Barcode Reader ASCII Protocol – RS232 and TCP/IP

Barcode is able to receive any ASCII encoded message from any equipment. It operates as a slave over Serial or TCP/IP networks.

Summary Information

Communication Driver Name: Barcode

Implementation DLL: T.ProtocolDriver. Barcode.dll

Protocol: ASCII (American Standard Code for Information Interchange)

Interface: TCP/IP and Serial

Equipment Types Supported: Any equipment compatible with ASCII encoding

Protocol Options: BlockSize, Start Char, and End Char

Multi-threading: Only one instance for channel is allowed

Max Number of Nodes: Only one Node should be used for each channel

PC Hardware Requirements: Standard PC Ethernet interface board, RS485, or RS232 port

Channel Configuration

Protocol Options

BlockSize: Defines the maximum length of a single item. The default value is 256

StartChar: Determines the start character of incoming messages. When empty, starts receiving the message with any character

EndChar: Determines the end character of incoming messages. When empty, stops receiving the message when it reaches the block size length

Settings

Serial and Multi-Serial channels: Consult the Serial Port configuration from your equipment

TCP/IP channels:

ListeningPort: Defines the TCP/IP port that will be listening for the connections

Node Configuration

The PrimaryStation and the BackupStation should be blank

Point Configuration

The syntax for the Barcode communication points is: <Number of Bytes>

Where:

- **<Number of Bytes>** means how many bytes (characters) the Tag will receive

  The value 0 (zero) means that the Tag will receive the whole message

  Example: The message received was “ABCDXYZ”

Case1:

TagName	Address	Value
TagA	4	ABCD
TagB	3	XYZ

Case2:

TagName	Address	Value
TagC	0	ABCXYZ

Troubleshoot

The status of the driver's execution can be observed through the diagnostic tools, which are:

- Trace window
- Property Watch
- Module Information

The above tools indicate if the operations have succeeded or have failed. A status of 0 (zero) means communication is successful. Negative values indicate internal driver errors, and positive values indicate protocol error codes.