Chapters and sections in this guide
  - Related documents
- **Bold** text is used to highlight the following:
  - Dialog box, window and screen names
  - Drop-down list and list box names
  - Check box and radio button names
  - Icons on a screen
  - Key names on a keypad
  - Button names on a screen.
- Bullets (•) indicate:
  - Action items
  - Lists of alternatives
  - Lists of required steps that are not necessarily sequential.
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.

Related Documents and Software

Documentation

- *DS3578 With FIPS Digital Scanner Product Reference Guide* (p/n 72E-153466-xx) provides general instructions for setting up, operating, maintaining, and troubleshooting the digital scanner.
- *DS3578 with FIPS Quick Start Guide* (p/n 72-151247-xx) provides information to get the digital scanner up and running; provides Regulatory and Waste Electrical and Electronic Equipment information.
- *STB3578 With FIPS/STB3574 Cradle Quick Reference Guide* (p/n 72-150220-xx) provides basic instructions on setting up and using the STB3578/STB3574 cradles; provides Regulatory and Waste Electrical and Electronic Equipment information.

Software

For the latest version of the Zebra Industrial Ethernet Software, and the latest versions of all guides, go to: http://www.zebra.com/support.

Service Information

If you have a problem with your equipment, contact Zebra Global Customer Support for your region. Contact information is available at: http://www.zebra.com/support.

When contacting support, please have the following information available:

- Serial number of the unit
- Model number or product name
- Software type and version number.

Zebra responds to calls by email, telephone or fax within the time limits set forth in support agreements.

If your problem cannot be solved by Zebra Customer Support, you may need to return your equipment for servicing and will be given specific directions. Zebra is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your Zebra business product from a Zebra business partner, contact that business partner for support.

CHAPTER 1 INITIAL SETUP AND INDUSTRIAL ETHERNET SOFTWARE

Introduction

This chapter provides information about how to perform initial setup of the STB3574 and the Zebra Industrial Ethernet Software Package that provides configuration capabilities. The latest version of this software can be downloaded from the Zebra support website at: http://www.zebra.com/support.

Hardware/Software Prerequisites

The following components are required for initial setup and a fully functional Zebra Industrial Ethernet solution.

- STB3574 Zebra Industrial Ethernet Cradle (cradle with 9v power supply) - p/n STB3574-C100F7WW.
- A compatible Zebra cordless scanner (such as a DS3578).
- A PC running Windows 7, or higher.
- An Industrial Ethernet PLC (Programmable Logic Controller) that supports one of the support protocols (EtherNet/IP, PROFINET, or Modbus TCP).

IMPORTANT Industrial Ethernet testing was performed with the following PLCs and software:
- Siemens SIMATIC S7-1200 PLC (6ES7 215-1AG40-0XB0) and TIA v13 SP1 Software
- Rockwell Compact Logix L24ER (QB1B) and Logix Studio 5000 v24 Software

- Ethernet switch or router (if not connecting STB3574 directly to a PLC) and Ethernet cables.
- Zebra Industrial Ethernet Software Package.
  - The software package includes the following components:
    - Zebra Industrial Ethernet Configuration Utility
    - Device integration (GSD, EDS) files to support (PLC) application development
    - Sample applications to help accelerate development.

Initial Setup

Industrial Ethernet Software Installation

To verify STB3574 functionality and view scanned bar codes, it is recommended that the Zebra Industrial Ethernet Software be used initially. This software requires a PC running Windows 7, or higher. To install, double click the install package and follow the InstallShield prompts. Upon completion of the installation process, a new Program Files folder is created that includes shortcuts to the locations of sample applications and device definition files along with a shortcut to the Zebra Configuration Utility.

STB3574 Factory Default Settings

The factory default configuration of the STB3574 is as follows:

- Active Industrial Protocol is set to EtherNet/IP.
- IP Address mode: DHCP (Dynamic Host Control Protocol) with a 30 second timeout.

✓ ***NOTE*** The STB3574 factory default Ethernet IP address configuration has DHCP enabled, and a 30 second timeout on DHCP. In the event that the DHCP address is not received within the timeout value of 30 seconds, the device falls back to an AutoIP address of 192.168.0.100.

Connecting to the STB3574 for the First Time

Before changing the configuration, or developing industrial applications for the Zebra industrial device (STB3574), the utility should be verified to ensure it works within your network infrastructure. The following steps illustrate the recommended way to connect to the STB3574.

1. Connect the STB3574 to the network using an Ethernet cable.

✓ ***NOTE*** It is expected that the network contains a DHCP server so that the STB3574 can obtain an initial IP address for configuration.

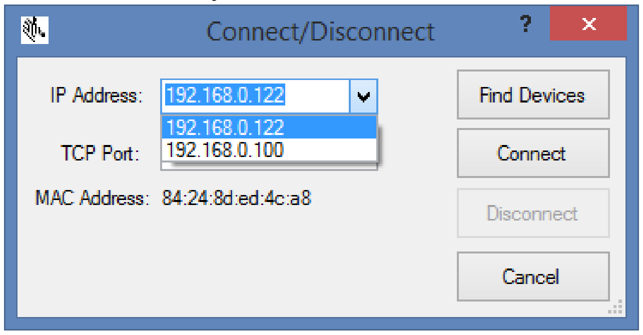
2. Power the STB3574 using the 9v power supply.

a. The blue LED should be lit to indicate that the STB3574 is powered and working properly.

3. Verify that the Windows 7, or higher PC running the Zebra Industrial Ethernet Configuration Utility is on the same network as the STB3574.

4. Run the Zebra Industrial Ethernet Configuration Utility by clicking on its shortcut under Program Files > Zebra Scanner > Industrial Ethernet Software.

5. From the Utility, click the **Connect/Disconnect** button. The following dialog displays.



IP Address: 192.168.0.122
192.168.0.122
192.168.0.100
TCP Port:
MAC Address: 84:24:8d:ed:4c:a8

Figure 1-1 *Connect/Disconnect Dialog*

6. Click **Find Devices** to find all STB3574 devices on the network. If any devices are found, their IP addresses display in the IP Address drop-down list.

✓ ***NOTE*** This action sends out a broadcast message that is responded to by any/all Zebra Industrial Ethernet cradles on the network. The first IP address in the list may NOT be the device you want to configure. To ensure you have the correct IP address, check the MAC address and match that to the cradle you are attempting to configure.

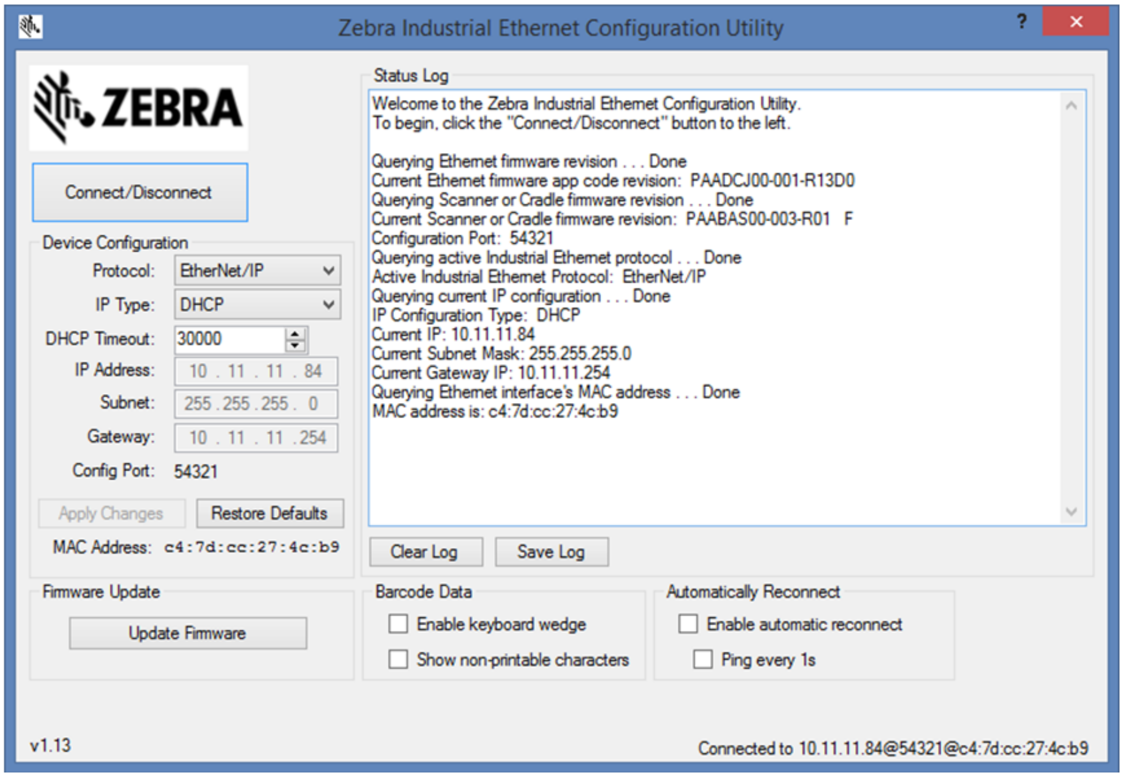
7. From the **IP Address:** drop-down list, choose the IP address of the device to which you want to connect and click **Connect**.

   If the connection fails, or IP addresses do not populate when **Find Devices** is clicked see *Appendix A, Troubleshooting*.

Zebra Industrial Ethernet Configuration Utility

The Zebra Industrial Ethernet Configuration Utility provides an application that can be used to configure protocol and IP addresses of Zebra Industrial Ethernet devices as well as test and perform basic scanning operations.

After installing, running, and connecting to an STB3574 as described in *Initial Setup on page 1-2*, a screen similar to *Figure 1-2* displays. Once connected, all fields are populated with current information and the Status Log is updated with specific information pulled from the cradle, including firmware versions.



Protocol: EtherNet/IP
IP Type: DHCP
DHCP Timeout: 30000
IP Address: 10 . 11 . 11 . 84
Subnet: 255 . 255 . 255 . 0
Gateway: 10 . 11 . 11 . 254
Config Port: 54321
MAC Address: c4:7d:cc:27:4c:b9

```
Welcome to the Zebra Industrial Ethernet Configuration Utility.
To begin, click the "Connect/Disconnect" button to the left.

Querying Ethernet firmware revision . . . Done
Current Ethernet firmware app code revision:  PAADCJ00-001-R13D0
Querying Scanner or Cradle firmware revision . . . Done
Current Scanner or Cradle firmware revision:  PAABAS00-003-R01   F
Configuration Port:  54321
Querying active Industrial Ethernet protocol . . . Done
Active Industrial Ethernet Protocol:  EtherNet/IP
Querying current IP configuration . . . Done
IP Configuration Type:  DHCP
Current IP: 10.11.11.84
Current Subnet Mask: 255.255.255.0
Current Gateway IP: 10.11.11.254
Querying Ethernet interface's MAC address . . . Done
MAC address is: c4:7d:cc:27:4c:b9
```

☐ Enable keyboard wedge
☐ Show non-printable characters
☐ Enable automatic reconnect
☐ Ping every 1s

Connected to 10.11.11.84@54321@c4:7d:cc:27:4c:b9

Figure 1-2 *Zebra Industrial Ethernet Configuration Utility Home Screen*

Verifying Reception of Bar Code Data

To ensure bar code data was received:

1. Ensure the DS3578, or equivalent, scanner is paired to the STB3574 cradle by scanning the pairing bar code on the cradle. (Refer to the *DS3578 With FIPS Digital Scanner Product Reference Guide*, p/n 72E-153466-xx, for more information about pairing a scanner to a cradle.)
2. Follow the steps in *Connecting to the STB3574 for the First Time on page 1-2* to open a connection to the STB3574 cradle, and verify that a connection was made. The connection status displays Connect to <device information>, where <device information> includes the IP address, port, and MAC address of the cradle. This displays on the bottom right of the utility.
3. Scan bar codes with the DS3578, or equivalent, paired scanner. The bar code data displays in the Status Log on the Zebra Industrial Ethernet Configuration Utility screen (*Figure 1-2*).

✓ ***NOTE*** In the event that a valid connection does not exist, the scanner still decodes a bar code but it times outs, emits four reds LED flashes, and sounds an error beep. This error occurs when the scanner does not have an active connection to send bar code data. Refer to your scanner Product Reference Guide for more information about error conditions.

Setting the Active Industrial Ethernet Protocol

The STB3574 Industrial Ethernet Cradle supports three protocols:

- EtherNet/IP
- PROFINET
- Modbus TCP.

Only one of these protocols can be used at a time. This section describes how to set the active protocol.

✓ ***NOTE*** If the current protocol selected on the Zebra Industrial Ethernet Configuration Utility screen is the protocol you wish to use no changes are necessary and the following steps can be skipped.

To set the active protocol:

1. Ensure that a valid connection between the cradle and the Industrial Ethernet Configuration Utility exists.
2. Select the Industrial Ethernet Protocol from the Protocol: drop-down list. Selecting a new protocol from the list activates the **Apply Changes** button.
3. Click the **Apply Changes** button to save the protocol change.

✓ ***NOTE*** The change in IE protocol does not take effect until the module is reset.

Configuring an IP Address

The default IP address configuration for the STB3574 cradle uses DHCP to obtain the IP address. This can be re-configured to use a Static IP address, if desired.

To configure a static address:

1. Ensure that a valid connection between the cradle and the Industrial Ethernet Configuration Utility exists.

2. Select Static from the IP Type: drop-down list. Selecting a new IP type from the list changes the IP address, Subnet, and Gateway fields to writable fields, and activates the **Apply Changes** button.
3. Modify the IP address, Subnet, and Gateway fields, as needed.
4. Click the **Apply Changes** button to save all changes.

✓ ***NOTE*** After clicking **Apply Changes** the utility is disconnected from the device. Follow the steps in *Connecting to the STB3574 for the First Time on page 1-2* to reconnect to the STB3574 cradle using the new IP address configuration.

Configuring the DHCP

When configured for DHCP, the cradle expects the DHCP server to respond in a timely manner. By default, the DHCP timeout is set to 30 seconds, and 30000 (milliseconds) displays in the DHCP Timeout field. Depending on the DHCP server location and network environment, this timeout can be lengthened or shortened using the DHCP timeout setting.

In the event that a timeout occurs (the DHCP server does not respond within the timeout period), the cradle falls back to the AutoIP address 192.168.0.100.

Advanced Features

The Zebra Industrial Ethernet Configuration Utility supports a number of advanced features to help with initial setup and testing.

Status Log

The Status Log provides information about initial connection, settings changes, and displays bar code data. The Status Log can be cleared using the **Clear Log** button, or saved using the **Save Log** button.

Bar Code Data

The Barcode Data group box provides two check boxes.

- **Enable keyboard wedge**: When checked all bar code data is sent to the Status Log, and also sent to the top most window. This feature can be used to send bar code data to a document, or spreadsheet.
- **Show non-printable characters**: When checked any non-printable characters are converted to a readable string. This feature is useful when checking that non-printable characters (such as a carriage return "\r") are scanned and received correctly.

Ping Every 1 Second

When connected, the Configuration Utility can ping the STB3574 on a 1 sec interval to ensure a connection is working. In the event that ping replies are not received, the utility disconnects.

Automatically Reconnect

When the Enable automatic reconnect check-box is checked, the utility attempts to reconnect to the cradle automatically until either the check-box is unchecked, a manual disconnect occurs, or the utility is closed out. Note that when automatic reconnect is enabled, the Ping every 1 second check-box is automatically checked, since automatic reconnect relies on it.

Firmware Update

Firmware updates to the STB3574 cradle may be available on the Zebra support website. To update the firmware, download the latest firmware load by clicking the **Load Firmware** button, and opening it with the *Open* dialog.

Restore Defaults

Device configuration settings can be reset to factory defaults by clicking the **Restore Defaults** button. A confirmation dialog appears to confirm the action before the reset is complete.

✓ ***NOTE*** This feature can only be used if a valid connection to the cradle can be made. If the IP address settings are such that a connection to the utility cannot be made, see *Chapter 7, Restoring Settings to Factory Defaults* for information about how to use parameter bar codes to perform a factory reset.

Device Definition Files

The Zebra Industrial Ethernet Software Package includes device definition files required to develop industrial Ethernet applications that run on PLCs. *Table 1-1* lists the files.

Table 1-1 *Definition File Listing*

Industrial Protocol (Development Environment)	**Device Definition File**	**Image File**	**Folder Location**
EtherNet/IP (Logix 5000 Studio)	Zebra_STB3574.eds	Zebra_STB3574.ico	Zebra Scanner\Industrial Ethernet Software\EtherNetIP\
PROFINET (Totally Integrated Automation)	GSDML-V2.31-Zebra-STB3574-1-20151123.xml	GSDML-034B-0001-STB3574.bmp	Zebra Scanner\Industrial Ethernet Software\PROFINET\

For detailed information about how to install these device definition files see the appropriate chapter for your Industrial Protocol.

Sample Application Files

To aid in initial application development, sample applications are provided as part of the Zebra Industrial Ethernet Software Package. The sample applications are located in the folders that match their respective protocols.

- EtherNet/IP samples are located underneath the folder Zebra Scanner\Industrial Ethernet Software\EtherNetIP\.
- PROFINET samples are located underneath the folder Zebra Scanner\Industrial Ethernet Software\PROFINET\.
- Modbus TCP sample is located underneath the folder Zebra Scanner\Industrial Ethernet Software\PROFINET\.

✓ ***NOTE*** These files must be installed into the appropriate development environment for the sample applications to work correctly.

CHAPTER 2 ETHERNET INTERFACE

Introduction

This chapter provides an overview of the Industrial Ethernet (IE) interface.

Industrial Protocol Support

The cradle Industrial Ethernet interface supports the following standard protocols:

- PROFINET (for details, see *Chapter 4, PROFINET Interface*)
- EtherNet/IP (for details, see *Chapter 5, Ethernet/IP Interface*)
- Modbus TCP (for details, see *Chapter 6, Modbus TCP Interface*).

Selecting the Active Industrial Ethernet Protocol

The cradle supports one Industrial Ethernet protocol at a time. The active protocol is selected with the Industrial Ethernet Configuration Utility, and any selection change takes effect after the cradle is power cycled.

For more information about how to configure the Active IE protocol see *Chapter 1, Initial Setup and Industrial Ethernet Software*.

Setting IP Address Configuration

In order to communicate with the cradle over Ethernet, the IP address must be set to a valid address on the same subnet as the computer, or controller used for communication. This may be done using DHCP, or the address may be configured statically.

PROFINET does not require the cradle to have an IP address configuration to start communication on the network; the controller may be configured to set the IP address of the devices based on the PROFINET name. The IP configuration need not be set if the active protocol is PROFINET. However, if another protocol is selected, or if non-industrial tools are to be used for functions like firmware updates or other non-I/O functions, the cradle must have an IP address configured for these purposes.

The IP address configuration used by the Ethernet interface may be set with the Industrial Ethernet Configuration Utility. Any change to the configuration takes effect immediately.

See *Chapter 1, Initial Setup and Industrial Ethernet Software* for configuration procedures.

CHAPTER 3 BAR CODE TRANSFER

Introduction

The transfer of bar code data from the cradle to the controller is performed using the same mechanism for all Industrial Ethernet protocols.

This chapter provides the details of how this transfer is accomplished, independent of the protocol being used.

Transfer Modes

The cradle provides three modes of bar code transfer mechanisms. The three modes provide increasing level of data integrity and data size capabilities.

Basic Mode

When using Basic Mode transfer, the bar code data is sent to the controller as it is received. There are no checks to verify that the controller has read or processed the data; new data overwrites previous data when it is received by the cradle from the scanner.   Basic Mode is the simplest transfer to implement as it does not require logic in the controller to interact with the cradle.

Handshake Mode

Handshake Mode adds a level of data integrity to the bar code transfer as the controller must acknowledge each piece of bar code data. The cradle will not send new data until the controller indicates that it has completed processing the previous data. Bar codes that are received from the scanner by the cradle while the controller is busy are cached in the cradle until they can be transferred to the controller.

Fragmentation Mode

Some of the Industrial Ethernet protocols have limited message buffer sizes.   Fragmentation Mode allows bar code data that is larger than the protocol message buffer to be transferred to the controller. When using Fragmentation Mode if a bar code is larger than the message buffer it will be broken up and sent to the controller in multiple blocks and the controller will re-assemble the blocks to create the complete bar code.

Fragmentation Mode requires the use of Handshake Mode to guarantee that each fragment block is received by the controller.

Bar Code Caching

Bar codes received from the scanner by the cradle will be cached until they are able to be transferred to the controller. This alleviates the possibility of losing bar code data that is read when a transfer of a previously read bar code is in progress with the controller.

The cradle will cache up to 10 bar codes, or 12K bytes of bar code data, whichever comes first (i.e., if bar codes are small, 10 bar codes may be cached, if they are large, less than 10 may be cached if their total size reaches 12K).

If the bar code cache is full and a new bar code is received by the cradle, the oldest bar code in the cache will be discarded. In the case where a bar code is lost from the cache, the Cache Overflow bit in the Status Register will be set to notify the controller of the issue.

I/O Data Format

The I/O data exchanged between the cradle and the controller carries the bar code data itself as well as a collection of status and control information used for the transfer of the bar code.

Status and Bar Code Input Data

The Status and Bar Code Input Data is sent from the cradle to the controller. The input data holds the current status of the bar code transfer and the bar code data itself. The format of the input data is described below.

Table 3-1 *Status and Bar Code Input Format*

Status and Bar Code Input Data Format				
Offset	**Length**	**Parameter**	**Data Type**	**Description**
0	2	Status Register	Bit string	Bit string indicating the current status of the cradle interface and bar code data transfer. See the bit assignment details in the table below.
2	2	Update Counter	UINT16	Value that indicates to the controller that the bar code data has been updated. This is incremented by 1 each time new bar code data is loaded into the Data field. Normal range is 1-65535. zero (0) indicates that a transfer error has been detected by the cradle and the controller must resynchronize.
4	2	Length	UINT16	The length of the data in the Data field in bytes.

Table 3-1 *Status and Bar Code Input Format (Continued)*

Status and Bar Code Input Data Format				
Offset	**Length**	**Parameter**	**Data Type**	**Description**
6	2 (EIP) 0 (PB/MB)	Reserved	BYTE[2/0]	Pad bytes required by EtherNet/IP. **Note:** This field is not included for PROFINET and Modbus TCP.
8 (EIP) 6 (PB/MB)	---	Data	BYTE[]	The bar code data. This may be an entire bar code or a fragment block as indicated by the Status Register. The overall length of the Data field is dependent on the protocol being used and will be specified in the individual protocol sections below. Values in the Data field beyond Length bytes will be set to 0.

Table 3-2 *Status Register Bit Assignments*

Status Register Bit Assignments		
Bit	**Parameter**	**Description**
0	Bar Code Transfer	Set if bar code Transfer is enabled.
1	Handshake Mode	Set if Handshake Mode is enabled.
2	Fragmentation Mode	Set if bar code Fragmentation Mode is enabled.
3	Bar Code Cache Overflow	Set when the bar code cache is full and a new bar code is received from the scanner by the cradle. This indicates that the data for one or more bar codes have been lost. This bit is “sticky” and is cleared by setting the Clear Faults bit in the Output Control Register.
4	Input Data Overflow	Set if Fragmentation Mode is not enabled and the current bar code data does not fit in the Data. The current bar code data has been truncated to fit in the Data field. This bit is not used if Fragmentation Mode is enabled.
5	Waiting for Handshake	Indicates that the cradle is waiting for the Update/ACK Counter handshake. The bit is set when the cradle is waiting for the ACK Counter to be updated by the controller. It is cleared when the ACK Counter has been set to match the Update Counter. This bit is not used if Handshake Mode is disabled.
6-7	Reserved	Set to zero (0).
8	Bar Code Fragmented	Set if the current bar code data is larger than the Data field and is being sent in blocks. Cleared if the current bar code data fits in the Data field. This bit is not used if Fragmentation Mode is disabled.
9	First Fragment	Set on the first block of a fragmented bar code transfer. This bit is not used if Fragmentation Mode is disabled.

Table 3-2 *Status Register Bit Assignments (Continued)*

Status Register Bit Assignments		
Bit	**Parameter**	**Description**
10	Middle Fragment	Set on all blocks of a fragmented bar code transfer except the first and last. This bit is not used if Fragmentation Mode is disabled.
11	Last Fragment	Set on the last block of a fragmented bar code transfer. This bit is not used if Fragmentation Mode is disabled.
12-15	Reserved	Set to zero (0).

Bar Code Transfer Control Output Data

The Bar Code Transfer Control Output Data is sent from the controller to the cradle. The output data is used for handshaking during bar code transfer by the controller. The format of the output data is described below.

Table 3-3 *Bar Code Transfer Control Output Data Format*

Bar Code Transfer Control Output Data Format				
Offset	**Length**	**Parameter**	**Data Types**	**Description**
0	2	Control Register	Bit string	Bit string used for bar code transfer control. See the bit assignment details in the table below.
2	2	ACK Counter	UINT16	Counter used for transfer handshaking. The value is set to match the Input data Update Counter to indicate that the controller has read the Data from the input data. Normal range is 1-65535. Zero (0) indicates that a transfer error has been detected by the controller and the cradle must resynchronize. This field is ignored by the cradle if Handshake Mode is disabled.

Table 3-4 *Control Register Bit Assignments*

Control Register Bit Assignments		
Bit	**Parameter**	**Description**
0	Bar Code Transfer	Set to enable Bar Code Transfer. If cleared, bar code data will be cached by the cradle but will not be transferred to the controller. A bar code transfer that is in progress when Bar Code Transfer is disabled will be resent after it is enabled.
1	Handshake Mode	Set to enable Handshake Mode. The status of this bit is only read on the rising edge of the Bar Code Transfer bit; any changes made otherwise are ignored (i.e., the Handshake Mode is set when the Bar Code Transfer is enabled).
2	Fragmentation Mode	Set to enable Bar Code Fragmentation Mode. The status of this bit is only read on the rising edge of the Bar Code Transfer bit; any changes made otherwise are ignored (i.e., the Fragmentation Mode is set when the Bar Code Transfer is enabled). This bit will be ignored if Handshake Mode is not set.
3	Reserved	Set to zero (0).
4	Clear Faults	Set to clear any active interface faults. This bit is only effective on the rising edge.
5-15	Reserved	Set to zero (0).

Reading Bar Code Data from the Cradle

Enabling Bar Code Transfer

Bar code data transfer from the cradle to the controller must be enabled by the controller. Transfer is enabled by setting the Bar code Transfer bit in the Control Register.

Transfer may be stopped by clearing the Bar Code Transfer bit. If transfer is disabled while a bar code transfer is in progress, that bar code will be cached and resent when transfer is enabled.

Selecting Transfer Mode

The type of transfer mode to be used (Basic, Handshake, Fragmentation) is selected by the controller using the Handshake Mode and Fragmentation Mode bits in the Control Register. These bit are only read by the cradle on the rising edge of the Bar Code Transfer bit; hence, transfer mode is only set when bar code transfer is enabled.

Table 3-5 describes the transfer mode selection with the Handshake Mode and Fragmentation Mode bits.

Table 3-5 *Transfer Mode Selection with Handshake Mode and Fragmentation Mode Bits*

Handshake Mode	Fragmentation Mode	Resulting Transfer Mode
OFF	OFF	Basic
ON	OFF	Handshake
ON	ON	Handshake with Fragmentation
OFF	ON	Invalid (defaults to Basic)

Handshake Mode is required for Fragmentation Mode. If Handshake Mode is not set when Fragmentation Mode is set, the combination is invalid and the transfer mode defaults to Basic.

Basic Mode

Transfer Control

In Basic Mode bar code data will be loaded into the Input Data buffer as it is read by the cradle.

When new data is loaded into the Input Data field, the Update Counter will be incremented by 1 to indicate new data is available. Basic Mode provides no acknowledgment mechanism from the controller to indicate that it has completed processing of the data.   There is no guarantee that bar code data in the Input Data buffer will not be overwritten before it is processed by the controller.

Bar Code Data

In Basic Mode each bar code is sent in a single block in the Input Data field to the controller.

If the bar code data is larger than the size of the Input Data field it is truncated to the available size and the Input Data Overflow bit is set in the Status Register to indicate that the bar code data is truncated.

Handshake Mode

Transfer Control

Handshake Mode provides an acknowledgment mechanism that allows the controller to acknowledge that the bar code data in the input buffer has been read.   This alleviates the possibility of the cradle overwriting the input data before it has been read by the controller.

When new data is loaded into the Input Data field, the Update Counter will be incremented by 1 to indicate new data is available. The Input Data field will not be overwritten by the cradle until the ACK Counter in the Output data is set to match the Update Counter.   The controller uses the change in the Update Counter to determine that data is available, and updates the ACK Counter once it has completed processing the data.

The Update Counter increments consecutively in the range of 1-65535. Zero (0) is a special value and is used to indicate that there is an issue with the bar code transfer.

Bar Code Data

In Handshake Mode (non-fragmented) each bar code is sent in a single block in the Input Data field to the controller.

If the bar code data is larger than the size of the Input Data field it is truncated to the available size and the Input Data Overflow bit is set in the Status Register to indicate that the bar code data is truncated.

Bar Code Transfer Error Handling

If the cradle detects an error in the bar code transfer (the connection was reset, transfer was disabled, etc.), the Update Counter will be set to zero (0) and any bar code transfer that is in progress will stop.

When the controller detects an error in the bar code transfer it should set the ACK Counter to zero (0). When the cradle detects that the ACK Counter has been set to zero (0) then any bar code transfer in progress will be stopped and the Update Counter set to zero (0).

Once the cradle has set the Update Counter to zero (0), it will wait for the ACK Counter to be set to zero (0) by the controller before proceeding. Once both counters are zero (0), any pending bar code transfer will be restarted.

Sequence Chart

Figure 3-1 illustrates the Handshake Mode mechanism when transferring two bar codes.

The chart does not show every I/O message, but shows the changes in the I/O message content. Depending on the I/O connection cyclic interval, a given message content may be repeated multiple times between the cradle and controller.

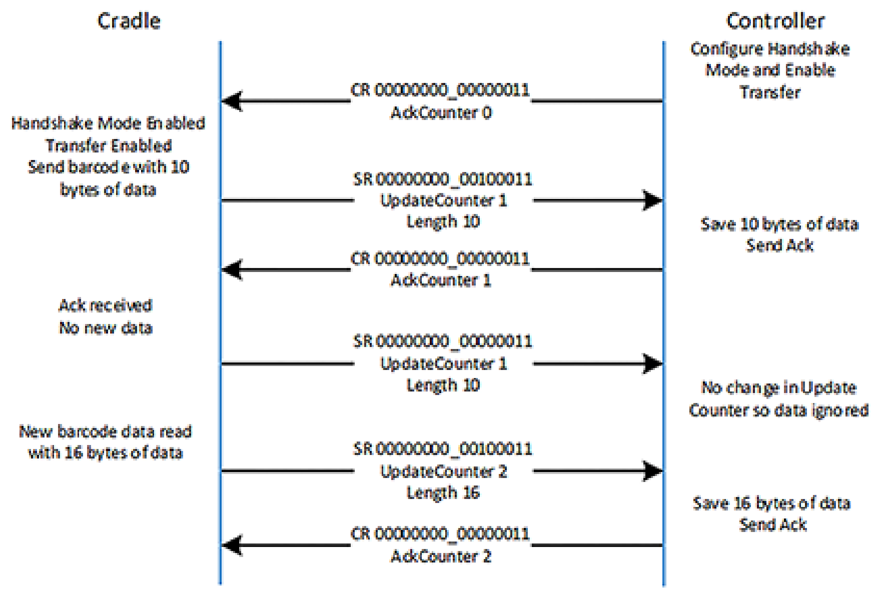

Cradle
Controller
Configure Handshake Mode and Enable Transfer
CR 00000000_00000011 AckCounter 0
Handshake Mode Enabled Transfer Enabled Send barcode with 10 bytes of data
SR 00000000_00100011 UpdateCounter 1 Length 10
Save 10 bytes of data Send Ack
CR 00000000_00000011 AckCounter 1
Ack received No new data
SR 00000000_00000011 UpdateCounter 1 Length 10
No change in Update Counter so data ignored
New barcode data read with 16 bytes of data
SR 00000000_00100011 UpdateCounter 2 Length 16
Save 16 bytes of data Send Ack
CR 00000000_00000011 AckCounter 2

Figure 3-1 *Handshake Mode Mechanism When Transferring Two Bar Codes.*

Fragmentation Mode

Transfer Control

Fragmentation Mode uses the Handshake Mode mechanism for data transfer control and adds the ability to handle bar code data that is larger than the Input Data field. See the Handshake Mode section above for complete details on transfer control.

Bar Code Data Fragmentation Control

When Fragmentation Mode is enabled, bar code data that is larger than the Input Data field will be broken up and sent in multiple blocks to the controller.

If the bar code data fits entirely in the Input Data field, the Bar Code Fragmented bit in the Status register will be cleared and the transfer mechanism described in the Handshake Mode section above will be used.

If the bar code is larger than the Input Data field, the following transfer mechanism is used.

- First block
  - Bar Code Fragmented and First Fragment bits in the Status Register are set.
  - Input Length is set to the overall size of the Input Data field.
  - The Input Data field contains the first size bytes of the bar code data, where size is the overall size of the Data field.
- Blocks 2 through (n-1)
  - Bar Code Fragmented and Middle Fragment bits in the Status Register are set.
  - Input Length is set to the overall size of the Input Data field.
  - The Input Data field contains the next size bytes of the bar code data, where size is the overall size of the Data field.
- Final block:
  - Bar Code Fragmented and Last Fragment bits in the Status Register are set
  - Input Length is set to remaining size of the bar code data
  - The Input Data field contains final portion of the bar code data

Sequence Chart

Figure 3-2 illustrates the mechanism used to transfer a 1000 byte bar code when the Input Data field size is limited to 400 bytes.

The chart does not show every I/O message, but shows the changes in the I/O message content. Depending on the I/O connection cyclic interval, a given message content may be repeated multiple times between the cradle and controller.

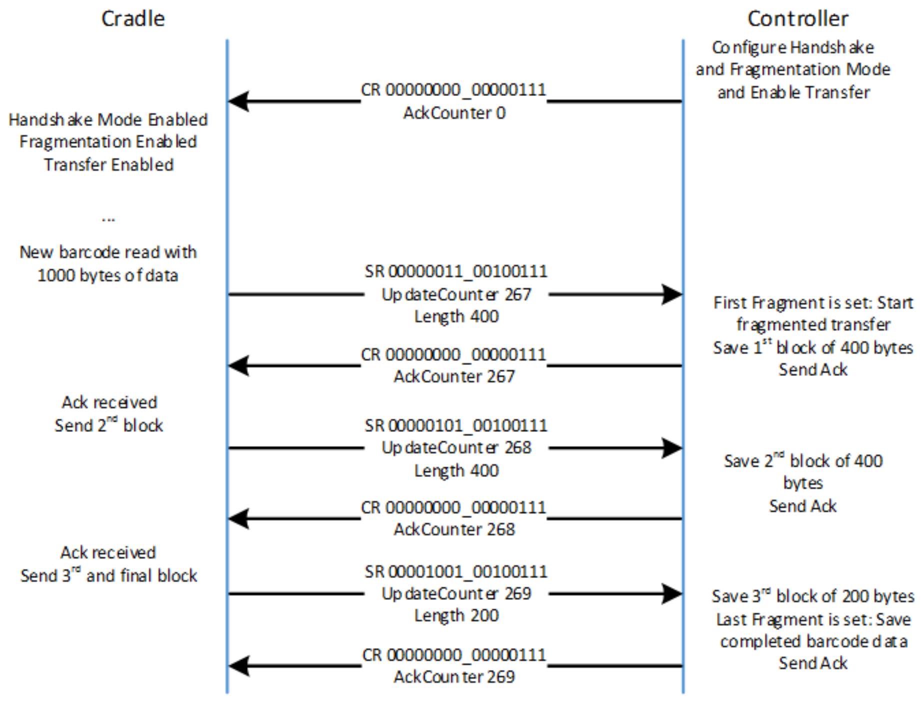

Cradle
Controller
Configure Handshake and Fragmentation Mode and Enable Transfer
CR 00000000_00000111 AckCounter 0
Handshake Mode Enabled Fragmentation Enabled Transfer Enabled
...
New barcode read with 1000 bytes of data
SR 00000011_00100111 UpdateCounter 267 Length 400
First Fragment is set: Start fragmented transfer Save 1st block of 400 bytes Send Ack
CR 00000000_00000111 AckCounter 267
Ack received Send 2nd block
SR 00000101_00100111 UpdateCounter 268 Length 400
Save 2nd block of 400 bytes Send Ack
CR 00000000_00000111 AckCounter 268
Ack received Send 3rd and final block
SR 00001001_00100111 UpdateCounter 269 Length 200
Save 3rd block of 200 bytes Last Fragment is set: Save completed barcode data Send Ack
CR 00000000_00000111 AckCounter 269

Figure 3-2 *Mechanism Transfer 1000 Byte Bar Code When Input Data Field Size Is Limited To 400 Bytes*

CHAPTER 4 PROFINET INTERFACE

Introduction

This chapter provides information about using the PROFINET communications on the cradle.

Communications Profile

The PROFINET interface on the STB3574 scanner cradle supports PROFINET-IO Device functionality. The device is able to receive, or be the target of, I/O connections from a PROFINET Controller, but is not able to originate connections itself.

The interface supports the Generic device profile. The interface is PROFINET Conformance Class A / RT - 1.

GSDML File

The PROFINET GSDML file describes the Identity and I/O capabilities of the STB3574 scanner cradle. The file is used by controller configuration tools to configure the I/O connections and data tags used to communicate with the cradle over the PROFINET network.

The latest GSDML file can be acquired in the Industrial Ethernet Software Package (see *Chapter 1, Initial Setup and Industrial Ethernet Software*).

Identification and Maintenance Functions

The PROFINET interface supports the I&M0, I&M1, I&M2 and I&M3 record interfaces which provide identification and maintenance information about the cradle.

PROFINET IO Modules

The I/O interface includes a single I/O module that contains parameters and data used in the transfer of bar code data to the controller.

Status and Bar Code Data IO Module

Module ID:	48
Submodule ID:	1
Access:	Input / Output
Input Size:	70 bytes
Output Size:	4 bytes

The Status and Bar Code Data IO module holds the current status of the bar code transfer and the bar code data itself as well as the transfer control information used by the Controller. The format of the module data follows that described in the *Status and Bar Code Data IO Module* section above.

The Input Data field size 64 bytes.

Configuring Siemens S7 Communications

Register the GSD File

Before the communications to the cradle can be configured, the GSD file must be registered with TIA Portal. This only has to be done once.

The steps to register the GSD file using TIA Portal:

1. In the **Project View** menu, select: Options > Install General Station Description File (GSD).
2. Browse to the location of, and select the STB3574 GSD file.
3. Click **Install**.

Adding the Cradle to the I/O Configuration

In order for the controller to communicate with the cradle, it must be added to the I/O configuration in the controller program.

To add the cradle to the controller's I/O configuration using TIA Portal:

✓ ***NOTE*** These steps assume a PLC and a PROFINET network were configured in the project.

1. Open the **Network View** tab to show the PLC and the PROFINET network.

2. Find the cradle in the hardware catalog.

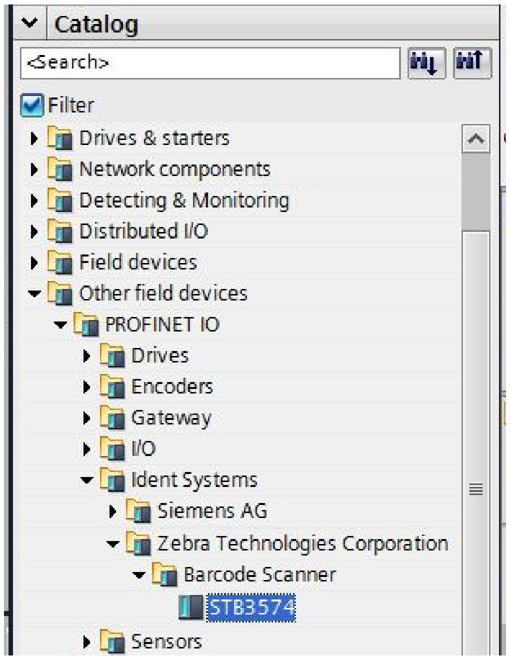


Figure 4-1 *Hardware Catalog Showing Cradle*

3. Select the cradle from the hardware catalog and drag it into the Network View.

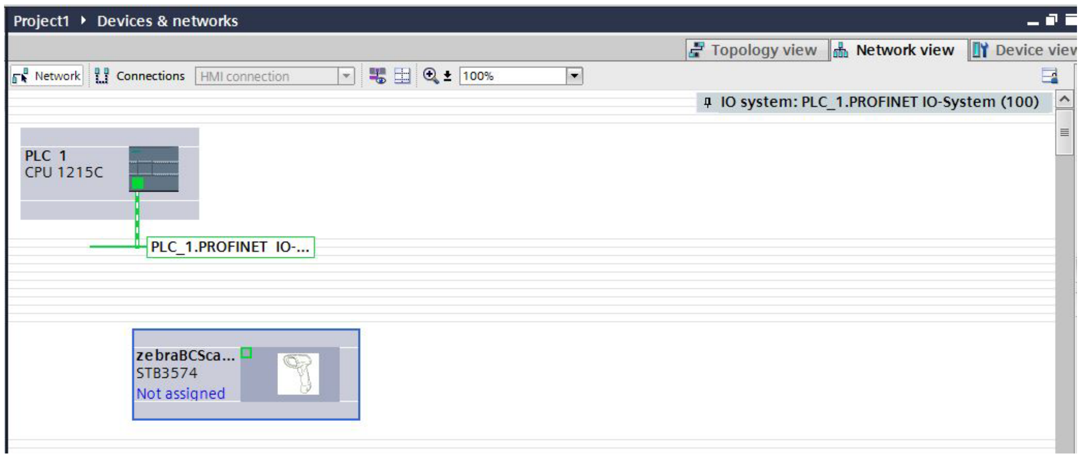
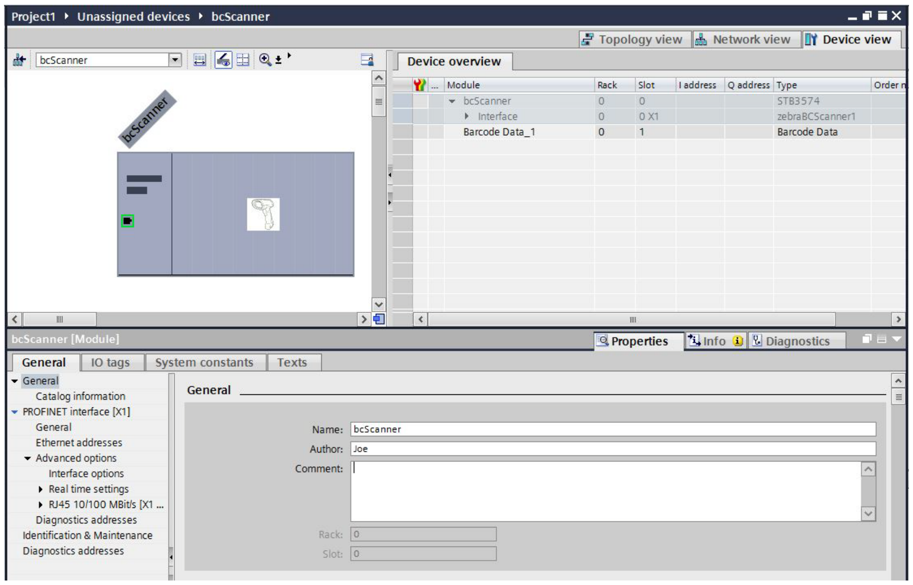


Figure 4-2 *Network View with Cradle Added*

4. Double-click on the cradle in the Network View. The Device View tab will be displayed for the cradle, with the module properties displayed below the Device View.

5. Select the General tree entry in the Properties tab and set the name that will be used to reference the module in the S7 program. The name defaults to zebrascanner1 but can be changed to any valid name that makes sense for the application. In the example the Name is set to bcScanner.



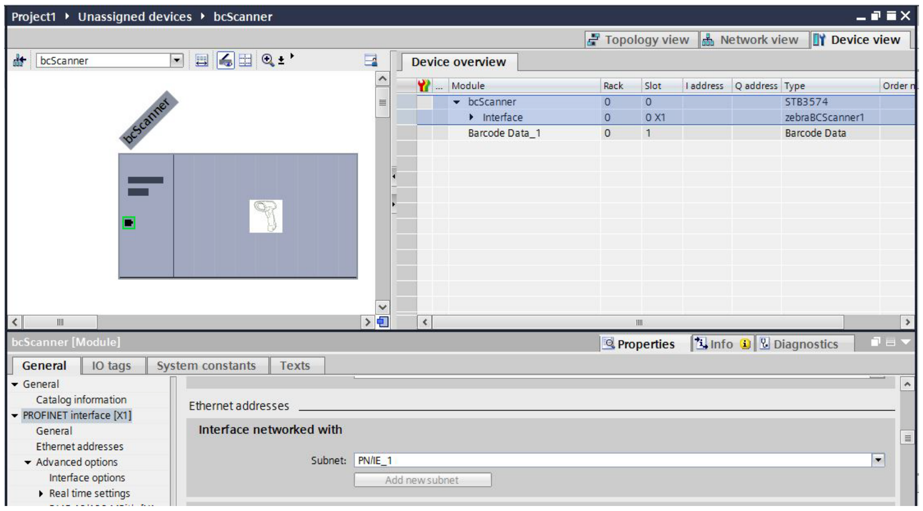
Module	Rack	Slot	I address	Q address	Type	Order n
bcScanner	0	0			STB3574	
Interface	0	0 X1			zebraBCScanner1	
Barcode Data_1	0	1			Barcode Data	

General

Name: bcScanner
Author: Joe
Comment:
Rack: 0
Slot: 0

Figure 4-3 *Cradle Properties, General*

6. Select the PROFINET Interface tree entry in the Properties tab and set the Subnet to the PROFINET network by selecting it in the drop-down box. In the example the network is named PN/IE_1.



Module	Rack	Slot	I address	Q address	Type	Order n
bcScanner	0	0			STB3574	
Interface	0	0 X1			zebraBCScanner1	
Barcode Data_1	0	1			Barcode Data	

Ethernet addresses

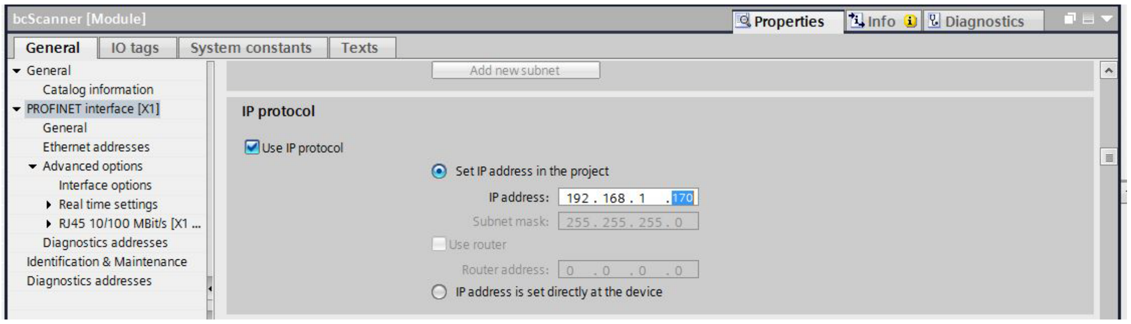
Interface networked with

Subnet: PN/IE_1

Figure 4-4 *Cradle Properties, PROFINET*

7. Scroll the PROFINET Interface properties down to show the IP Protocol properties. The IP address should be configured as required by the application.

   In the example below, the IP address is configured in the controller project and the cradle's IP address is set by the controller when the PROFINET communication is established. If the cradle's IP address is to be configured using another means (e.g., set statically in the device) the IP address is set directly radio button should be selected.



IP protocol

[✓] Use IP protocol

(●) Set IP address in the project

IP address: 192 . 168 . 1 . 170

Subnet mask: 255 . 255 . 255 . 0

[ ] Use router

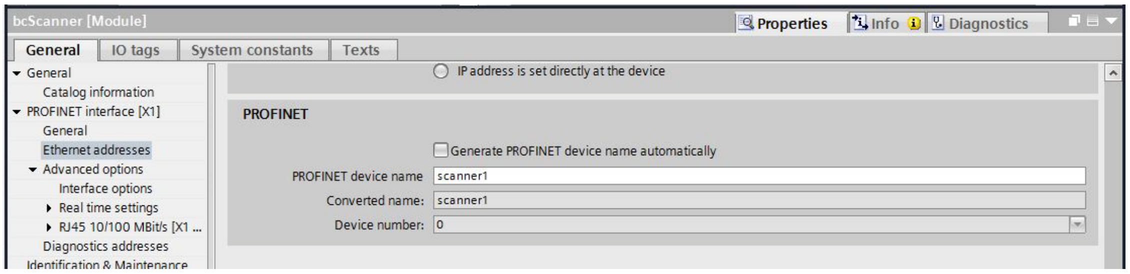
Router address: 0 . 0 . 0 . 0

( ) IP address is set directly at the device

Figure 4-5 *Cradle Properties, IP Configuration*

8. Scroll the PROFINET Interface properties down to show the PROFINET properties. The PROFINET device name should be configured as required by the application. This the name used by the Controller to locate and address the cradle on the network.

In the example, the name is set to scanner1. If Generate PROFINET device name automatically is checked the PROFINET device name will be set based on the name set in the General properties above.



( ) IP address is set directly at the device

PROFINET

[ ] Generate PROFINET device name automatically

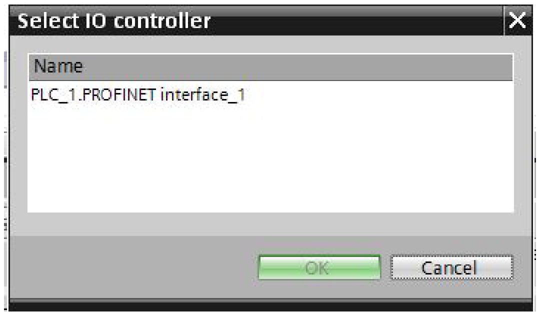
PROFINET device name: scanner1

Converted name: scanner1

Device number: 0

Figure 4-6 *Cradle Properties, PROFINET Device Name*

9. Return to the **Network view** tab. To add the cradle to the controller's I/O system, right-click on the Not Assigned text in the cradle module and select **Assign to New IO Controller**. The **Select IO Controller** dialog is displayed. Select the PLC and click **OK**.


Select IO controller

Name
PLC_1.PROFINET interface_1

Figure 4-7 *Select IO Controller*

The cradle is now assigned to the PLC I/O system and appears in the Distributed I/O tree under the PROFINET network.

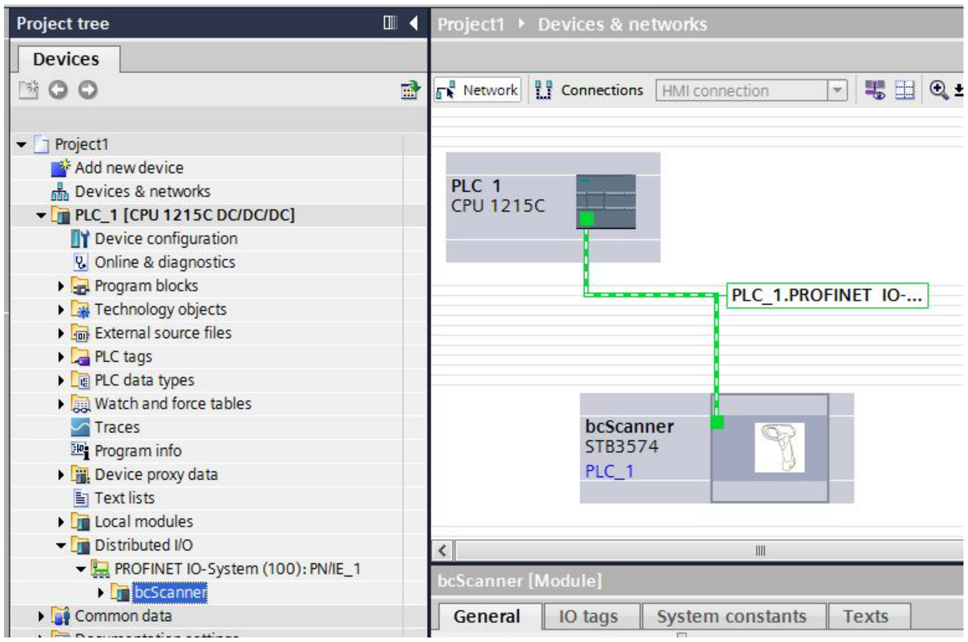
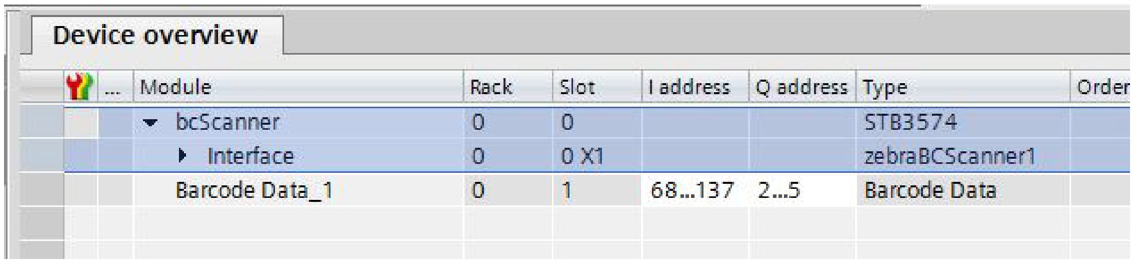


Figure 4-8 *Network View with Controller Set*

I/O Data Mapping

When the cradle is added to the I/O Configuration, input and output data is automatically mapped in the controllers I/O system based on the GSD file information. The I/O mapping can be viewed in TIA Portal by selecting the cradle module and going to the Device View tab. The Device Overview provides the input and output address mapping.



Device overview

	...	Module	Rack	Slot	I address	Q address	Type	Order
		bcScanner	0	0			STB3574	
		Interface	0	0 X1			zebraBCScanner1	
		Barcode Data_1	0	1	68...137	2...5	Barcode Data	

Figure 4-9 *Device Overview Tab*

In the example above, the input data has been mapped to addresses I68 - I567. These addresses hold data this is received from the cradle. The buffer is 500 bytes long and is formatted as described in Input Data table in the Status and Bar Code Data section.

The output data is mapped to addresses Q2 - Q5. These addresses hold data that will be sent to the cradle. The buffer is 4 bytes long and is formatted as described in the Output Data table the Bar Code Transfer Control Assembly section.

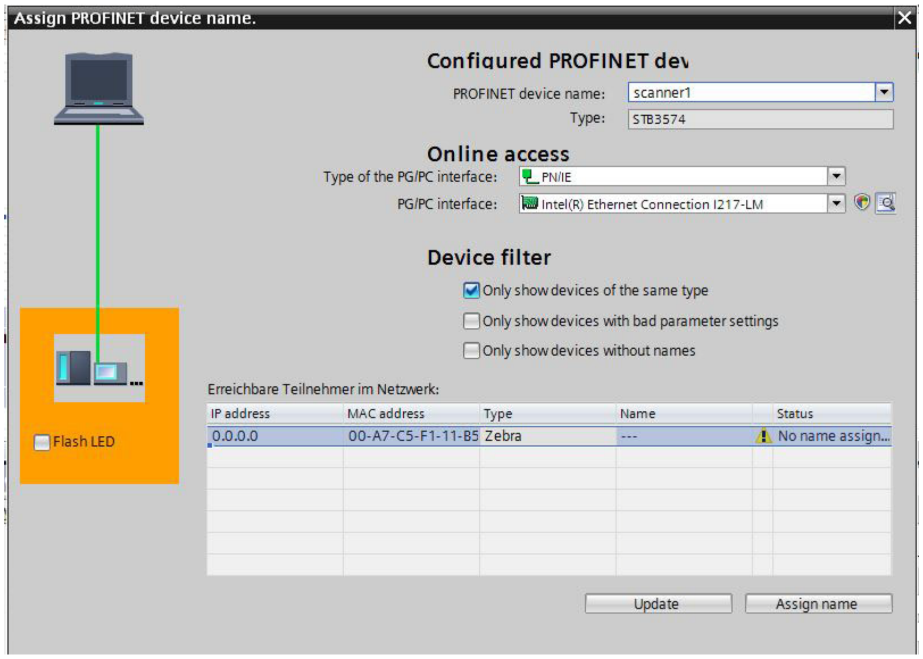
Setting the PROFINET Device Name

The PROFINET device name is used as an identification for devices, and is used to locate the device for communications purposes. The name must be set in the cradle module to match the PROFINET device name used in the controller configuration above in order for I/O communications to take place.

To set the cradle's name using TIA Portal:

1. Verify that the cradle is on the Ethernet network and is powered on.
2. In the **Network view**, right-click on the PROFINET network and select **Assign device name**. The **Assign PROFINET device name** dialog displays showing all devices on the network in the device list.
3. Select the cradle PROFINET device name from the **PROFINET device name** drop-down menu, in the example this is scanner1.

4. Select the desired cradle module in the device list and click **Assign Name**. This will assign the name in the upper drop-down menu to the actual cradle on the network.


Assign PROFINET device name.

Configured PROFINET dev

PROFINET device name: scanner1

Type: STB3574

Online access

Type of the PG/PC interface: PN/IE

PG/PC interface: Intel(R) Ethernet Connection I217-LM

Device filter

[✓] Only show devices of the same type

[ ] Only show devices with bad parameter settings

[ ] Only show devices without names

[ ] Flash LED

Erreichbare Teilnehmer im Netzwerk:

IP address	MAC address	Type	Name	Status
0.0.0.0	00-A7-C5-F1-11-B5	Zebra	---	No name assign...

Figure 4-10 *Assign PROFINET Name*

NOTE If there is more than one cradle on the network, the cradles can be identified by matching the MAC address shown in the device list with the MAC address on the cradle's label.

Transferring Bar Code Data from the Cradle

The PROFINET interface supports all three transfer modes for transferring bar code data from the cradle to the controller.

See *Reading Bar Code Data from the Cradle on page 3-5* for details on the bar code transfer mechanisms.

Example Bar Code Transfer Logic

The example transfer logic consist of simple controller programs for a Siemens S7 PLC that include the logic required to transfer bar codes from the cradle to a controller tag.

Three examples are provided:

- *Basic Mode Transfer Example Logic* (below)
- *Handshake Mode Transfer Example Logic on page 4-11*
- *Fragmentation Mode Transfer Example Logic on page 4-14*.

Basic Mode Transfer Example Logic

Folder Information

Table 4-1 includes the name of the folder for the Basic Mode transfer example. See *Chapter 1, Initial Setup and Industrial Ethernet Software* for more information about where to find sample applications.

Table 4-1 *Basic Mode Transfer Example Files*

Basic Mode Transfer Folder	Description
PROFINET\Samples\simpleBarcode	Siemens TIA project directory containing the Basic Mode example S7 program.

Using the Bar Code Transfer Logic

The bar code transfer logic included in the example program is encapsulated in a Function Block. This allows the bar code interface to be added easily to any program by simply adding a call to the Function Block. The Function Block will place the bar code information into a Data Block when it is received from the cradle.

Bar Code Handler Function Block

The Function Block used for the bar code transfer logic is the BarcodeHandler block. A Function Block call instruction should be added to the main block so that is can be executed each scan.   The BarcodeHandler block will handle all bar code transfer logic required to receive bar code information from the cradle.

Block Interface

The block interface for the BarcodeHandler function block contains the parameters in *Table 4-2* that must be set in the block call.

Table 4-2 *Parameters in the BarcodeHandler Function Block*

Parameter	Description
bcIn	The input data received from the cradle. This will be set to a tag that is mapped to the cradle PROFINET input data address.
bcOut	The output data to be sent to the cradle. This will be set to a tag that is mapped to the cradle PROFINET output data address.
EnableTransfer	A flag set to enable bar code transfer.

Data Types

The data types in *Table 4-3* were defined and are used in the bar code handling function block.

Table 4-3 *Bar Code Handling Function Block Data Types*

Data Type	Description
bcScannerInStruct	The structure of the PROFINET input data received from the cradle.
bcScannerOutStruct	The structure of the PROFINET output data sent to the cradle.

Bar Code Data Block

The BarcodeData block holds information about the bar code that was received from the cradle. *Table 4-4* includes the tags contained within the data block.

Table 4-4 *Data Block Tags*

Tag	Description
Counter	The current bar code update counter value received from the cradle. A change in this tag's value indicates that new bar code data has been received.
Length	The length in bytes of the bar code data that was received.
Data	The actual bar code data received from the cradle. Only *BarcodeLength* bytes of the buffer will be used. The total buffer size is 64 bytes.
Overflow	A flag indicating that the actual bar code data is longer than 64 bytes and was truncated by the PROFINET Interface in the cradle.

Bar Code Transfer Logic Details

Function Blocks

BarcodeHandler Function Block

The BarcodeHandler function block handles all bar code transfer logic. This subroutine should be called by the main block every scan.

The subroutine performs the following logic:

- Enable bar code transfer with the cradle using the output Control Register Bar Code Transfer bit.
- Monitor the Update Counter from the cradle for changes.
- If the Update Counter changes:
  - Store the Update Counter in the BarcodeData.Counter tag.
  - Store the Length in the BarcodeData.Length tag.
  - Store the Data field in the BarcodeData.Data tag.
  - Update the BarcodeData.Overflow tag based on the state of the Input Data Overflow bit in the input Status Register.

Tags

Global Tags

Table 4-5 *Global Tags*

Tag	Description
bcScannerIn	Mapped to the PROFINET input data from the cradle. This uses the *bcScannerInStruct* data type to define the data format.
bcScannerOut	Mapped to the PROFINET output data to the cradle. This uses the *bcScanerOutStruct* data type to define the data format.

BarcodeHandler Function Block Internal Tags

Table 4-6 includes the Static tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 4-6 *Internal Function Block Static Tags*

Tag	Description
EnableTransferEdge	Flag used to track the status of the *TransferEnable* parameter in the block interface.
LastUpdateCounter	Holds the previous value of the input Update Counter.
NewCodeEdge	Flag used to track the status *NewCodeRxed* temporary tag.

Table 4-7 includes the Temporary tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 4-7 *Internal Function Block Temporary Tags*

Tag	Description
NewCodeRxed	Flag that indicates that new bar code data has been received from the cradle. This flag is only active for a single scan when new data has been detected.

Handshake Mode Transfer Example Logic

Folder Information

Table 4-8 includes the name of the folder for the Handshake Mode transfer example. See *Chapter 1, Initial Setup and Industrial Ethernet Software* for more information about where to find sample applications.

Table 4-8 *Handshake Mode Transfer Example Files*

Basic Mode Transfer Folder	Description
PROFINET\Samples\barcodeHandshake	Siemens TIA project directory containing the Handshake Mode example S7 program.

Using the Bar Code Transfer Logic

The bar code transfer logic included in the example program is encapsulated in a Function Block. This allows the bar code interface to be added easily to any program by simply adding a call to the Function Block. The Function Block will place the bar code information into a Data Block when it is received from the cradle.

Bar Code Handler Function Block

The Function Block used for the bar code transfer logic is the BarcodeHandler block. A Function Block call instruction should be added to the main block so that is can be executed each scan. The BarcodeHandler block will handle all bar code transfer logic required to receive bar code information from the cradle.

Block Interface

The block interface for the BarcodeHandler function block contains the parameters in *Table 4-9* that must be set in the block call.

Table 4-9 *Parameters in the BarcodeHandler Function Block*

Parameter	Description
bcIn	The input data received from the cradle. This will be set to a tag that is mapped to the cradle PROFINET input data address.
bcOut	The output data to be sent to the cradle. This will be set to a tag that is mapped to the cradle PROFINET output data address.
EnableTransfer	A flag set to enable bar code transfer.

Data Types

The data types in *Table 4-10* were defined and are used in the bar code handling function block.

Table 4-10 *Bar Code Handling Function Block Data Types*

Data Type	Description
bcScannerInStruct	The structure of the PROFINET input data received from the cradle.
bcScannerOutStruct	The structure of the PROFINET output data sent to the cradle.

Bar Code Data Block

The BarcodeData block holds information about the bar code that was received from the cradle. *Table 4-11* includes the tags contained within the data block.

Table 4-11 *Data Block Tags*

Tag	Description
Counter	The current bar code update counter value received from the cradle. A change in this tag's value indicates that new bar code data has been received.
Length	The length in bytes of the bar code data that was received.
Data	The actual bar code data received from the cradle. Only *BarcodeLength* bytes of the buffer will be used. The total buffer size is 64 bytes.
Overflow	A flag indicating that the actual bar code data is longer than 64 bytes and was truncated by the PROFINET Interface in the cradle.

Bar Code Transfer Logic Details

Function Blocks

BarcodeHandler Function Block

The BarcodeHandler function block handles all bar code transfer logic. This subroutine should be called by the main block every scan.

The subroutine performs the following logic:

- Enable bar code transfer and handshake mode with the cradle using the output Control Register Bar Code Transfer and Handshake Mode bits.
- Handle Update Counter resets (when Update Counter is 0, ACK Counter should be reset to zero)
- Monitor for the Update Counter to differ from the ACK Counter to indicate new bar code data.
- If new bar code data is received:
  - Store the Update Counter in the BarcodeData.Counter tag.
  - Store the Length in the BarcodeData.Length tag.
  - Store the Data field in the BarcodeData.Data tag.
  - Update the BarcodeData.Overflow tag based on the state of the Input Data Overflow bit in the input Status Register
  - Update the ACK Counter to match the Update Counter.

Tags

Global Tags

Table 4-12 *Global Tags*

Tag	Description
bcScannerIn	Mapped to the PROFINET input data from the cradle. This uses the *bcScannerInStruct* data type to define the data format.
bcScannerOut	Mapped to the PROFINET output data to the cradle. This uses the *bcScanerOutStruct* data type to define the data format.

BarcodeHandler Function Block Internal Tags

Table 4-13 includes the Static tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 4-13 *Internal Function Block Static Tags*

Tag	Description
NewCodeEdge	Flag used to track the status *NewCodeRxed* temporary tag.

Table 4-7 includes the Temporary tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 4-14 *Internal Function Block Temporary Tags*

Tag	Description
NewCodeRxed	Flag that indicates that new bar code data has been received from the cradle. This flag is only active for a single scan when new data has been detected.

Fragmentation Mode Transfer Example Logic

Folder Information

Table 4-15 includes the name of the folder for the Fragmentation Mode transfer example. See *Chapter 1, Initial Setup and Industrial Ethernet Software* for more information about where to find sample applications.

Table 4-15 *Fragmentation Mode Transfer Example Files*

Fragmentation Mode Transfer Folder	Description
PROFINET\Samples\barcodeFragment	Siemens TIA project directory containing the Fragmentation Mode example S7 program.

Using the Bar Code Transfer Logic

The bar code transfer logic included in the example program is encapsulated in a Function Block. This allows the bar code interface to be added easily to any program by simply adding a call to the Function Block. The Function Block will place the bar code information into a Data Block when it is received from the cradle.

Bar Code Handler Function Block

The Function Block used for the bar code transfer logic is the BarcodeHandler block. A Function Block call instruction should be added to the main block so that is can be executed each scan.   The BarcodeHandler block will handle all bar code transfer logic required to receive bar code information from the cradle.

Block Interface

The block interface for the BarcodeHandler function block contains the parameters in *Table 4-16* that must be set in the block call.

Table 4-16 *Parameters in the BarcodeHandler Function Block*

Parameter	Description
bcIn	The input data received from the cradle. This will be set to a tag that is mapped to the cradle PROFINET input data address.
bcOut	The output data to be sent to the cradle. This will be set to a tag that is mapped to the cradle PROFINET output data address.
EnableTransfer	A flag set to enable bar code transfer.

Data Types

The data types in *Table 4-17* were defined and are used in the bar code handling function block.

Table 4-17 *Bar Code Handling Function Block Data Types*

Data Type	Description
bcScannerInStruct	The structure of the PROFINET input data received from the cradle.
bcScannerOutStruct	The structure of the PROFINET output data sent to the cradle.

Bar Code Data Block

The BarcodeData block holds information about the bar code that was received from the cradle. *Table 4-18* includes the tags contained within the data block.

Table 4-18 *Data Block Tags*

Tag	Description
Counter	The current bar code update counter value received from the cradle. A change in this tag’s value indicates that new bar code data has been received.
Length	The length in bytes of the bar code data that was received.
Data	The actual bar code data received from the cradle. Only *BarcodeLength* bytes of the buffer will be used. The total buffer size is 4096 bytes.
Overflow	A flag indicating that the actual bar code data is longer than 4096 bytes and was truncated by the PROFINET Interface in the cradle.

Bar Code Transfer Logic Details

Function Blocks

BarcodeHandler Function Block

The BarcodeHandler function block handles all bar code transfer logic. This subroutine should be called by the main block every scan.

The subroutine performs the following logic:

- Enable bar code transfer and fragmentation mode with the cradle using the output Control Register Bar Code Transfer, Handshake Mode and Fragmentation Mode bits.
- Handle Update Counter resets (when Update Counter is 0, ACK Counter should be reset to 0)
- Monitor for the Update Counter to differ from the ACK Counter to indicate new a bar code data block.
- If a new bar code data block is received:
  - If first fragment block, or not fragmented, clear FragBuffer and reset FragOffset.
  - Store the Data field in the FragBuffer at the current FragOffset.
  - Update the FragOffset by Length.
  - Update the ACK Counter to match the Update Counter
  - If last fragment block, or not fragmented:

- Store the Update Counter in the BarcodeData.Counter tag.
- Store the FragOffset in the BarcodeData.Length tag.
- Store the FragBuffer in the BarcodeData.Data tag.
- Update the BarcodeData.Overflow tag based on the state of the Input Data Overflow bit in the input Status Register.

Tags

Global Tags

Table 4-19 *Global Tags*

Tag	Description
bcScannerIn	Mapped to the PROFINET input data from the cradle. This uses the *bcScannerInStruct* data type to define the data format.
bcScannerOut	Mapped to the PROFINET output data to the cradle. This uses the *bcScanerOutStruct* data type to define the data format.

BarcodeHandler Function Block Internal Tags

Table 4-20 includes the Static tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 4-20 *Internal Function Block Static Tags*

Tag	Description
FragOffset	Offset into the FragBuffer where the next data block from the cradle should be copied. This is also the current length of the bar code data in the FragBuffer.
FragBuffer	Buffer used to re-assemble the fragmented bar code data blocks. This is 4096 bytes in size.
FullCodeEdge	Flag used to track the status of the FullCodeRxed temporary tag.
NewBlockEdge	Flag used to track the status of the NewBlockRxed temporary tag.

Table 4-21 includes the Temporary tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 4-21 *Internal Function Block Temporary Tags*

Tag	Description
FullCodeRxed	Flag that indicates that a complete bar code has been received from the cradle. This may be triggered on the last fragment block of a fragmented transfer, or on a non-fragmented block. This flag is only active for a single scan when a full bar code has been assembled.
NewBlockRxed	Flag that indicates that a new bar code data block has been received from the cradle. This flag is only active for a single scan when new data is detected from the cradle.

CHAPTER 5 ETHERNET/IP INTERFACE

Introduction

This chapter provides information about I/O assemblies, I/O connections, steps to configure Rockwell ControlLogix communication, and transferring bar code data from the cradle.

Communications Profile

The EtherNet/IP interface on the STB3574 scanner cradle supports CIP Adapter functionality. The device is able to receive, or be the target of, I/O connections from a CIP Scanner, but is not able to originate connections itself.

The interface supports the Generic device profile. The Generic profile provides for all CIP objects that are required by the EtherNet/IP specification.

EDS File

The EtherNet/IP EDS file describes the Identity and I/O capabilities of the STB3574 cradle. The file is used by PLC configuration tools to configure the I/O connections and data tags used to communicate with the cradle over the EtherNet/IP network.

The latest EDS file can be acquired in the Industrial Ethernet Software Package. See *Chapter 1, Initial Setup and Industrial Ethernet Software*.

Supported Objects

The EtherNet/IP interface will support the following CIP objects:

- Identity
- Message Router
- TCP/IP Interface
- Ethernet Link
- Connection Manager
- Assembly.

TCP/IP Interface Object

The TCP/IP Interface object provides the ability to get and set TCP/IP configuration parameters. The IP address configuration may be set through the TCP/IP Interface object, but will require a device reset before any changes take effect.

✓ ***NOTE*** The EtherNet/IP interface does not support EtherNet/IP Address Conflict Detection.

Ethernet Link Object

The Ethernet Link object provides the ability to get link speed and duplex configuration parameters. The link configuration may not be changed through this object; all attributes are read-only.

I/O Assemblies

The EtherNet/IP interface includes two Assembly object instances that hold parameters and data used in the transfer of bar code data received from the scanner by the cradle to the controller.

Status and Barcode Data Assembly

Instance: 100

Access: Get

Size: 500 bytes

The Status and Barcode Data assembly holds the current status of the bar code transfer and the bar code data itself. The format of the assembly data follows that described in the *Status and Barcode Data Assembly* section above.

The Input Data field size 492 bytes.

Barcode Transfer Control Assembly

Instance: 150

Access: Get / Set

Size: 4 bytes

The Bar Code Transfer Control assembly is used for handshaking during bar code transfer by the controller. The format of the assembly data follows that described in the*Barcode Transfer Control Assembly* section above.

I/O Connections

The EtherNet/IP interface supports a single I/O connection that is used to transfer bar code data received from the scanner by the cradle to the controller. See *Transferring Bar Code Data from the Cradle on page 5-10* for the complete details of the bar code transfer operation.

Exclusive Owner Connection

Trigger and Transport:	Class 1, Cyclic
RPI Range:	10 - 1000 ms

O -> T

Connection Point:	150
Size:	4 bytes
Format:	Assembly instance 150

T -> O

Connection Point:	100
Size:	500 bytes
Format:	Assembly instance 100

Configuring Rockwell ControlLogix Communication

Configuring Using EDS Profile (ControlLogix v. 20 and later)

When using ControlLogix version 20 and later, the cradle communications can be completely configured using the EDS file in RSLogix.

Register the Cradle EDS File

Before the communications to the cradle can be configured, the EDS file must be registered with RSLogix. This only has to be done once.

To register the EDS file:

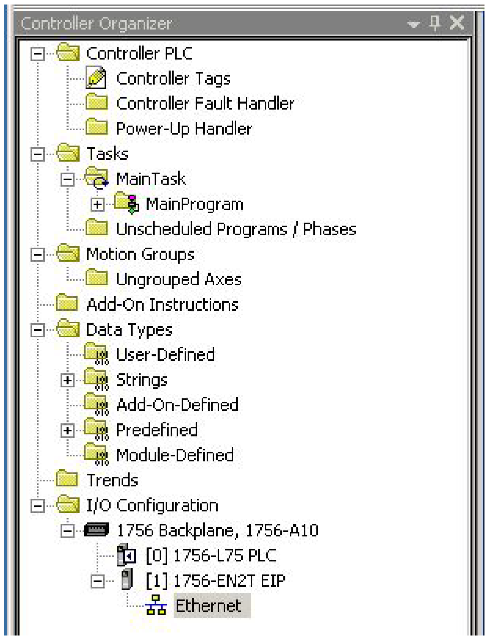
1. In the **RSLogix** menu, select Tools > EDS Hardware Installation Tool.
2. The EDS Wizard displays. Click **Next**.
3. Select the **Register an EDS File** radio button, and click **Next**.
4. Select the **Register a Single File** radio button.
5. Browse to the location of, and select the STB3574 EDS file.
6. Click **Next**.
7. The verification test results display. Click **Next**.
8. The **Icon** selection displays. Click **Next**.
9. The **Summary** displays. Click **Next**.
10. Click **Finish**.

Add the Cradle to the I/O Configuration

In order for the PLC to communicate with the cradle, it must be added to the I/O configuration in the program.

To add the I/O configuration:

1. Expand the I/O Configuration tree in the Organizer pane to display the Ethernet network.

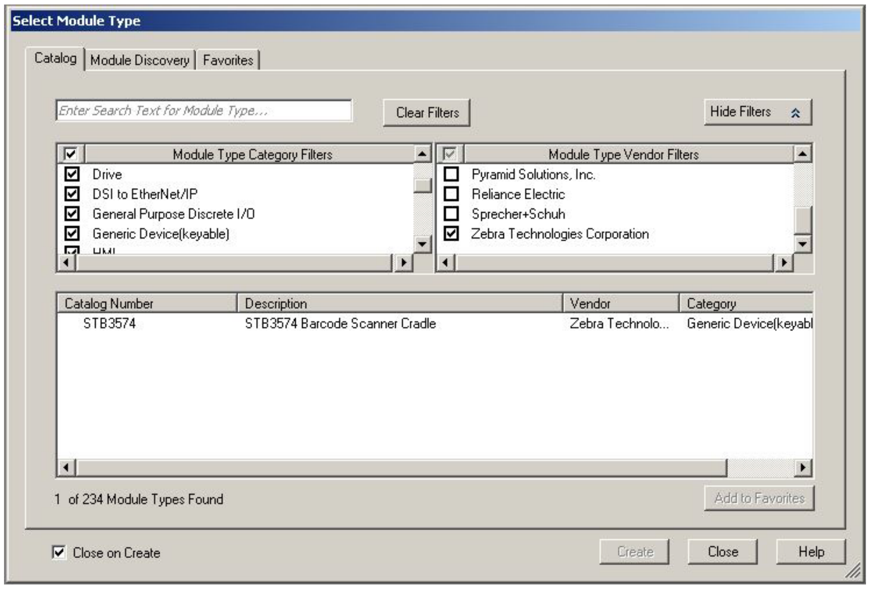


```
Controller Organizer
Controller PLC
    Controller Tags
    Controller Fault Handler
    Power-Up Handler
Tasks
    MainTask
        MainProgram
    Unscheduled Programs / Phases
Motion Groups
    Ungrouped Axes
Add-On Instructions
Data Types
    User-Defined
    Strings
    Add-On-Defined
    Predefined
    Module-Defined
Trends
I/O Configuration
    1756 Backplane, 1756-A10
        [0] 1756-L75 PLC
        [1] 1756-EN2T EIP
            Ethernet
```

Figure 5-1 *I/O Configuration Tree*

NOTE In *Figure 5-1*, a 1756-EN2T is used as the Ethernet interface. This may be another type of Ethernet interface module depending on the application.

2. Right-click on the **Ethernet** node in the tree and select **New Module**…
3. The **Select Module Type** dialog will be displayed. Change the Vendor filter to only select Zebra Technologies. The STB3574 cradle displays in the device list.



Select Module Type

Catalog | Module Discovery | Favorites

Enter Search Text for Module Type... Clear Filters Hide Filters

☑ Module Type Category Filters
☑ Drive
☑ DSI to EtherNet/IP
☑ General Purpose Discrete I/O
☑ Generic Device(keyable)
☑ HMI

☑ Module Type Vendor Filters
☐ Pyramid Solutions, Inc.
☐ Reliance Electric
☐ Sprecher+Schuh
☑ Zebra Technologies Corporation

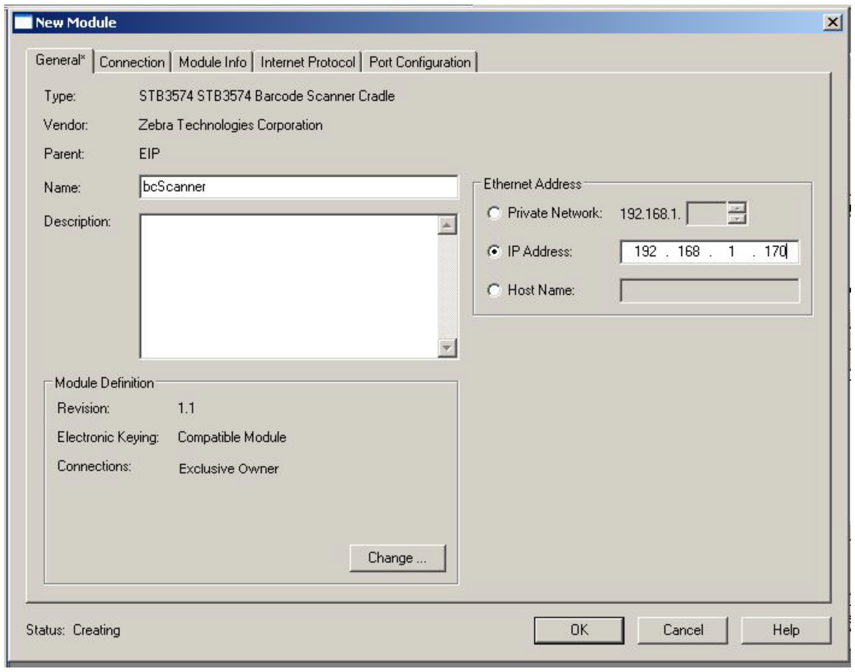
Catalog Number	Description	Vendor	Category
STB3574	STB3574 Barcode Scanner Cradle	Zebra Technolo...	Generic Device(keyabl

1 of 234 Module Types Found Add to Favorites

☑ Close on Create Create Close Help

Figure 5-2 *Select Module Type*

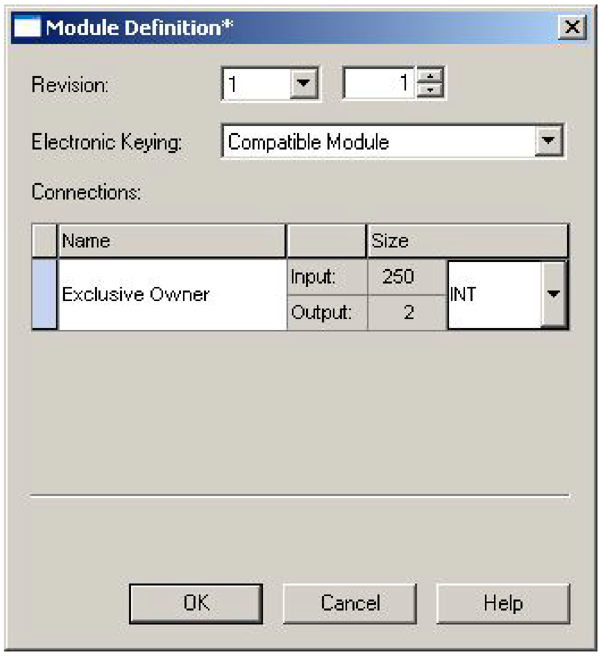
4. Select the STB3574 from the list and click **Create** button.
5. The **New Module** dialog will be displayed. Enter the desired name and IP address of the cradle.



Type: STB3574 STB3574 Barcode Scanner Cradle
Vendor: Zebra Technologies Corporation
Parent: EIP
Name: bcScanner
Description:
Module Definition
Revision: 1.1
Electronic Keying: Compatible Module
Connections: Exclusive Owner
Ethernet Address
Private Network: 192.168.1.
IP Address: 192 . 168 . 1 . 170
Host Name:
Status: Creating

Figure 5-3 *New Module, Setting Name and IP Address*

6. Click **Change…** to configure the I/O connection. The **Module Definition** dialog displays. Using the **Data Type** drop-down box, set the data type to INT. (Other data types may be used, but the 2 byte INT type is recommended for working with the 16-bit Status and Control registers in the I/O data.)



Revision: 1 1
Electronic Keying: Compatible Module
Connections:

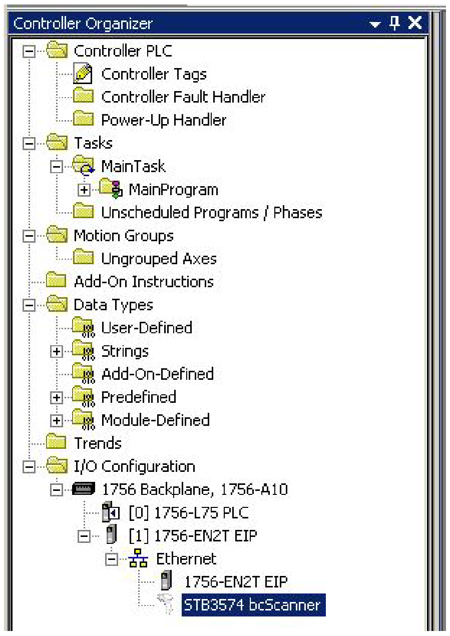
Name		Size	
Exclusive Owner	Input:	250	INT
	Output:	2	

Figure 5-4 *New Module, Selecting I/O Connection Format*

7. Press **OK**.
8. Press **Yes** in the dialog warning about changing the module definition.

9. Press **OK**.

10. The cradle is added to the I/O configuration, and appears in the tree.



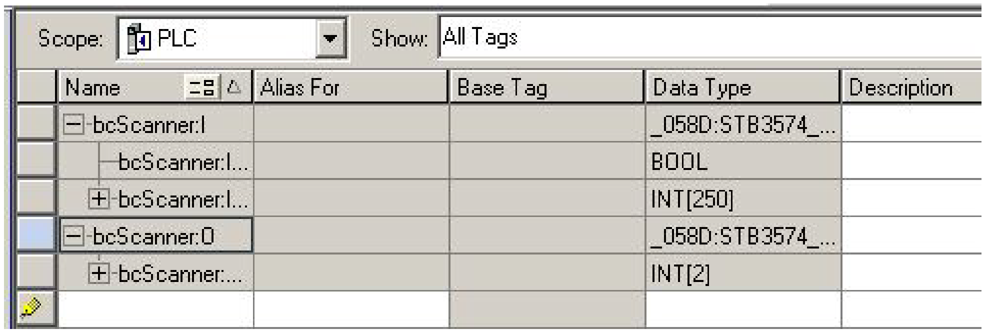
```
Controller Organizer
Controller PLC
    Controller Tags
    Controller Fault Handler
    Power-Up Handler
Tasks
    MainTask
        MainProgram
    Unscheduled Programs / Phases
Motion Groups
    Ungrouped Axes
Add-On Instructions
Data Types
    User-Defined
    Strings
    Add-On-Defined
    Predefined
    Module-Defined
Trends
I/O Configuration
    1756 Backplane, 1756-A10
        [0] 1756-L75 PLC
        [1] 1756-EN2T EIP
            Ethernet
                1756-EN2T EIP
                STB3574 bcScanner
```

Figure 5-5 *I/O Tree with Cradle Added*

All I/O connection parameters and I/O Tags are automatically configured when the module is added to the I/O Configuration.

Cradle I/O Tags

When the cradle is added to the I/O configuration, a set of tags is created to allow the PLC logic to read and write data to the cradle through the I/O connection. *Figure 5-6* shows the tags that are created.



Scope: PLC Show: All Tags

Name	Alias For	Base Tag	Data Type	Description
bcScanner:I			_058D:STB3574_...	
bcScanner:I...			BOOL	
bcScanner:I...			INT[250]	
bcScanner:O			_058D:STB3574_...	
bcScanner:...			INT[2]	

Figure 5-6 *Cradle Related Tags*

✓ ***NOTE*** The tag names are based on the name that was configured in the **New Module** dialog when the cradle was added to the I/O Configuration. In the example in *Figure 5-6*, the module was named "bcScanner".

Table 5-1 refers to *Figure 5-6*.

Table 5-1 *Cradle I/O Tags*

Name	Data Type Description
scannerName:I.ConnectionFaulted	A Boolean tag that indicates if the I/O connection with the cradle is faulted.
scannerName:I.Data	The Input data buffer holding data this is received from the cradle. The buffer is 250 words (500 bytes) long and is formatted as described in the Status and Barcode Data section.
scannerName:O.Data	The Output data buffer holding data that will be sent to the cradle. The buffer is 2 words (4 bytes) long and is formatted as described in the Barcode Transfer Control Assembly section.

Configuring Using Generic EtherNet/IP Module

The I/O communications with the cradle may also be configured using a Generic EtherNet/IP Module. This is the required means of configuration for RSLogix versions earlier than v20; however, it may also be used in versions 20 and later, if desired.

Adding the Cradle to the I/O Configuration

In order for the PLC to communicate with the cradle, it must be added to the I/O Configuration in the program.

To add the PLC to the I/O configuration program:

1. Expand the I/O configuration tree in the Organizer pane to display the Ethernet network interface.

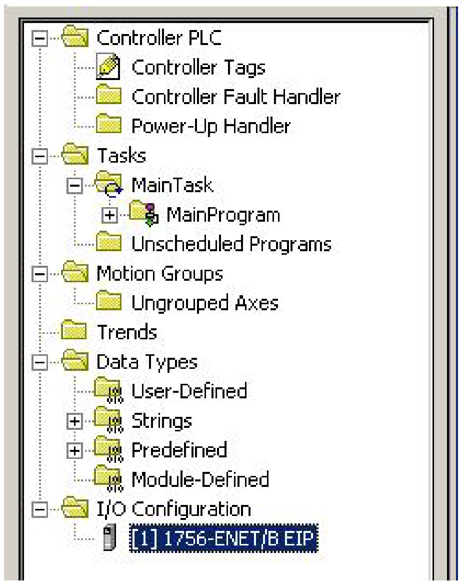
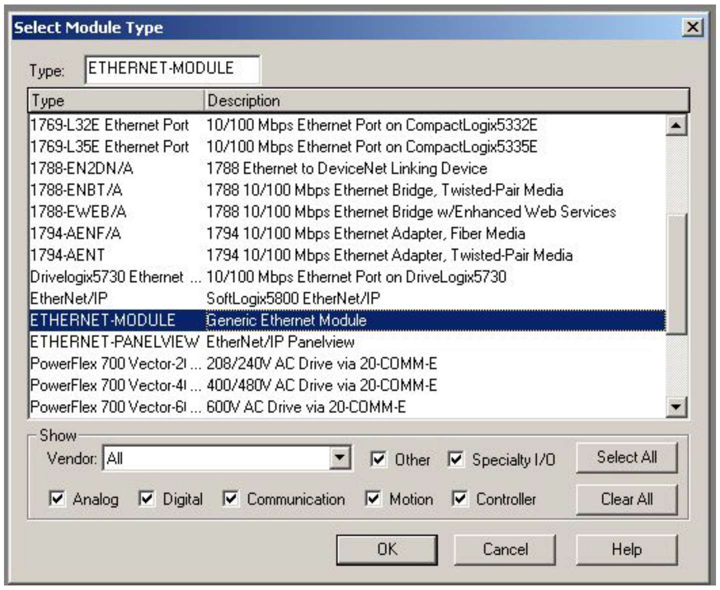


Figure 5-7 *I/O Tree Showing Network Interface*

✓ ***NOTE*** In the screen shot above, a 1756-ENET is used as the Ethernet interface. This may be another type of Ethernet interface module depending on the application.

2. Right click on the Ethernet interface node in the tree and select **New Module**…

3. The **Select Module Type** dialog displays. Select the ETHERNET_MODULE Generic Ethernet Module entry and click **OK**.



Select Module Type

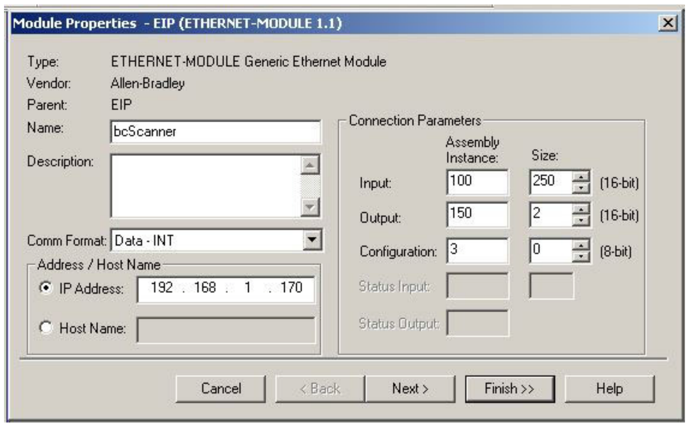
Type: ETHERNET-MODULE

Type	Description
1769-L32E Ethernet Port	10/100 Mbps Ethernet Port on CompactLogix5332E
1769-L35E Ethernet Port	10/100 Mbps Ethernet Port on CompactLogix5335E
1788-EN2DN/A	1788 Ethernet to DeviceNet Linking Device
1788-ENBT/A	1788 10/100 Mbps Ethernet Bridge, Twisted-Pair Media
1788-EWEB/A	1788 10/100 Mbps Ethernet Bridge w/Enhanced Web Services
1794-AENF/A	1794 10/100 Mbps Ethernet Adapter, Fiber Media
1794-AENT	1794 10/100 Mbps Ethernet Adapter, Twisted-Pair Media
Drivelogix5730 Ethernet ...	10/100 Mbps Ethernet Port on DriveLogix5730
EtherNet/IP	SoftLogix5800 EtherNet/IP
ETHERNET-MODULE	Generic Ethernet Module
ETHERNET-PANELVIEW	EtherNet/IP Panelview
PowerFlex 700 Vector-2I ...	208/240V AC Drive via 20-COMM-E
PowerFlex 700 Vector-4I ...	400/480V AC Drive via 20-COMM-E
PowerFlex 700 Vector-6I ...	600V AC Drive via 20-COMM-E

Show
Vendor: All
☑ Other ☑ Specialty I/O
☑ Analog ☑ Digital ☑ Communication ☑ Motion ☑ Controller

Figure 5-8 *Select Generic Module Type*

4. The **Module Properties** dialog displays.



Module Properties - EIP (ETHERNET-MODULE 1.1)

Type: ETHERNET-MODULE Generic Ethernet Module
Vendor: Allen-Bradley
Parent: EIP
Name: bcScanner
Description:
Comm Format: Data - INT
Address / Host Name
● IP Address: 192 . 168 . 1 . 170
○ Host Name:

Connection Parameters

	Assembly Instance:	Size:	
Input:	100	250	(16-bit)
Output:	150	2	(16-bit)
Configuration:	3	0	(8-bit)
Status Input:			
Status Output:			

Figure 5-9 *Module Properties for Generic Module Type*

a. Enter the desired name for the cradle.

b. Enter the IP address of the cradle.

c. Set the Comm Format to Data - INT. (Other data types may be used, but the 2 byte INT type is recommended for working with the 16-bit Status and Control registers in the I/O data.)

d. Set the Connection Parameters as:

i. Input: Instance 100, size 250

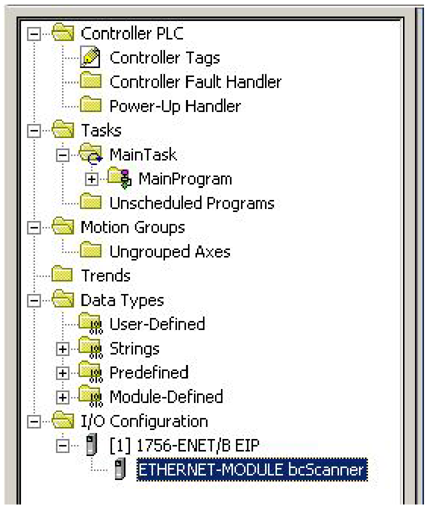
ii. Output: Instance 150, size 2

iii. Config: Instance 3, size 0

✓ ***NOTE*** These sizes are based on 2 byte INT data types, if another data type is used, the sizes must be altered to translate to 500 bytes Input and 4 bytes Output.

e. Click **Finish**.

5. The cradle is added to the I/O configuration and displays in the tree.

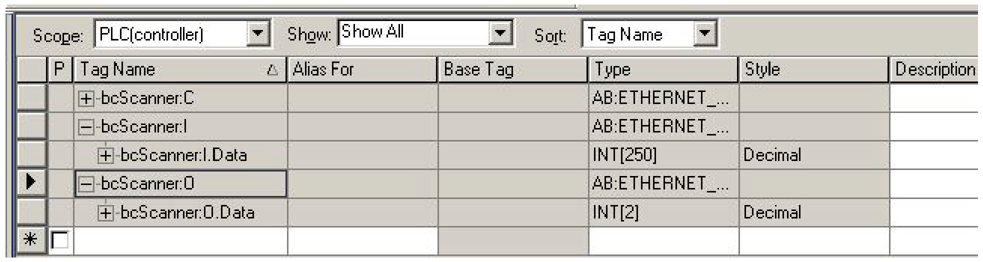


```
Controller PLC
    Controller Tags
    Controller Fault Handler
    Power-Up Handler
Tasks
    MainTask
        MainProgram
    Unscheduled Programs
Motion Groups
    Ungrouped Axes
Trends
Data Types
    User-Defined
    Strings
    Predefined
    Module-Defined
I/O Configuration
    [1] 1756-ENET/B EIP
        ETHERNET-MODULE bcScanner
```

Figure 5-10 *I/O Tree after Cradle Add*

Cradle Tags

When the cradle is added to the I/O Configuration, a set of tags is created to allow the PLC logic to read and write data to the cradle through the I/O connection. *Figure 5-11* displays the tags that are created.



Scope: PLC(controller) Show: Show All Sort: Tag Name

P	Tag Name	Alias For	Base Tag	Type	Style	Description
	bcScanner:C			AB:ETHERNET_...		
	bcScanner:I			AB:ETHERNET_...		
	bcScanner:I.Data			INT[250]	Decimal	
	bcScanner:O			AB:ETHERNET_...		
	bcScanner:O.Data			INT[2]	Decimal	

Figure 5-11 *Tags Created with Generic Module*

✓ ***NOTE*** The tag names are based on the name that was configured in the Module Properties dialog when the cradle was added to the I/O Configuration. In the screen shot example the module was named "bcScanner".

Table 5-2 refers to *Figure 5-10*.

Table 5-2 *Cradle Tags*

Tag Name	Type Description
scannerName:C	Unused configuration data. When a generic Ethernet module is added to the configuration, RSLogix automatically creates a configuration data buffer. In the case of the cradle, this buffer is not used.
scannerName:I.Data	The Input data buffer holding data this is received from the cradle. The buffer is 250 words (500 bytes) long and is formatted as described in the Status and Barcode Data section.
scannerName:O.Data	The Output data buffer holding data that will be sent to the cradle. The buffer is 2 words (4 bytes) long and is formatted as described in the Barcode Transfer Control Assembly section.

Transferring Bar Code Data from the Cradle

The EtherNet/IP interface supports all three transfer modes for transferring bar code data from the cradle to the controller.

See *Reading Bar Code Data from the Cradle on page 3-5* for details on the bar code transfer mechanisms.

Example Bar Code Transfer Logic

The example transfer logic consist of simple controller programs for a Rockwell ControlLogix PLC that include the logic required to transfer bar codes from the cradle to a controller tag.

Three examples are provided:

- *Basic Mode Transfer Example Logic* (below)
- *Handshake Mode Transfer Example Logic on page 5-13*
- *Fragmentation Mode Transfer Example Logic on page 5-15*.

Basic Mode Transfer Example Logic

Files

Table 5-3 includes the file provided for the Basic Mode transfer example. *Chapter 1, Initial Setup and Industrial Ethernet Software* for more information about where to find sample applications.

Table 5-3 *Basic Mode Transfer Example Files*

Basic Mode Transfer Folder/File	Description
EtherNetIP\Samples\basicBarcode.acd	Rockwell ControlLogix program containing the Basic Mode transfer logic

Using the Bar Code Transfer Logic

The bar code transfer logic included in the example program is encapsulated in a subroutine. This allows the bar code interface to be added easily to any program by simply adding a call to the subroutine. The subroutine places the bar code information into a Data Block when it is received from the cradle.

Block Interface Subroutine

The main subroutine used for the bar code transfer logic is the BarcodeInterface subroutine.   A JSR instruction to run this subroutine on each scan should be added to the main program.   This subroutine will handle all bar code transfer logic required to receive bar code information from the cradle.

Block Information Tags

Table 5-4 includes tags used by the BarcodeInterface subroutine.

Table 5-4 *tags Used by the BarcodeInterface Subroutine*

Tag	Description
EnableBarcodeTransfer	A flag used to enable and disable bar code transfer with the cradle. Note that the example program sets this tag on first scan to automatically enable bar code transfer.
BarcodeCounter	This tag is set to the current bar code update counter value received from the cradle. A change in this tag's value indicates that new bar code data has been received.
BarcodeLength	The length in bytes of the bar code data that was received.
BarcodeData	The actual bar code data received from the cradle. Only BarcodeLength bytes of the buffer will be used. The total buffer size is 492 bytes.
BarcodeDataOverflow	A flag indicating that the actual bar code data is longer than 492 bytes and was truncated by the EtherNet/IP interface in the cradle.

Bar Code Transfer Logic Details

Subroutines

BarcodeInterface Subroutine

The BarcodeInterface subroutine is the main subroutine that handles all bar code transfer logic. This subroutine should be called by the main program logic every scan.

The subroutine performs the following logic:

- Enable bar code transfer with the cradle using the output Control Register Barcode Transfer bit.
- Monitor the Update Counter from the cradle for changes.
- If the Update Counter changes, run the BarcodeStore subroutine.

BarcodeStore Subroutine

The BarcodeStore subroutine handles updating the bar code information tags with the data received from the cradle. This subroutine is called from within the BarcodeInterface subroutine when new bar code data is received from the scanner by the cradle.

The subroutine performs the following logic:

- Store the Update Counter in the BarcodeCounter tag.
- Store the Length in the BarcodeLength tag.
- Store the Input Data field in to the BarcodeData tag.
- Update the BarcodeDataOverflow tag based on the state of the Data Overflow bit in the Status Register.

Tags

Controller Tags

When the cradle is added to the ControlLogix program as an EtherNet/IP module, the following controller-based tags are automatically created.

Table 5-5 *Controller Tags*

Tag	Description
bcScanner:I	The input data from the cradle. This is 500 bytes and follows the format described in the section *Status and Barcode Data Assembly on page 5-2*.
bcScanner:O	The output data to the cradle. This is 4 bytes and follows the format described in the section *Status and Barcode Data Assembly on page 5-2*.

Program Tags

The program-based tags are divided into two groups: global to be used by the main program, and internal to be used by the bar code transfer subroutines. The global tags are described in *Table 5-5* above. *Table 5-6* describes the internal tags.

Table 5-6 *Internal Tags*

Tag	Description
bcInitOneshot	One-shot flag used to initialize the last update counter when transfer is enabled.
bcLastUpdateCounter	Holds the previous value of the input Update Counter.
bcNewCodeRxed	Flag that indicates that new bar code data has been received from the cradle. This flag is only active for a single scan when new data has been detected.
bcWaitForNewCode	State storage tag used in the OSR that controls the bcNewCodeRxed tag state.

Table 5-7 includes the Temporary tags that are used internally in the function block, defined by the BarcodeHandler block interface.

Table 5-7 *Temporary Tags*

Tag	Description
NewCodeRxed	Flag that indicates that new bar code data has been received from the cradle. This flag is only active for a single scan when new data has been detected.

Handshake Mode Transfer Example Logic

Files

Table 5-8 includes the file provided in the Handshake Mode transfer example.See *Chapter 1, Initial Setup and Industrial Ethernet Software* for more information about where to find sample applications

Table 5-8 *Handshake Mode Transfer Example Files*

Handshake Mode Transfer Folder/File	Description
EtherNetIP\Samples\barcodeHandshake.acd	Rockwell ControlLogix program containing the Handshake Mode transfer logic.

Using the Bar Code Transfer Logic

The bar code transfer logic included in the example program is encapsulated in a set of subroutines. This allows the bar code interface to be added easily to any program by simply adding a call to the subroutine. The subroutines will place the bar code information into a set of tags when it is received from the cradle.

Bar Code Interface Subroutine

The main subroutine used for the bar code transfer logic is the BarcodeInterface subroutine. A JSR instruction to run this subroutine on each scan should be added to the main program. This subroutine will handle all bar code transfer logic required to receive bar code information from the cradle.

Bar Code Information Tags

Table 5-9 describes the tags used by the BarcodeInterface subroutine.

Table 5-9 *Bar Code Information Tags*

Tag	Description
EnableBarcodeTransfer	A flag used to enable and disable bar code transfer with the cradle. Note that the example program sets this tag on first scan to automatically enable bar code transfer.
BarcodeCounter	This tag is set to the current bar code update counter value received from the cradle. A change in this tag's value indicates that new bar code data has been received.
BarcodeLength	The length in bytes of the bar code data that was received.
BarcodeData	The actual bar code data received from the cradle. Only BarcodeLength bytes of the buffer will be used. The total buffer size is 492 bytes.
BarcodeDataOverflow	A flag indicating that the actual bar code data is longer than 492 bytes and was truncated by the EtherNet/IP interface in the cradle.

Bar Code Transfer Logic Details

Subroutines

BarcodeInterface Subroutine

The BarcodeInterface subroutine is the main subroutine that handles all bar code transfer logic. This subroutine should be called by the main program logic every scan.

The subroutine performs the following logic:

- Enable bar code transfer and enable Handshake Mode using the output Control Register Barcode Transfer and Handshake Mode bits.
- Handles the error case of a 0 input Update Counter: The output ACK Counter should be reset to 0.
- Watch for the input Update Counter to differ from the output ACK Counter, which indicates new data.
- If new data has been received run the BarcodeStore subroutine.

BarcodeStore Subroutine

The BarcodeStore subroutine handles updating the bar code information tags with the data received from the cradle.   This subroutine is called from within the BarcodeInterface subroutine when new bar code data is received from the scanner by the cradle.

The subroutine performs the following logic:

- Store the Update Counter in the BarcodeCounter tag.
- Store the Length in the BarcodeLength tag.
- Store the Input Data field in the BarcodeData tag.
- Update the BarcodeDataOverflow tag based on the state of the Data Overflow bit in the Status Register.
- Update the output ACK Counter to match the input Update Counter.

Tags

Controller Tags

When the cradle is added to the ControlLogix program as an EtherNet/IP module, controller-based tags in *Table 5-10* are automatically created.

Table 5-10 *Controller Tags*

Tag	Description
bcScanner:I	The input data from the cradle. This is 500 bytes and follows the format described in the section *Status and Barcode Data Assembly on page 5-2*.
bcScanner:O	The output data to the cradle. This is 4 bytes and follows the format described in the section *Barcode Transfer Control Assembly on page 5-2*.

Program Tags

The program-based tags are divided into two groups: global to be used by the main program, and internal to be used by the bar code transfer subroutines. The global tags are described in *Bar Code Information Tags*, *Table 5-10 on page 5-14*. The internal tags are described in *Table 5-11* below.

Table 5-11 *Internal Bar Code Information Tags*

Tag	Description
NewCodeEdge	Flag used to track the status *NewCodeRxed* temporary tag.

Fragmentation Mode Transfer Example Logic

Files

Table 5-12 includes the file provided in the Fragmentation Mode transfer example. See *Chapter 1, Initial Setup and Industrial Ethernet Software* for more information about where to find sample applications.

Table 5-12 *Fragmentation Mode Transfer Example Files*

Fragmentation Mode Transfer Folder/File	Description
EtherNetIP\Samples\barcodeFragment.acd	Rockwell ControlLogix program containing the Fragmentation Mode transfer logic.

Using the Bar Code Transfer Logic

The bar code transfer logic included in the example program is encapsulated in a set of subroutines. This allows the bar code interface to be added easily to any program by simply adding a call to the subroutine. The subroutines will place the bar code information into a set of tags when it is received from the cradle.

Bar Code Interface Subroutine

The main subroutine used for the bar code transfer logic is the BarcodeInterface subroutine. A JSR instruction to run this subroutine on each scan should be added to the main program. This subroutine will handle all bar code transfer logic required to receive bar code information from the cradle.

Bar Code Information Tags

Table 5-13 includes the tags that are filled with bar code information by the BarcodeInterface subroutine.

Table 5-13 *Tags in the BarcodeInterface Subroutine*

Tag	Description
EnableBarcodeTransfer	A flag used to enable and disable bar code transfer with the cradle. Note that the example program sets this tag on first scan to automatically enable bar code transfer.
BarcodeCounter	This tag is set to the current bar code update counter value received from the cradle. A change in this tag's value indicates that new bar code data has been received.

Table 5-13 *Tags in the BarcodeInterface Subroutine (Continued)*

Tag	Description
BarcodeLength	The length in bytes of the bar code data that was received.
BarcodeData	The actual bar code data received from the cradle. Only BarcodeLength bytes of the buffer will be used. The total buffer size is 4096 bytes.

Bar Code Transfer Logic Details

Subroutines

BarcodeInterface Subroutine

The BarcodeInterface subroutine is the main subroutine that handles all bar code transfer logic. This subroutine should be called by the main program logic every scan.

The subroutine performs the following logic:

- Enable bar code transfer and enable Handshake and Fragmentation Modes using the output Control Register Barcode Transfer, Handshake Mode, and Fragmentation Mode bits.
- Handle the error case of 0 input Update Counter: the output ACK Counter should be reset to 0.
- Watch for the input Update Counter to differ from the output ACK Counter, which indicates a new block of data has been sent.
- If a new data block has been received
  - If it is not fragmented run the BarcodeStore subroutine.
  - If it is fragmented run the BarcodeFragHandler subroutine.
  - Update the output ACK Counter to match the input Update Counter.

BarcodeFragHandler Subroutine

The BarcodeFragHandler subroutine handles the reassembly of fragmented bar code data. This subroutine is called from within the BarcodeInterface subroutine when a new fragmented data block is received from the cradle.

The subroutine performs the following logic:

- If this is the First Fragment of the transfer, reset the fragment offset and clear the fragment buffer.
- Copy Length bytes from the input Data field into the fragment buffer at the current fragment offset.
- Update the fragment offset by the Length.
- If this is the last fragment of the transfer, run the BarcodeStore subroutine.

BarcodeStore Subroutine

The BarcodeStore subroutine handles updating the bar code information tags with the data received from the cradle. This subroutine is called from within the BarcodeInterface or the BarcodeFragHandler subroutines when complete bar code information is available to be stored in the tags.

The subroutine performs the following logic:

- Store the Update Counter from the input data in the BarcodeCounter tag.

- Set the BarcodeLength tag:
    - If the bar code is not fragmented, from the input Length field
    - If the bar code is fragmented, from the current fragment offset.
- Store the bar code data in the BarcodeData tag:
    - If the bar code is not fragmented, from the input Data field
    - If the bar code is fragmented, from the fragment buffer.

Tags

Controller Tags

When the cradle is added to the ControlLogix program as an EtherNet/IP module, the controller-based tags in *Table 5-14* are automatically created.

Table 5-14 *Controller Tags*

Tag	Description
bcScanner:I	The input data from the cradle. This is 500 bytes and follows the format described in the section *Status and Barcode Data Assembly on page 5-2*.
bcScanner:O	The output data to the cradle. This is 4 bytes and follows the format described in the section *Barcode Transfer Control Assembly on page 5-2*.

ProgramTags

The program-based tags are divided into two groups: global to be used by the main program, and internal to be used by the bar code transfer subroutines.   The global tags are described above in *Table 5-14* above. *Table 5-15* describes the internal tags.

Table 5-15 *Program Tags*

Tag	Description
bcNewBlockRxed	Flag that indicates that new bar code data block has been received from the cradle. This flag is only active for a single scan when new data has been detected.
bcWaitForNewBlock	State storage tag used in the OSR that controls the bcNewBlockRxed tag state.
bcFragBuffer	The fragmentation holding buffer. Fragmented bar code data is assembled in this buffer.
bcFragOffset	The offset into the fragmentation buffer where the next block is to be written.

CHAPTER 6 MODBUS TCP INTERFACE

Introduction

Communications Profile

This chapter provides information about using the Modbus TCP interface to transfer bar code data from the cradle.

The Modbus TCP interface on the STB3574 scanner cradle supports Modbus TCP server / slave functionality. The device is able to receive, or be the target of, I/O connections from a Modbus TCP client / master, but is not able to originate connections itself.

Modbus Unit Identifier

The Unit Identifier (or slave identifier) should be set to 1 for all Modbus TCP requests sent to the cradle.

Supported Modbus Functions

The following Modbus TCP function codes are supported by the cradle.

Table 6-1 *Program Tags*

Code	Function	Description
03	Read Holding Registers	Read from up to 125 Holding registers from a given starting address.
04	Read Input Registers	Read from up to 125 Input registers from a given starting address.
06	Write Single Register	Write a single Holding register.
16	Write Multiple Registers	Write into up to 125 Holding registers from a given starting address.

Modbus Register Mapping

The Modbus TCP interface provides access to registers that hold parameters and data used in the transfer of bar code data read by the cradle to the controller.

Cradle Identity and Version Information Registers

Register Type: Input

Register Range: 1 - 42

Access Functions: (04) Read Input Registers

The Identity and Version registers hold information about the cradle device and its software. The registers are mapped as shown in *Table 6-2*.

Table 6-2 *Identity and Version Registers*

Cradle Identity and Version Register Mapping				
Register Address	**Register Count**	**Parameter**	**Data Type**	**Description**
1	1	Firmware Revision	UINT16	High byte is the Major revision. Low byte is the Minor revision.
2	16	Serial Number	String	The serial number of the cradle.
18	25	Product Name	String	The cradle product / model name.

Status and Barcode Data Registers

Register Type: Input

Register Range: 200 - 2250

Access Functions: (04) Read Input Registers

The Status and Barcode Data registers holds the current status of the bar code transfer and the bar code data itself. The format of the register data follows the description above. The register mapping of each input data field is defined in *Table 6-3*.

Table 6-3 *Status and Barcode Data Registers*

Status and Barcode Data Register Mapping		
Register Address	**Register Count**	**Field**
200	1	Status Register
201	1	Update Counter
202	1	Length
203	2048	Data

✓ ***NOTE*** The Input Data field length is 4096 bytes.

Barcode Transfer Control Registers

Register Type: Holding

Register Range: 1 - 2

Access Functions: (03) Read Holding Registers

(06) Write Single Register

(16) Write Multiple Registers

The Barcode Transfer Control register is used for handshaking during bar code transfer by the controller. The format of the register data follows that described in *Bar Code Transfer Control Output Data on page 3-4*. The register mapping of each output data field is defined in *Table 6-4*.

Table 6-4 *Register Mapping of Each Output Data Field*

Barcode Transfer Control Register Mapping		
Register Address	**Register Count**	**Parameter**
1	1	Control Register
2	1	ACK Counter

Transferring Bar Code Data from the Cradle

The Modbus TCP interface supports Basic Mode and Handshake Mode transfers from the cradle to the controller. Fragmentation Mode is not required because the Modbus register data supports up to 4096 bytes of bar code data which is the largest bar code that the cradle interface supports.

See *Reading Bar Code Data from the Cradle on page 3-5* for details on the bar code transfer mechanisms.

Example Bar Code Transfer Register Commands

When using the Modbus TCP protocol, all interaction between the controller and cradle is done through register read and write commands. This section discusses the register command sequences required to perform typical bar code transfers.

Enable Transfer and Select Transfer Mode

Bar code transfer is enabled and the desired transfer mode is selected by writing to the Control Register.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Write	1	1	0001 for Basic Mode 0003 for Handshake Mode	n/a

Basic Mode Transfers

Detecting New Bar Code Data

New bar code data is detected by monitoring the Input Update Counter for a change.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Read	201	1	n/a	Current Update Counter value

If the Update Counter is non-zero and is not equal to the last Update Counter, new data is available.

Reading Bar Code Data

The bar code data and its length are retrieved from the cradle using the following range of registers.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Read	202	1	n/a	Bar code length
Read	203	2048	n/a	Bar code data

The controller is not required to read all 2048 registers of the bar code data. Only the first length/2 registers will contain valid data. All registers values after the actual bar code data will be set to 0.

Handshake Mode Transfers

Detecting New Bar Code Data

New bar code data is detected by monitoring the Input Update Counter for a change.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Read	201	1	n/a	Current Update Counter value

If the Update Counter is non-zero and is not equal to the last Update Counter, new data is available.

Reading and Acknowledging Bar Code Data

The bar code data and its length are retrieved from the cradle using the following range of registers.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Read	202	1	n/a	Bar code length
Read	203	2048	n/a	Bar code data

The controller is not required to read all 2048 registers of the bar code data. Only the first length/2 registers will contain valid data. All registers values after the actual bar code data will be set to 0.

Once the controller has read the bar code data, it must indicate to the cradle that it is safe to write the next bar code's data into the registers. This is done by setting the Output ACK Counter to match the Input Update Counter.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Write	2	1	Current input Update Counter value	n/a

Handshake Resynchronization

If the cradle detects an issue with the transfer (connection is broken, controller stopped, etc.) it will set the Input Update Counter to 0. The bar code transfer to the controller will stop until the controller has written a 0 into the Output ACK Counter.

Command	Start Register	Register Count	Data Sent to Cradle	Data Received from Cradle
Write	2	1	0000	n/a

Once both counters are 0, transfer of bar code data will resume.

CHAPTER 7 RESTORING SETTINGS TO FACTORY DEFAULTS

Introduction

The following bar codes can be scanned to restore settings to the factory defaults.

CAUTION The following bar codes do not reset any standard cradle and/or scanner parameters. Only Industrial Ethernet settings are affected. Refer to the *DS3578 With FIPS Digital Scanner Product Reference Guide* (p/n 72E-153466-xx) for information about setting and resetting standard parameters.

Options:

- Restore Only IP Address - Restores only the IP address setting to the factory defaults.
- Restore All Settings - Restores all settings, including the IP address, to factory defaults.

Restore Only IP Address

Restore All Settings

APPENDIX A TROUBLESHOOTING

Troubleshooting

Table A-1 *Troubleshooting the STB3574*

Symptom	Potential Issue	Solution
Find devices fails	Cradle is not connected to the network, or powered on.	Verify that the blue LED on the cradle is on (the cradle has power) and the Ethernet cable is plugged into the cradle, and into a network switch. Both Ethernet lights on the cradle should be on.
	Cradle is unable to get a valid IP address through DHCP.	Verify that the network the cradle is plugged into is also connected to a DHCP server that has available addresses in its address pool.
	Cradle is set to a Static IP address that is not accessible on the PC subnet.	Reset to factory defaults using the parameter bar codes in *Chapter 7, Restoring Settings to Factory Defaults*. Recycle power to the STB3574 and try again.
Sample applications not loading	The sample applications were created with the following software packages: TIA v13 SP1 or Logix Studio 5000 v24. Only these packages, or later versions are able to load them.	If your TIA or Logix development environment cannot load the sample application(s), then sample application(s) should be recreated using the information provided in this guide.
Cannot connect PLC to the STB3574	For PROFINET, the cradle needs to be assigned a device name that matches the one used in the application.	Assign a PROFINET device name through TIA (see *Setting the PROFINET Device Name on page 4-7*. Note: If you are using Step 7, ensure the PROFINET device name is all lower case.

Table A-1 *Troubleshooting the STB3574 (Continued)*

Symptom	Potential Issue	Solution
Red LED high-low error beeps occur when scanning bar codes	This occurs when the STB3574 is not connected to the Zebra Industrial Ethernet Utility and bar codes are scanned.	Connect to the STB3574 through the utility. Bar code data displays in the Status Log.
	PLC has not enabled transfers.	For bar code data to be sent to the PLC, the PLC must have the STB3574 Ctrl_EnableTransfer bit set.
Unable to load firmware	An error occurred during a firmware update.	Recycle power to the STB3574. Connect and try again.

INDEX

Zebra Technologies Corporation, Inc.
3 Overlook Point
Lincolnshire, IL 60069, U.S.A.
http://www.zebra.com

MN-002694-01 Revision A - MAY 2016