A Sierra Monitor Company

Driver Manual

(Supplement to the FieldServer Instruction Manual)

FS-8704-03 ModbusTCP

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after January 2013

Driver Version:	1.05
Document Revision:	4

TABLE OF CONTENTS

1 MODBUS/TCP DESCRIPTION

The Modbus TCP Driver allow the FieldServer to transfer data to and from devices over Ethernet using Modbus TCP Protocol. The Modbus TCP driver uses port 502 and this port is not configurable. The driver was developed for Modbus Application Protocol Specification V1.1a" from Modbus-IDA. The specification can be found at www.modbus.org. The FieldServer can emulate either a Server or Client.

The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

There are various register mapping models being followed by various vendors. To cover all these models FieldServer uses the following three Models

- **Modicon_5digit** – Use this format where addresses are defined in 0xxxx, 1xxxx, 3xxxx or 4xxxx format. A maximum of 9999 registers can be mapped of each type. This is FieldServer driver's default format.
- **ADU** –Application Data Unit address. Use this format where addresses of each type are defined in the range 1-65536
- **PDU** –Protocol Data unit address. Use this format where addresses of each type are defined in the range 0-65535.

The key difference between ADU and PDU is for example if Address_Type is ADU and address is 1, the driver will poll for register 0. If Address_Type is PDU, the driver will poll for address 1.

Note 1: If vendor document shows addresses in extended Modicon (i.e. 6 digit) format like 4xxxxx then consider these addresses as xxxxx (omit the first digit) and use either ADU or PDU

Note 2: The purpose of providing 3 different ways of addressing the Modbus registers is to allow the user to choose the addressing system most compatible with the address list being used. At the protocol level, the same protocol specification is used for all three with the exception of the limited address range for Modicon_5digit.

2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by FieldServer Technologies for this Driver

FieldServer Technologies PART #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection[1]

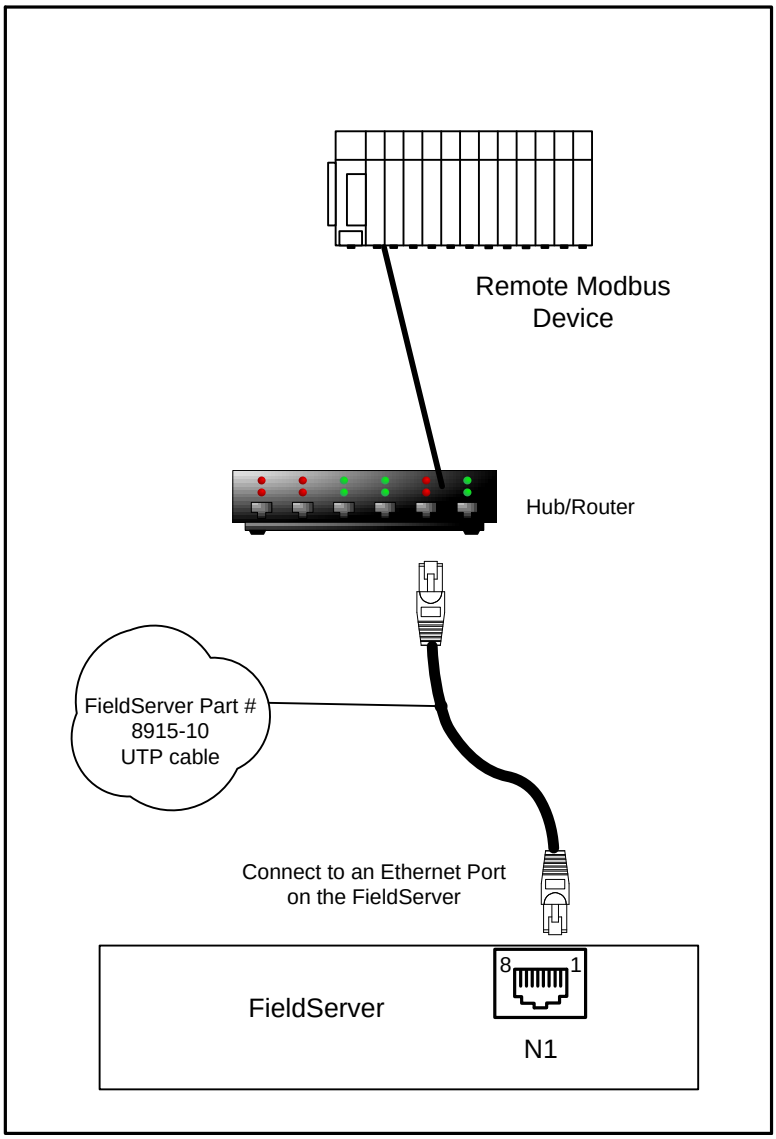
2.2 Provided by Supplier of 3rd Party Equipment

Part #	Description
	Modbus/TCP Server, e.g. Quantum PLC [2]
	Modbus/TCP Host Node, e.g. Intellution Fix, Wondereware Intouch, GE Cimplicity, Quantum PLC (Master).[3]

[1] This cable is necessary for connection to the driver. It is shipped with the FieldServer and not separately with the driver.
[2] If FieldServer used as Modbus/TCP Client.
[3] If FieldServer used as Modbus/TCP Server

3 HARDWARE CONNECTIONS

Configure the PLC according to manufacturer's instructions.



4 DATA ARRAY PARAMETERS

Data Arrays are "protocol neutral" data buffers for storage of data to be passed between protocols. It is necessary to declare the data format of each of the Data Arrays to facilitate correct storage of the relevant data.

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	UINT 16, UINT 32, SINT 16, SINT 32, BIT, FLOAT
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1 – 255

Example

```
//  Data Arrays
Data_Arrays
Data_Array_Name   , Data_Array_Format   , Data_Array_Length
DA_AI_01          , Float               , 200
DA_AO_01          , Float               , 200
DA_DI_01          , Bit                 , 200
DA_DO_01          , Bit                 , 200
```

5 CONFIGURING THE FIELDSERVER AS A MODBUS/TCP CLIENT

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" sample files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a Modbus TCP Server.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Modbus TCP communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the Servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that * indicates an optional parameter, with the bold legal value being the default.

5.1 Client Side Connection Parameters

Section Title		
Connections		
Column Title	Function	Legal Values
Adapter	Specify which adapter this protocol uses	N1
Protocol	Specify protocol used	Modbus/TCP
Poll Delay*	Time interval between polls	0-32000 s, **0.05 s.**

Example:

```
// Client Side Connections

Connections
Adapter      , Protocol      , Poll_Delay
N1           , Modbus/TCP    , 0.05s
```

5.2 Client Side Node Parameters

Section Title		
Nodes		
Column Title	Function	Legal Values
Node_Name	Provide name for Node	Up to 32 alphanumeric characters
Node_ID	Station Address of Remote Server Node	1 – 255
Protocol	Specify protocol used	Modbus/TCP
Adapter	Specify which adapter this protocol uses	N1
IP_address	IP address of client PLC	Valid IP address, e.g. 192.168.1.13
Address_Type[4]	Specify Register Mapping Model	ADU,PDU, **Modicon_5digit**
Modbus_TCP_IP_Port*	Select remote Internet Protocol Port	1-65534, **502**
Write_Fnc*	Set to Multiple if Remote Server Node only supports Write Multiple function code 15 & 16	Multiple, **-**,
Write_Length*	Set to Map Descriptor Length if write-thru operation should write all registers as specified by Map Descriptor length. By default write-thru writes a single register. If Write_Length also specified on Map Descriptor, Map Descriptor's parameter will be used.	Map Descriptor Length, **-**, **1**,

Example

For new devices where 65536 registers could be available in each memory area

```
//  Client Side Nodes
Nodes
Node_Name        , Node_ID   , Protocol      , Adapter   , Address_Type   , IP_Address
Modbus device 1  , 1         , Modbus/TCP   , N1        , ADU            , 192.168.1.172
Modbus device 2  , 2         , Modbus/TCP   , N1        , PDU            , 192.168.1.172
```

For devices where only 9999 registers could be available in each memory area

```
//  Client Side Nodes
Nodes
Node_Name        , Node_ID   , Protocol      , Adapter   , IP_Address
Modbus device 3  , 3         , Modbus/TCP   , N1        , 192.168.1.172
```

4 Optional for Modicon 5 digit devices

5.3 Client Side Map Descriptor Parameters

5.3.1 FieldServer Related Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Dame of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section 4
Data_Array_Offset	Starting location in Data Array	0 to (Data_Array_Length -1) as specified in Section 4
Function	Function of Client Map Descriptor	Rdbc, Wrbc, Wrbx, Passive

5.3.2 Driver Related Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the Node names specified in Section 0
Data_Type[5]	Specify memory area. Refer to Appendix A.1.2 on how to transfer 32 Bit values using Modbus registers.	Address_Type = ADU
		Coil, Discrete_Input, Input_Register, Holding_Register, Single_Coil, Single_Register
		Address_Type = PDU
		FC01, FC02, FC03, FC04, FC05, FC06, FC15, FC16
		Address_Type = Modicon_5digit
		- (Dash), Single_Register, Single_Coil
Address	Starting address of read block	Address_Type = ADU
		1-65536
		Address_Type = PDU
		0-65535
		Address_Type = Modicon_5digit
		40001, 30001, etc
		All Address_Type
		Float_Reg, 32-Bit_Reg, Input_Float, Input_Reg_32Bit
Length	Length in items to record from PLC	1 – 125 (depending on type)
Write_Length*	Set to Map Descriptor Length if write-thru operation should write all registers as specified by Map Descriptor length. By default write-thru writes a single register. If Write_Length also specified on Map Descriptor, Map Descriptor's parameter will be used.	Map Descriptor Length, **-**, **1**,

[5] Optional only for Modicon_5digit addressing, and only if Single writes do not need to be forced

Data_Array_Low_Scale*	Scaling zero in Data Array	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647. **0**
Data_Array_High_Scale*	Scaling max in Data Array	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647. **100**
Node_Low_Scale*	Scaling zero in Connected Node	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647. **0**
Node_High_Scale*	Scaling max in Connected Node	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647. **100**

5.3.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval*	Seconds per Scan	0-32000, **20**

5.3.4 Map Descriptor Example.

```
//  Client Side Map Descriptors
// Note: All three examples below are addressing the same Modbus registers.

// For Nodes where Address_Type is ADU
Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name         , Data_Type          , Address   , Length   , Scan_Interval
CMD_AI_01             , DA_AI_01          , 0                   , Rdbc       , MODBUS DEVICE1    , Input_Register     , 1         , 20       , 1.000s
CMD_AO_01             , DA_AO_01          , 0                   , Rdbc       , MODBUS DEVICE1    , Holding_Register   , 1         , 20       , 1.000s
CMD_DI_01             , DA_DI_01          , 0                   , Rdbc       , MODBUS DEVICE1    , Discrete_Input     , 1         , 20       , 1.000s
CMD_DO_01             , DA_DO_01          , 0                   , Rdbc       , MODBUS DEVICE1    , Coil               , 1         , 20       , 1.000s

// For Nodes where Address_Type is PDU
Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name         , Data_Type          , Address   , Length   , Scan_Interval
CMD_AI_02             , DA_AI_02          , 0                   , Rdbc       , MODBUS DEVICE2    , FC04               , 0         , 20       , 1.000s
CMD_AO_02             , DA_AO_02          , 0                   , Rdbc       , MODBUS DEVICE2    , FC03               , 0         , 20       , 1.000s
CMD_DI_02             , DA_DI_02          , 0                   , Rdbc       , MODBUS DEVICE2    , FC02               , 0         , 20       , 1.000s
CMD_DO_02             , DA_DO_02          , 0                   , Rdbc       , MODBUS DEVICE2    , FC01               , 0         , 20       , 1.000s

// For Nodes where Address_Type is Modicon_5digit.
Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name                              , Address   , Length   , Scan_Interval
CMD_AI_03             , DA_AI_03          , 0                   , Rdbc       , MODBUS DEVICE3                        , 30001     , 20       , 1.000s
CMD_AO_03             , DA_AO_03          , 0                   , Rdbc       , MODBUS DEVICE3                        , 40001     , 20       , 1.000s
CMD_DI_03             , DA_DI_03          , 0                   , Rdbc       , MODBUS DEVICE3                        , 10001     , 20       , 1.000s
CMD_DO_03             , DA_DO_03          , 0                   , Rdbc       , MODBUS DEVICE3                        , 00001     , 20       , 1.000s
```

6 CONFIGURING THE FIELDSERVER AS A MODBUS/TCP SERVER

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" sample files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a Modbus TCP Client.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Modbus TCP communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the FieldServer virtual Node(s) needs to be declared in the "Server Side Nodes" section, and the data to be provided to the clients needs to be mapped in the "Server Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the **bold** legal value being the default.

6.1 Server Side Connection Parameters

Section Title		
Connections		
Column Title	Function	Legal Values
Adapter	Specify which adapter this protocol uses.	N1
IP_Port	Specify internet protocol Port	1-65534
Protocol	Specify protocol used	Modbus/TCP

Example:

```
// Server Side Connections

Connections
Adapter                    , Protocol
N1                         , Modbus/TCP
```

6.2 Server Side Node Parameters

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Node ID of physical server node	1 – 255 (Optional)
Protocol	Specify protocol used	Modbus/TCP
Address_Type*[6]	Specify Register Mapping Model	ADU,PDU, **Modicon_5digit**
Node_Offline_Response*	Set the FieldServer response to the Modbus TCP Client when the Server Node supplying the data has gone offline	**No_Response**, Old_Data, Zero_Data, FFFF_Data, Refer to Appendix A.1

Note that for this protocol, the IP address for the FieldServer is configured using the "I" menu option on the Remote User Interface.

Example

```
//  Server Side Nodes
// For devices where 65536 addresses are available in each memory area.
Nodes
Node_Name          , Node_ID      , Protocol      , Address_Type
MB_Srv_11          , 11           , Modbus/TCP    , ADU
MB_Srv_12          , 12           , Modbus/TCP    , PDU
// For devices where only 9999 registers are available in each memory area.
Nodes
MB_Srv_13          , 13           , Modbus/TCP    , Modicon_5digit
MB_Srv_14          , 14           , Modbus/TCP    , -
```

6.3 Server Side Map Descriptor Parameters

6.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Dame of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section 4.
Data_Array_Offset	Starting location in Data Array	0 to (Data_Array_Length 1-1) as specified in Section 4.
Function	Function of Client Map Descriptor	Passive

[6] Optional for Modicon 5 digit devices

6.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node this Map description is associated with	One of the node names specified in "Node_Name" above
Data_Type[7]	Specify memory area	Address_Type = ADU
		Coil, Discrete_Input, Input_Register, Holding_Register, Single_Coil, Single_Register
		Address_Type = PDU
		FC01, FC02, FC03, FC04, FC05, FC06, FC15, FC16
		Address_Type = Modicon_5digit
		- (Dash), Single_Register, Single_Coil
Length	Length of Map Descriptor	Address_Type = ADU
		1-65535
		Address_Type = PDU
		1-65535
		Address_Type = Modicon_5digit
		1-9999
Address	Starting address of read block	Address_Type = ADU
		1-65536
		Address_Type = PDU
		0-65535
		Address_Type = Modicon_5digit
		40001, 30001, etc
Data_Array_Low_Scale*	Scaling zero in Data Array	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647, **0**
Data_Array_High_Scale*	Scaling max in Data Array	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647, **100**
Node_Low_Scale*	Scaling zero in Connected Node	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647, **0**
Node_High_Scale*	Scaling max in Connected Node	Any signed 32 bit integer in the range: -2,147,483,648 to 2,147,483,647, **100**

[7] Optional only for Modicon_5digit addressing, and only if Single writes do not need to be forced

6.3.3 Map Descriptor Examples

All three examples below are addressing the same Modbus registers.

```
//  Server Side Map Descriptors where Node Address_Type is ADU
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , Data_Type , Address , Length , Data_Array_Low_Scale , Data_Array_High_Scale , Node_Low_Scale , Node_High_Scale
SMD_AI_01 , DA_AI_01 , 0 , Passive , MB_Srv_11 , Input_Register , 1 , 200 , 0 , 100 , 0 , 10000
SMD_AO_01 , DA_AO_01 , 0 , Passive , MB_srv_11 , Holding_Register , 1 , 200 , 0 , 100 , 0 , 10000
```

```
//  Server Side Map Descriptors where Node Address_Type is PDU
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , Data_Type , Address , Length , Data_Array_Low_Scale , Data_Array_High_Scale , Node_Low_Scale , Node_High_Scale
SMD_AI_02 , DA_AI_02 , 0 , Passive , MB_Srv_12 , FC04 , 0 , 200 , 0 , 100 , 0 , 10000
SMD_AO_02 , DA_AO_02 , 0 , Passive , MB_srv_12 , FC03 , 0 , 200 , 0 , 100 , 0 , 10000
```

```
// For Nodes where Address_Type is Modicon_5digit.
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , Address , Length , Data_Array_Low_Scale , Data_Array_High_Scale , Node_Low_Scale , Node_High_Scale
SMD_AI_01 , DA_AI_01 , 0 , Passive , MBP_Srv_13 , 30001 , 200 , 0 , 100 , 0 , 10000
SMD_AO_01 , DA_AO_01 , 0 , Passive , MBP_Srv_13 , 40001 , 200 , 0 , 100 , 0 , 10000
```

Appendix A. Useful Features

Appendix A.1. Managing Floating points with Modbus

Modbus as a standard does not support floating point formats. Many vendors have written higher level communications software to use two 16 bit registers to represent floating point or 32 bit integers. This requires conversion software on both ends of the communication channel. The FieldServer supports this function and also provides other options to resolve this issue.

Appendix A.1.1. Transferring non-integer values with one register

It is possible to represent values higher than 32767 using one register in one of two ways:

- Declare data arrays as type Uint16 (Unsigned integer). This will give you a range from 0 to 65535.
- Use the scaling function on the FieldServer, which will allow you to set up any range, with 16 bit resolution.

The following example shows how scaling can be achieved on the Server side of the configuration. Note that scaling can also be done on the Client side to scale down a value that was scaled up by a Modbus vendor. Further information regarding scaling can be found in the FieldServer Configuration manual.

Example :

```
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , Address , Length , Data_Array_Low_Scale , Data_Array_High_Scale , Node_Low_Scale , Node_High_Scale
SMD_AI1             , DA_AI_01        , 0                 , Passive  , MBP_Srv_11 , 30001   , 200    , 0                  , 100                   , 0             , 10000
SMD_AO1             , DA_AO_01        , 0                 , Passive  , MBP_Srv_11 , 40001   , 200    , 0                  , 100                   , 0             , 10000
```

This example multiplies the values in the data array by 100 (10000 on Node_High_Scale is 100X larger than 100 on Data_Array_High_Scale). This is most commonly used when the user wants to introduce values after the decimal point. For example, a value of 75.6 will be sent as 7560, which can then be rescaled by the Modbus master.

Appendix A.1.2. Transferring Float/32 bit values with two registers

If a Modbus Server sends two consecutive registers to the FieldServer representing either a floating point value or a 32 bit integer value, the FieldServer can combine and decode these registers back into their original format. To do this declare Data Array of type Float or UINT32 and set the Map Descriptor Data_Type as 'Float_Reg' or '32-Bit_Reg' etc

Example:

```
Data_Arrays
Data_Array_Name   , Data_Format   , Data_Array_Length
DA1               , Float         , 20
DA2               , UInt32        , 20
DA3               , Float         ,20
DA4               ,UInt32         ,20
```

```
//  Client Side Map Descriptors
// For Nodes where Address_Type is PDU
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name,  , Data_Type      , Address , Length , Scan_Interval
CMD_AO_01           , DA1             , 0                 , Rdbc     , MODBUS DEVICE2 , Float_Reg    , 0       , 20     , 1.000s
CMD_AO_02           , DA2             , 0                 , Rdbc     , MODBUS DEVICE2 , 32-Bit_Reg   , 0       , 20     , 1.000s
CMD_AI_01           , DA3             , 0                 , Rdbc     , MODBUS DEVICE2 , Input_Float  , 0       , 20     , 1.000s
CMD_AI_02           , DA4             , 0                 , Rdbc     , MODBUS DEVICE2 , Input_Reg_32Bit , 0    , 20     , 1.000s
```

Each Map Descriptor will read 20 pairs of registers and store them as 32-bit floating number or 32-bit Integer.

If some how server device send swapped registers (low value register first) then use corresponding _swapped data_types.

Note: Please note starting addresses per address_type, see section 5.3.2

Appendix A.2. Node_Offline_Response

In systems where data is being collected from multiple Server Nodes and made available on a FieldServer configured as a Modbus TCP Server, when a Server Node goes offline the default behavior of the FieldServer would be to stop responding to polls for this data. This might not be what the user wants. Various options exist making it possible to signal that the data quality has gone bad without creating error conditions in systems sensitive to the default option.

The following options can be configured under the Node parameter, Node_Offline_Response, to set the response of the FieldServer to the Modbus TCP Client when the Server Node supplying the data is offline:

- No_Response - this is the default option. The FieldServer simply does not respond when the corresponding Server Node is offline.
- Old_Data - The FieldServer will respond, but with the last known value of the data. This maintains the communication link in an active state, but may hide the fact that the Server Node is offline.
- Zero_Data - The FieldServer will respond, but with the data values set to zero. If the user normally expects non-zero values, this option will signal the offline condition without disrupting communications.

- FFFF_Data - The FieldServer will respond, but with the data values set to FFFF (hex). If the user normally expects other values, this option will signal the offline condition without disrupting communications.

When configured as a Server this parameter can force a desired exception response as follows:

- Node_Offline_Message or Exception_4 - FieldServer's response will be Exception 4
- Gateway_Path_Unavailable or Exception_A - FieldServer's response will be Exception A
- Gateway_Device_Failed or Exception_B - FieldServer's response will be Exception B

Example:

```
Nodes
Node_Name   ,Node_ID  ,Protocol      , Node_Offline_Response      , Port
DEV11       , 11      , Modbus/TCP   , No_Response                , -
DEV12       , 12      , Modbus/TCP   , Old_Data                   , -
DEV15       , 15      , Modbus/TCP   , Zero_Data                  , -
DEV16       , 16      , Modbus/TCP   , FFFF_Data                  , -
DEV17       , 17      , Modbus/TCP   , Exception_4.               , -
DEV18       , 18      , Modbus/TCP   , Gateway_Path_Unavailable   , -
```

Appendix B. Troubleshooting

Appendix B.1. Server Configuration of System Station Address

When using the FieldServer as a Modbus Server, the FieldServer System Station address must be configured to be different from any of the configured Modbus Server Node_ID's. Configuring these to be the same invokes proprietary system information to be transmitted, and should therefore be avoided.

Appendix C. Reference

Appendix C.1. Data Types

If Node parameter Address_Type is set as ADU or PDU, then Data_Type must be specified as follows

Appendix C.1.1. For Address_Type ADU :

Address range	Data_Type	Function Code (Write)	Function Code (Read)
1 - 65536	Coil	15	1
1 – 65536	Discrete_Input	n/a.	2
1 – 65536	Input_Register	n/a.	4
1 - 65536	Holding_Register	16	3

Appendix C.1.2. For Address_Type PDU :

Address range	Data_Type	Function Code (Write)	Function Code (Read)
0 - 65535	FC01	15	1
0 – 65535	FC02	n/a.	2
0 – 65535	FC04	n/a.	4
0 – 65535	FC03	16	3

Appendix C.1.3. For Address_Type Modicon_5digit

When a Modbus address range is specified, a particular Data Type is implied. The defaults are as follows:

Address range	Data_Type	Function Code (Write)	Function Code (Read)
00001 - 09999	Coil	5,15	1
10001 - 19999	Discrete_Input	n/a.	2
30001 - 39999	Input_Register	n/a.	4
40001 - 49999	Holding_Register	6,16	3

Appendix C.2. Read/write Operation

If writing multiple registers the write function will 16

If writing multiple coils the write function will 15

If writing a single register the write function will be 6 unless Write_FNC parameter is set to “Multiple’

If writing a single coil the write function will be 5 unless Write_FNC parameter is set to “Multiple’

.

Appendix C.3. Driver Error Messages

Message	Description/Action
MB_TCP:#01 FYI. Server response extra bytes ignored. Cnt=%d %#x	This message is printed when the TCP frame contains more bytes than a single Modbus_TCP message but insufficient extra bytes to form a second complete Modbus message. There is no explanation for the 'padding' bytes, but since the Driver ignores the extra bytes and processes the complete message correctly, the message can be ignored. The driver prints this message once. It is suppressed on subsequent occurrences.
MB_TCP:#02 FYI. Master poll extra bytes ignored. Cnt=%d %#x	
MB_TCP:#03 Err. TCP Frame has multiple MB_TCP messages. Ignored 2nd	The driver has detected enough bytes in the TCP frame for two complete Modbus_TCP messages. The second message is ignored. If this is a problem, re-configure the remote node so that only one Modbus_TCP message is contained in a single TCP frame. The driver prints this message once. It is suppressed on subsequent occurrences