INDUSTRIAL CONTROL COMMUNICATIONS, INC.

Generic Socket Server Driver Manual

October 30, 2014

TABLE OF CONTENTS

1 Generic Socket Server

1.1 Overview

The Generic Socket server driver can be used to receive ASCII and raw data from a variety of TCP and UDP clients, such as barcode scanners, weigh scales and serial-to-Ethernet converters. The driver uses a connection object element to target the client device. The connection object defines a connection to a specific client endpoint (IP address and port number) and defines the behavior of the connection.

Some notes of interest are:

- Both TCP and UDP connections are supported. TCP is connection-oriented and guarantees the delivery and order of packets. UDP is faster than TCP, but does not guarantee the delivery or order of packets.
- The driver supports up to 10 total connection objects (TCP and/or UDP).

1.2 Connection Object Behavior

For TCP connection objects, upon boot-up the driver will wait for a persistent TCP socket connection to be established by the client. The act of establishing this connection serves as notification to the client that it may begin transmitting data over this connection. For UDP connection objects (which are by nature connectionless), upon boot-up the driver will begin listening for client data on the designated *Source Port*.

In some cases, the client may transmit data in an unsolicited fashion at a device-specific rate, where no acknowledgement sequence is required from the server. In this case, the server driver will never transmit anything to the client. In other cases, the client may require that the server transmit a specific acknowledgement sequence upon data reception in order to evoke subsequent data transmissions (similar to a typical “request/response” paradigm). Considerations for both of these scenarios are included in the driver.

1.3 Connection Object Settings

With the single exception of the “Destination Port” field available in UDP Connection Objects, both UDP and TCP Connection Object configuration is identical. To simplify the documentation, this section will therefore jointly summarize the settings for both types.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

IP Address

Defines the IP address of the client device that is permitted to target the connection object.

Destination Port

This field is available only for UDP Connection Objects. Defines the destination port (1…65535) on the client device to which optional acknowledgment sequences will be sent. This field is available only if a *Transmit Ack Sequence* is in use (if a *Transmit Ack Sequence* is not used, then the driver never sends anything to the client device.)

Source Port

Defines the local port (1…65535) which the driver will monitor for incoming data from the client device. Assigned source ports must be unique within TCP and UDP connection object groups.

Transmit Ack Sequence

Check this checkbox to send an optional acknowledgement sequence to the client after every completed processing action (regardless of whether or not a packet was successfully consumed.) Among other possible uses, an acknowledgement sequence may be necessary to evoke subsequent data transmissions from the client. Check the client device's documentation to determine whether or not an acknowledgement sequence is required.

Ack Sequence

This field is available only when the Transmit Ack Sequence checkbox is enabled. Enter the *Ack Sequence* required by the client as a string of hexadecimal values, two characters at a time. For example, entering "7E" in this field will result in the single hexadecimal byte 0x7E being transmitted in the acknowledgement sequence. If the client requires an ASCII-based sequence, then enter the appropriate ASCII codes (e.g. entering "4D36" results in the ASCII characters "M6" being sent by the driver). Up to 100 hexadecimal value codes can be entered in this field.

Receive Termination Mode

Select the appropriate receive termination mode. Once the termination mode is satisfied, the received data is considered complete and will be processed by the driver. The available selections are:

Termination Sequence..... The client sends a specific termination sequence.

Message Timeout ..... After initial detection of incoming data, the driver will wait a specified amount of time (the *Receive Delay* setting). All further incoming data will be processed until the receive delay time has elapsed.

Character/Packet Gap Delay ..... After initial detection of incoming data, all further incoming data will be processed until a reception gap occurs that meets or exceeds the *Receive Delay* setting.

Fixed Size..... The driver processes all data until a specified amount of data is received.

Receive Termination Sequence

This field is available only when Receive Termination Mode is set to "Termination Sequence". Enter the termination sequence generated by the client as a string of hexadecimal values, two characters at a time. For example, enter "7E" in this field if the client transmits the single hexadecimal byte 0x7E as its termination sequence. If the client generates an ASCII-based sequence, then enter the appropriate ASCII codes (e.g. enter "4D36" if the client transmits the

two ASCII characters “M6” as its termination sequence). One or two hexadecimal value codes can be entered in this field.

Strip Receive Termination Sequence

This field is available only when Receive Termination Mode is set to “Termination Sequence”. Check this checkbox to strip the termination sequence prior to storing data in the database. If unchecked, the termination sequence will be stored in the database along with the packet data.

Receive Delay

This field is available only when Receive Termination Mode is set to “Message Timeout” or “Character/Packet Gap Delay”.

- In “Message Timeout” mode, this is the amount of time during which all received data will be processed after initial detection of incoming data. This timer is started once when received data is initially detected, and expiration of this timer terminates reception.
- In “Character/Packet Gap Delay” mode, this is the length of the reception gap time. This timer is first started when received data is initially detected and restarted after each subsequent reception. Expiration of this timer terminates reception.

Receive Timeout

Defines the receive data timeout time. While its specific application varies somewhat depending on the selected *Receive Termination Mode*, expiration of this timer generally indicates an unsuccessful timeout condition. Refer to section 1.4 for a detailed summary of the Receive Timeout behavior.

Number of Bytes

Specifies the size of the database buffer, starting at the designated *Database Address*. The entire database buffer of size “Number of Bytes” is modified upon every successful reception (either with actual received data or zero-filled data appended to received data). Refer to section 1.4 for further details regarding the application of the Number of Bytes field for each *Receive Termination Mode*.

Database Address

Defines the start of the database buffer, the size of which is specified by the *Number of Bytes* field.

1.4 Packet Transfer and Termination Summary

This section serves as an overview of the reception flow processing action and evaluation criteria used to determine the successful or unsuccessful reception of packetized data for each of the Receive Termination Modes. For TCP connection objects, these processing actions assume that a TCP socket connection has already been established (as no processing takes place outside the context of a TCP connection). For UDP connection objects, these processing actions occur continuously, without any prior requirements.

1.4.1 Termination Sequence

- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, but a packet gap exceeding the *Receive Timeout* time occurs prior to detection of the *Receive Termination Sequence*, then the processing action terminates with a failed/timeout status.
- If received data is detected, but the cumulative number of bytes received exceeds 1460 bytes prior to detection of the *Receive Termination Sequence*, then the processing action terminates with a failed/too much data status.
- If received data is detected, and the *Receive Termination Sequence* is detected without a *Receive Timeout* gap, then a packet is consumed from the receive buffer and the processing action terminates successfully.
    - If the cumulative number of bytes in the packet exceeds the designated *Number of Bytes*, then only the first "*Number of Bytes*" is retained and the remainder is discarded.
    - If the cumulative number of bytes in the packet is less than the designated *Number of Bytes*, then the remainder of the database buffer is zero-filled.
- Note that depending on relative timing and the transmit characteristics of the client device, it may be possible to receive data that includes more than one *Receive Termination Sequence* (multiple packets may exist in the receive buffer). In such a scenario, only the "last" full packet is consumed, and the others are discarded. Any data received after the "last" full packet is assumed to be a partial packet for the next *Receive Termination Sequence*, and is therefore retained for the next processing action.
- If a *Transmit Ack Sequence* is in use, it is sent to the client device.

Example

Receive Termination Sequence = 0D (ASCII "carriage return")
Strip Receive Termination Sequence enabled

Received data (`N`, `P`, `W`… are any 8-bit values other than 0x0D):

`N P W 0x0D Y Z`

- If *Number of Bytes* = 2, then `NP` is stored in the database buffer
- If *Number of Bytes* = 3, then `NPW` is stored in the database buffer
- If *Number of Bytes* = 5, then `NPW00` is stored in the database buffer
- `YZ` remains in the receive buffer for a subsequent processing action, and will not be packetized until an additional carriage return is received.

1.4.2 Message Timeout

- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, then the *Receive Timeout* timer is stopped and the *Receive Delay* timer is started.
    - If the cumulative number of bytes received exceeds 1460 bytes prior to the *Receive Delay* timer expiration, then the processing action terminates with a failed/too much data status.

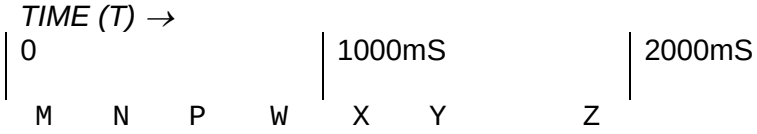
- All subsequent data is received until the *Receive Delay* timer expires. A packet is then consumed from the receive buffer and the processing action terminates successfully.
- If the cumulative number of bytes in the packet exceeds the designated *Number of Bytes*, then only the first "*Number of Bytes*" is retained and the remainder is discarded.
- If the cumulative number of bytes in the packet is less than the designated *Number of Bytes*, then the remainder of the database buffer is zero-filled.

- If a *Transmit Ack Sequence* is in use, it is sent to the client device.

Example

Receive Delay = 1000mS
Scan Rate = 0mS

Received data (`M`, `N`, `P`… are any 8-bit values):



- If *Number of Bytes* = 2, then `MN` is stored in the database buffer at time T=1S, and `XY` is stored in the database buffer at time T=2S
- If *Number of Bytes* = 4, then `MNPW` is stored in the database buffer at time T=1S, and `XYZ0` is stored in the database buffer at time T=2S

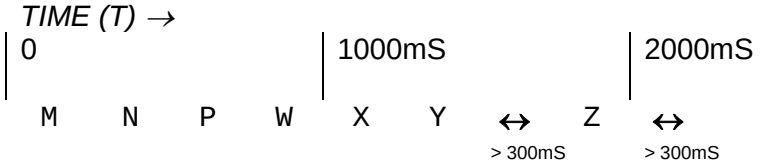
1.4.3 Character/Packet Gap Delay

- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, then the *Receive Timeout* timer is stopped and the *Receive Delay* timer is started.
  - After each data character/packet reception, the *Receive Delay* timer is restarted.
  - If the cumulative number of bytes received exceeds 1460 bytes prior to *Receive Delay* timer expiration, then the processing action terminates with a failed/too much data status.
  - When the *Receive Delay* timer expires (i.e. a packet gap is detected), then a packet is consumed from the receive buffer and the processing action terminates successfully.
  - If the cumulative number of bytes in the packet exceeds the designated *Number of Bytes*, then only the first "*Number of Bytes*" is retained and the remainder is discarded.
  - If the cumulative number of bytes in the packet is less than the designated *Number of Bytes*, then the remainder of the database buffer is zero-filled.
- If a *Transmit Ack Sequence* is in use, it is sent to the client device.

Example

Receive Delay = 300mS
Scan Rate = 0mS

Received data (`M`, `N`, `P`… are any 8-bit values):



- If *Number of Bytes* = 4, then `MNPW` is stored in the database buffer after the first gap detection, and `Z000` is stored in the database buffer after the second gap detection.

1.4.4 Fixed Size

- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, but a packet gap exceeding the *Receive Timeout* time occurs prior to receiving the designated *Number of Bytes*, then the processing action terminates with a failed/timeout status.
- If received data is detected, and the designated *Number of Bytes* is received without a *Receive Timeout* gap, then a packet is consumed from the receive buffer and the processing action terminates successfully.
- If a *Transmit Ack Sequence* is in use, it is sent to the client device.

Example

Number of Bytes = 4

Received data (`A`, `B`, `C`… are any 8-bit values):

`A B C D E F G H I J`

- Packet `ABCD` is stored in the database buffer after the first processing action.
- Packet `EFGH` is stored in the database buffer after the second processing action.
- `I` and `J` remain in the receive buffer for a subsequent processing action, and will not be packetized until two additional bytes are received.

ICC

INDUSTRIAL CONTROL COMMUNICATIONS, INC.

1600 Aspen Commons, Suite 210
Middleton, WI USA 53562-4720
Tel: [608] 831-1255 Fax: [608] 831-2045

http://www.iccdesigns.com

Printed in U.S.A