RDO® PRO-X and RDO® Titan

Modbus and SDI-12 Reference Guide

July 2015

rev. 003

Table of Contents

1. Controller Requirements and Connections

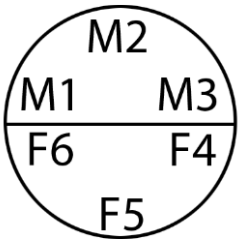
The Multi-PRO 400 may be connected to a controller or logger for communication via:

- SDI-12
- RS485 Modbus
- RS232 Modbus (with converter)

1.1 Wiring Overview

Refer to the diagrams on the following pages. Trim and insulate unused wires. The shielded wire should be wired to a chassis ground or earth ground.

Rugged Cable



Signal	Color	Pin
Ground/Return	Black	6
External Power	Red	5
No Connection	Brown	4
RS485 (-)	Green	3
RS485 (+)	Blue	2
SDI-12	White	1

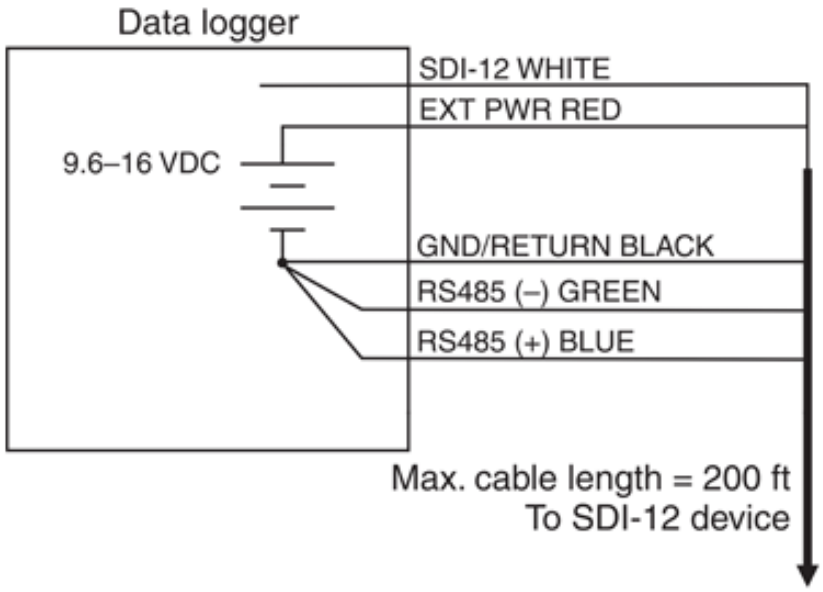
1.2 Power Connections

The red wire provides power for all system modes.

1.3 SDI-12 Wiring Diagram

Cable length must not exceed 60.9 m (200 ft).

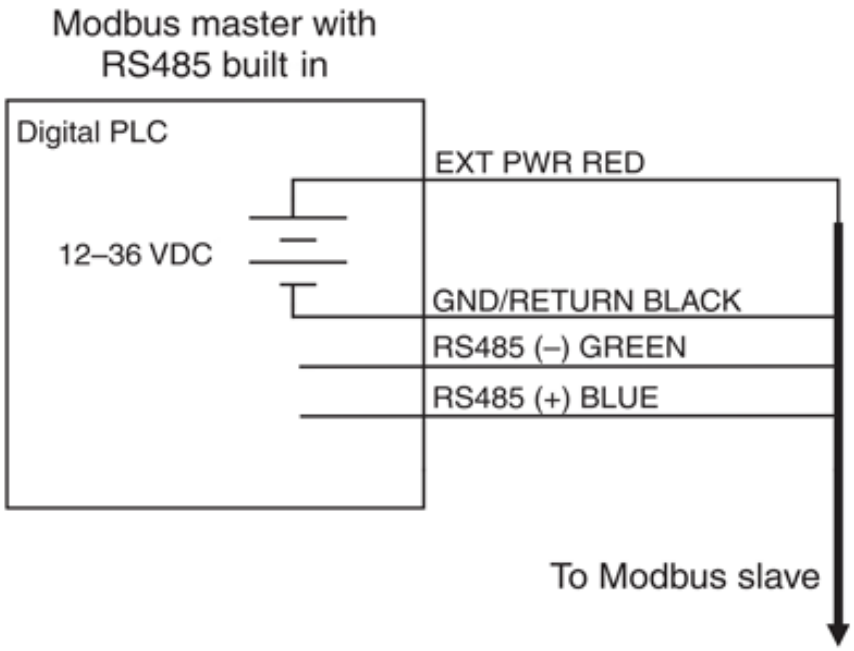
Signal	Color	Pin
Ground/Return	Black	6
External Power (9.6-16 VDC)	Red	5
SDI-12	White	1



1.4 Modbus Master RS485 Wiring Diagram

Cable length must not exceed 1219.2 m (4000 ft).

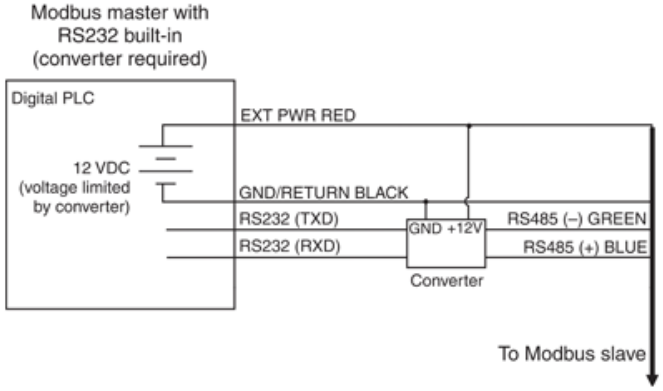
Signal	Color	Pin
Ground/Return	Black	6
External Power (12-36 VDC)	Red	5
RS485 (-)	Green	3
RS485 (+)	Blue	2

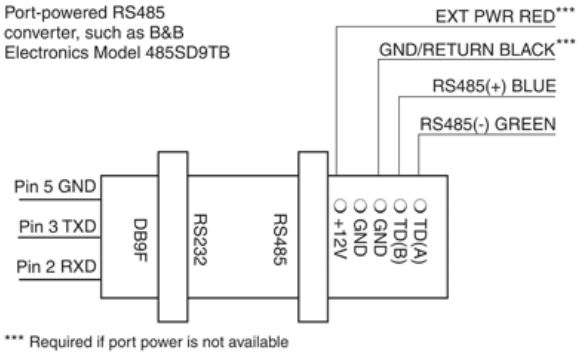


1.5 Modbus Master RS232 Wiring Diagram (Converter Required)

Cable length between Master and Slave must not exceed 1219.2 m (4000 ft). Rugged Cable length between Master and Converter must not exceed 6 m (20 ft).

Signal	**Color**	**Pin**
Ground/Return	Black	6
External Power (12-36 VDC, voltage limited by converter)	Red	5
RS485 (-)	Green	3
RS485 (+)	Blue	2



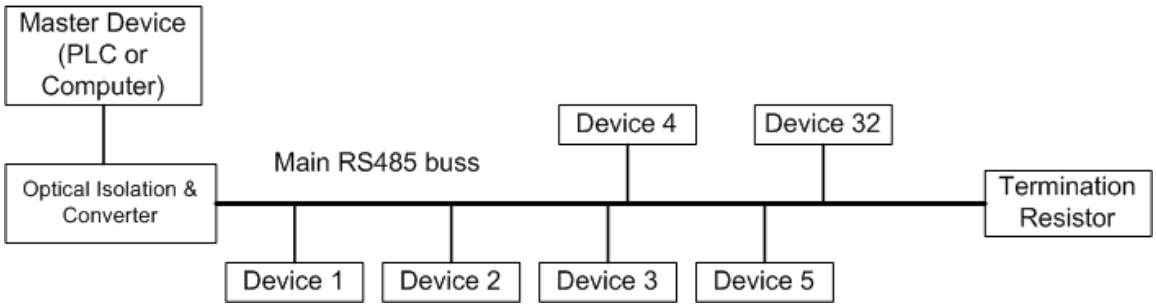


1.6 RS485 Network Guidelines

In-Situ uses RS485 as its main digital communications link. RS485/422 is often used in an industrial setting as a small device network. There are some installation guidelines to follow when configuring an RS485 network with implications to use with the instrument.

RS485 Rule 1

RS485 is a bus network. It does not work when configured in a star network topology. This means that a user can have a network that looks like 1 long wire (up to 4000 ft) with short stubs hanging off the main branch with a device. Each stub must be less than 1 meter in length. See picture below:



RS485 Rule 2

There should only be a 100 ohm terminating resistor at the end of the network. The bus is terminated on the long main bus wire at the opposite end from the Master.

RS485 Rule 3

This rule is not specific to RS485; rather it applies to any situation where you have long wires running across the ground or in the ground connected back to a computer. Always add an optical isolator to the link between the main bus wire and the Master device. This reduces the chance that a nearby lightning strike will damage the Master device.

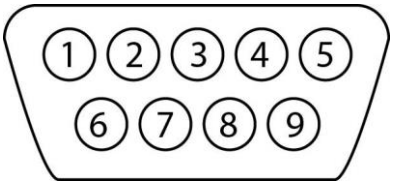
RS485 Rule 4

There can be only 32 devices per network, including the Master.

Implications to users of this instrument are as follows:

These devices are typically deployed on a cable of much greater length than the 1 meter stub supported by RS485. The above documented Rule 1 requires that only two devices are on an individual RS485 link, the PLC and the instrument. Many PLC's support multiple RS485 networks which can be used to connect multiple instruments to a single PLC.

DB-9 Pin Diagram



Pin	Signal Name	
1	Carrier Detector	DCD
2	Receive Data	RXD
3	Transmit Data	TXD
4	Data Terminal Ready	DTR
5	Signal Ground/Common	GND
6	Data Set Ready	DSR
7	Request to Send	RTS
8	Clear to Send	CTS
9	Ring Indicator	RI

2. SDI-12 Interface

The RDO PRO Instrument adheres to the Serial Digital Interface Standard for Microprocessor-based sensors, version 1.3, and the extensions to the specifications identified in this document. This section identifies the device-specific implementation of the standard. For more information on SDI-12 see *www.sdi-12.org*.

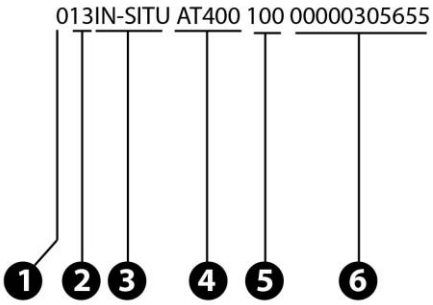
2.1 Instrument Defaults

The RDO PRO Instrument leaves the factory with the following settings:

Device Address	0
Parameters	1: DO concentration in mg/L 2: DO percent saturation 3: Temperature in °C

2.2 Device Identification

In response to the Send Identification command, the instruments respond as follows:



1.	Device address
2.	SDI-12 compatibility (Version 1.3)
3.	Manufacturer
4.	Instrument model
5.	Firmware version (100 = 1.00)
6.	Serial number

2.3 Basic Commands

The following table lists each basic SDI-12 command, its format, and the format of each response.

Name	Command	Response
Address Query	?!	a<CR><LF> The wildcard address '?' character is supported only for the Address Query command. It is ignored as an invalid address for all other commands.
Acknowledge Active	a!	a<CR><LF> Basic address characters in the range '0' to '9' and the extended address characters in the ranges 'A' to 'Z' and 'a' to 'z' are supported. All other characters are ignored as an invalid address. The default address is '0'.
Change Address	aAb!	b<CR><LF> Software changeable addresses and the Change Address command are supported.
Send Identification	aI!	a13IN-SITU AT400 vvv xxxxxxxxxx<CR><LF> vvv = device firmware version * 100 (120 = 1.20) xxx = 10-digit device serial number with leading zeroes
Start Verification	aV!	a0011<CR><LF> One result is available for reading by the Send Data command.
Send Data	aD0!	a+n<CR><LF> n = lower 16 bits of the device status (0-65535)
Additional Data	aD1!…aD9!	a<CR><LF> No values are returned after an Additional Data command.

Name	Command	Response
Start Measurement Start Measurement CRC	aM! aMC!	a002n<CR><LF> n parameters will be available for reading by the Send Data command within 2 seconds. A service request (a<CR><LF>) will be sent when the parameters are ready. The number of parameters returned is determined by the SDI-12 configuration file. The default value for n is 3. The default parameters are: 1: DO concentration in mg/L 2: DO percent saturation 3: Temperature in °C
Send Data	aD0!	a<values><CR><LF> or a<values><CRC><CR><LF> The number and type of parameters returned is determined by the SDI-12 configuration file. At most 3 parameters are returned in a Send Data command. If more than 3 parameters are available, they are returned using the Additional Data command.
Additional Data	aD1!...aD9!	a<values><CR><LF> or a<values><CRC><CR><LF> No values are returned after an Additional Data command.
Additional Measurements Additional with CRC	aM1! … aM9! aMC1!… aMC9!	a0000<CR><LF> No additional measurements are started by the device.
Send Data	aD0!	a<CR><LF>
Additional Data	aD1! … aD9!	No values are returned after an Additional Measurement command.

Name	Command	Response
Start Concurrent	aC!	a002nn<CR><LF>
Start Concurrent CRC	aCC!	nn parameters will be available for reading by the Send Data command within 2 seconds. No service request will be sent when the parameters are ready. The number of parameters returned is determined by the SDI-12 configuration file in the same manner as a Start Measurement command.
Send Data	aD0!	a<values><CR><LF> or a<values><CRC><CR><LF> The number and type of parameters returned is determined by the SDI-12 configuration file in the same manner as a Start Measurement command.
Additional Data	aD1! … aD9!	a<values><CR><LF> or a<values><CRC><CR><LF> The number and type of parameters returned is determined by the SDI-12 configuration file in the same manner as a Start Measurement command.
Additional Concurrent	aC1! … aC9!	a00000<CR><LF>
Additional with CRC	aCC1! … aCC9!	No additional concurrent measurements are started by the device.
Send Data	aD0!	a<CR><LF> or a<CRC><CR><LF>
Additional Data	aD1! … aD9!	No values are returned after an additional Concurrent Measurement command.
Continuous Measurement Continuous with CRC	aR0! … aR9! aRC0! … aRC9!	a<CR><LF> No values shall be returned after a Continuous Measurement command.

2.4 Extended Commands

The RDO PRO and Titan support the following extended SDI-12 commands.

Name	Command	Response
ISCO Compatibility	aXPR0!	aIxIxIxIx<CR><LF> where each Ix is a character pair identifying the parameter and units for each measurement. The number of Ix pairs must equal the number of data values returned for the Start Measurement and Start Concurrent commands. The following pairs are supported: DO Concentration, mg/L = “F0” Temperature, °C = “A0” Temperature, °F = “A1” DO % Saturation = “F1” If not listed above, all other parameter and unit combinations will return “??”.
ISCO Additional	aXPR1…XPR9!	a<CR><LF> No values are returned after an additional ISCO compatibility command.
Set Factory Defaults	aXFD!	a0011<CR><LF> One result is available in 1 second for reading by the Send Data command. Restores all settings and calibration values to their factory defaults.
Send Data	aD0!	a+s<CR><LF> s = command status, 1 = command successful, 0 = the command failed.
Set Output Sequence	aXOnnn!	a0001,CR><LF>where nnn = one or more parameter characters in required output order (C = DO concentration, T = Temperature, S = DO % saturation, P = oxygen partial pressure), in the required output order.
Send Data	aD0!	One result is available immediately for reading by the Send Data command.a+s<CR><LF>where: s = command status, 1 = command successful, 0 = invalid parameter characters.
Set DO Concentration Units	aXoUnnn!	a0001<CR><LF>where: nnn = the concentration units ID (same values as specified for Modbus register 0041). Three digits are required. One result is available immediately for reading by the Send Data command.
Send Data	aD0!	a+s<CR><LF>where: s = command status, 1 = command successful, 0 = invalid units ID or an attempt to change units while the device is logging.

Name	Command	Response
Set Temperature Units	aXTUnnn!	a0001<CR><LF>where: nnn = the temperature units id (same values as specified for Modbus register 40049). Three digits are required. One result is available immediately for reading by the Send Data command.
Send Data	aD0!	a+s<CR><LF>where: s = command status, 1 = command successful, 0 = invalid units id or an attempt to change units while the device is logging.
Set Calibration Slope	aXCSpd.d!	a0001<CR><LF>where:pd.d = cell constantp = polarity sign (+ or -)d = digits (1 to 7). = decimal point (optional)One result is available immediately for reading by the Send Data command.
Send Data	aD0!	a+s<CR><LF>where: s = command status, 1 = command successful, 0 = invalid parameter characters or an attempt to change the cell constant while the device is logging.
Set Calibration Offset	aXoOpd.d!	a0001<CR><LF>where:pd.d = conversion factor to pptp = polarity sign (+ or -)d = digits (1 to 7). = decimal point (optional)One result is available immediately for reading by the Send Data command.
Send Data	aD0!	where: s = command status, 1 = command successful, 0 = invalid parameter characters or an attempt to change the conversion factor while the device is logging.
Communication Diagnostics	aXCD!	a+A+C<CR><LF> A = contents of Modbus device address register 9200 C = contents of Modbus serial communication configuration register 9201

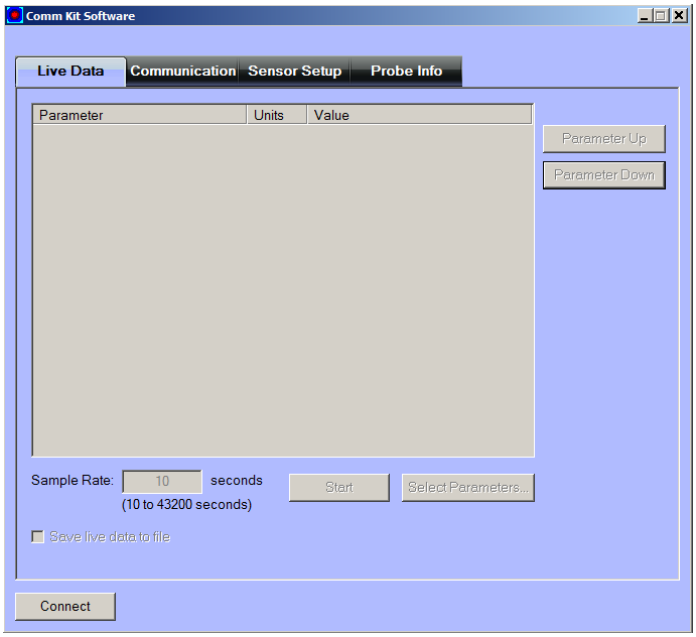
2.5 SDI-12 Configuration File

The SDI-12 configuration file can be edited using the Comm Kit Software and the Communication Device.

2.5.1 Connect the Communication Device and Comm Kit Software

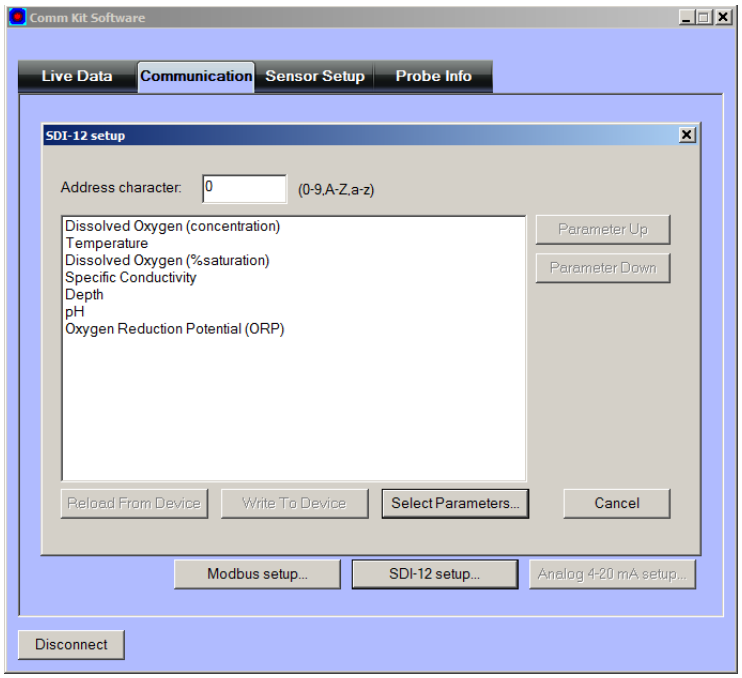
The Communication Device connects a stripped-and-tinned instrument to a computer via USB connection and enables a connection with the Comm Kit Software.

1. The communication device includes an electrical connection diagram label. To attach the sensor to the communication device, depress a lever and insert the appropriate wire in the location specified by the diagram.
2. Attach the USB connection to a computer.
3. Open the Comm Kit Software and click the Connect button.



2.5.2 SDI-12 Setup

SDI-12 setup allows you to set the instrument address, select the parameters you intend to log, and select the order in which the parameters will appear in the log.



Screen Element	Purpose
Address character (text field)	Allows you to assign a unique SDI-12 address to the instrument. Use 0-9, A-Z, or a-z.
Reload From Device (button)	Restores the settings that were last written to the instrument.
Write to Device (button)	Writes the parameters and the address to the instrument.
Select Parameters (button)	Allows you to enable and disable parameters. Click the Select Parameters button and click the checkboxes to select or clear parameters.
Parameter Up (button)	Allows you to move a parameter higher in the list. Click the parameter and then click the Parameter Up button until the desired location is reached.
Parameter Down (button)	Allows you to move a parameter lower in the list. Click the parameter and then click the Parameter Down button until the desired location is reached.

2.6 Device Status

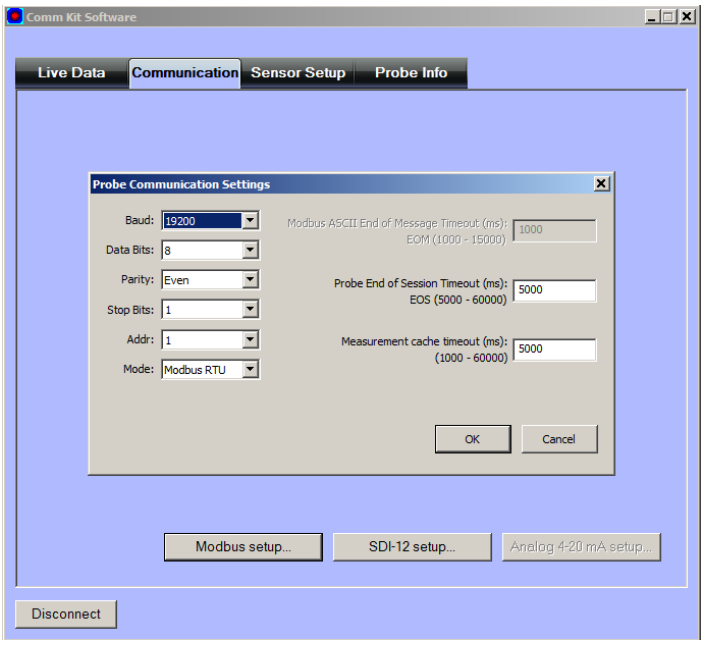
The device status register holds general status information. Each set bit represents a status value. There are a limited number of standardized predefined status values that all devices will support. These predefined status values are contained in the lower register. The upper register is reserved for device specific status values.

Bit	Category	Description
0	Alarm	Sensor high alarm
1	Warning	Sensor high warning
2	Warning	Sensor low warning
3	Alarm	Sensor low alarm
4	Warning	Sensor calibration warning
5	Alarm	Sensor malfunction
6-7	N/A	Reserved
8	N/A	Reserved
9	Status	Device off line
10	Alarm	Device hardware reset occurred
11	Alarm	Device malfunction
12	Status	No external power
13	N/A	Reserved
14	N/A	Reserved
15	N/A	Reserved

3. Modbus Registers—Probe

3.1 Modbus Communication Setup

See Connect the Communication Device on page 18. Click the Modbus setup button and assign instrument settings according to the requirements of your controller or PLC.



4. RDO PRO-X / RDO TITAN Common Registers

The following Modbus registers are specific to the RDO PRO-X and the RDO Titan. More information about Modbus, including protocol specifications can be downloaded from *www.modbus.org*.

Register XXXX	Size (register	Mode & Access Level (R/W)	Data Type	Description
9001	1	R/W	ushort	Device ID (Device ID = 19 for RDO Titan) Device ID (Device ID = 31 for RDO PRO-X)
9002-9003	2	R/W	ulong	Device serial number
9004	3	R/W	time	Manufacture date

4.1 Sensor Status Registers

Register XXXX	Size (register	Mode & Access Level (R/W)	Data Type	Description
0005	3	R1	time	Cap start date/time 0 = no cap
0008	3	R1	time	Cap end of usable life date/time 0 = no cap

4.2 Device Specific Registers

Register XXXX	Size (register	Mode & Access Level (R/W)	Data Type	Description
Dissolved Oxygen Concentration				
0038	2	R1	float	Measured value, C_0
0040	1	R1	ushort	Parameter ID = 20
0041	1	R1/W2	ushort	Units ID 117 = mg/L (default) 118 = µg/L
0042	1	R1	ushort	Data Quality ID (See Sensor Health Table in the Operator's Manual)
0043	2	R1/W3	float	Off line sentinel value (default = 0.0)
0045	1	R1	16 bits	Available Units = 0x0030 (48)
Temperature				
0046	2	R1	float	Measured value
0048	1	R1	ushort	Parameter ID = 1
0049	1	R1/W2	ushort	Units ID 1 = °C (default) 2 = °F
0050	1	R1	ushort	Data Quality ID
0051	2	R1/W3	float	Off line sentinel value (default = 0.0)
0053	1	R1	16 bits	Available Units = 0x0003 (3)
Dissolved Oxygen %Saturation				
0054	2	R1	float	Measured value
0056	1	R1/W2	ushort	Parameter ID = 21
0057	1	R1/W2	ushort	Units ID 177 = percent saturation (default)
0058	1	R1	ushort	Data Quality ID

Register XXXX	Size (register	Mode & Access Level (R/W)	Data Type	Description
0059	2	R1/W3	float	Off line sentinel value (default = 0.0)
0061	1	R1	16 bits	Available Units = 0x0001 (1)
Oxygen Partial Pressure				
0062	2	R1	float	Measured value
0064	1	R1	ushort	Parameter ID = 30 (pressure)
0065	1	R1/W2	ushort	Units ID 26 = Torr (default)
0066	1	R1	ushort	Data Quality ID
0067	2	R1/W3	float	Off line sentinel value (default = 0.0)
0069	1	R1	16 bits	Available Units = 0x0200 (512)

4.3 Sensor Health Table and Data Quality IDs

Abbreviation	Data Quality ID	Text	Description
None	0	None	Normal Data Quality
UC	1	User Cal Expired	Parameter measured without errors using an expired user calibration.
FC	2	Factory Cal Expired	Parameter measured without errors using an expired factory calibration.
ERR	3	Unknown Error	Parameter measured with error, sentinel value supplied.
WU	4	Sensor Warm-up	Sensor is warming up, sentinel value supplied.
DIS	5	Sensor Warning	Parameter measured but does not meet normal quality criteria. The sensor has sustained moderate damage, or the recommended lifespan has been reached.

Abbreviation	Data Quality ID	Text	Description
CAL	6	Sensor Calibrating	Sensor is calibrating, calibration value supplied.
OL	7	Sensor Missing	Sensor communication failed, sentinel value supplied. Make sure the sensor cap is installed and properly seated.

4.4 RDO Pro Calibration Registers.

Register XXXX	Size (register	Mode & Access Level (R/W)	Data Type	Description
0118	2	R1/W3	float	Live salinity value (PSU)
0120	2	R1/W3	float	Default salinity value (PSU, default = 0.0)
0122	2	R1/W3	float	Live barometric pressure (mbar)
0124	2	R1/W3	float	Default barometric pressure (mbar, default = 1013.25)
0126	2	R1/W3	float	100% saturation calibration reading (mg/L)
0128	2	R1/W3	float	100% saturation temperature reading (°C)
0130	2	R1/W3	float	100% saturation salinity value (PSU)
0132	2	R1/W3	float	100% saturation barometric pressure (mbar)
0134	2	R1/W3	float	0% saturation calibration reading (mg/L)
0136	2	R1/W3	float	0% saturation temperature reading (°C)
0138	2	R1/W3	float	Calibration slope (default = 1.0)
0140	2	R1/W3	float	Calibration offset (default = 0.0)

4.4.1 Live Salinity Value

The live salinity value is used to correct the oxygen concentration value for salinity. Values must be written in Practical Salinity Units (PSU) in the range 0 to 42 PSU. This is not a measured parameter.

4.4.2 Default Salinity Value

The default salinity value is loaded into the live salinity value register when power is first applied to the probe. The default salinity value is used in calculations until a live salinity value is written. This is not a measured parameter.

4.4.3 Live Barometric Pressure

The live barometric pressure is used in the calculation of percent saturation and to determine the theoretical saturation point during calibration. Values must be written in millibars in the range 506.625 to 1114.675 mbar. This is not a measured parameter.

4.4.4 Default Barometric Pressure

The default barometric pressure is loaded into the live barometric pressure register when power is applied to the probe. The default barometric pressure is used in calculations until a live barometric pressure is written. This is not a measured parameter.

4.4.5 100% Saturation Calibration Values

These values represent the sensor conditions while the probe is in a 100% saturation calibration environment. These are not measured values, they are written by the controller during the calibration process.

Writes to these registers are only accepted if the probe is in the calibration mode. The probe will return exception 0x85 (invalid device command sequence) if an attempt is made to write these registers when the calibration mode is off.

4.4.6 0% Saturation Calibration Values

These values represent the sensor conditions while the probe is in a 0% saturation calibration environment. These are not measured values, they are written by the controller during the calibration process.

Writes to these registers are only accepted if the probe is in the calibration mode. The probe will return exception 0x85 (invalid device command sequence) if an attempt is made to write these registers when the calibration mode is off.

4.4.7 Calibration Slope and Offset

These values represent the slope and offset that will be applied to the raw concentration reading from the sensor to generate the final values reported by the sensor parameters. These registers may be written independently of the normal internal calibration procedure.

4.4.8 Calibration Procedure

The probe is calibrated using the following procedure.

1. Write the Calibration Mode On command (0xE000) to the sensor command register 49305.
2. Update the live salinity and barometric pressure registers if necessary.
3. Prompt the user to place the probe in a 100% saturation environment.
4. Read the oxygen concentration and temperature parameters. When these values have reached equilibrium, record them in their respective 100% saturation calibration registers. Write the current live salinity and barometric pressure readings to their respective calibration registers.
5. Prompt the user to place the probe in a 0% saturation environment. When these registers have reached equilibrium, record them in their respective 0% saturation calibration registers. If a zero calibration is not to be performed, these registers can be set to zero or left at their previous values.
6. Write the Calibration Update command (0xE001) to the sensor command register. The sensor will calculate a new slope and offset, will write the current time to the last user calibration time register, and set the next user calibration time register to zero (disabled). If the concentrations at 100% and 0% saturation are equal the probe will return an exception response with code 0x97 (invalid calibration) and not attempt to compute a new slope and offset due to possible division by zero. If the slope does not calculate between 0.85 and 1.20 inclusive, or the offset does not calculate between -0.2 and +0.2 inclusive, the probe will return an exception response with code 0x97 (invalid calibration) The slope and offset will be available for read but will not be committed to flash.
7. Optional: read the last user calibration time register, add the next calibration interval, and write the result to the next user calibration time register.
8. Write the Calibration Mode Off command (0xE002) to the sensor command register to place the sensor in normal operation. If the calibration mode is turned off without a calibration update command, or the calibration command returned an exception, the previous calibration shall be restored.

5. Appendix—Modbus Additional Information

The Multi-PRO 400 Instrument supports Modbus as its primary communication protocol.
This appendix contains a brief Modbus tutorial intended to accelerate learning for a person who is not familiar with the protocol. This document is not an official protocol document. More information about Modbus, including protocol specifications, can be downloaded from *www.modbus.org*.
The remaining sections of this document provide the information required to program a PLC/DCS and/or SCADA system to poll data. The user is expected to have a working knowledge of Modbus.

5.1 Modbus Tutorial

Modbus is a lightweight communication protocol developed in the late 1970's by Modicon as a digital communication protocol for its PLC's. The protocol requires very little code space and processing power to implement and has become a worldwide favorite for embedded devices. Modbus is royalty free and the specifications and standards can be downloaded from the web.

Modbus is a transport protocol. What this means is that Modbus does not have any protocol features that indicate what types of data are being transported in a message packet. This is similar to TCP/IP, the internet protocol standard. If we use TCP/IP as an example, consider that when a person connects to a web site, the primary data content being moved via TCP/IP is HTML. When a person downloads an instrument manual from an FTP site, typically the data content being moved is an Adobe PDF file. In both cases, TCP/IP is used as the protocol. The TCP/IP protocol simply ensures all the bytes (whatever they represent) are transferred from point A to point B correctly. Modbus is similar in concept. It provides a transport mechanism to move data safely over a communication link from a device to a computer.

Modbus can be used on a wide variety of communication links. In most applications, the protocol is used over an RS485 or RS232 link. This is because these types of communication links are inexpensive and efficient, perfectly suited for communication with embedded devices. Modbus can also be used over wireless radios, satellites, TCP/IP (Ethernet, token ring, etc.) and any other protocol-independent communication link.

Modbus is designed to be a Master/Slave protocol. This means that the protocol assumes that a single Master computer will initiate control and commands to the slave devices. The slave devices do not send any data on the communication link unless specifically asked for that data by the Master

5.2 Modbus Modes

Modbus message packets can be formatted in three ways, RTU, ASCII and IP.

- RTU is the format of choice for hard connected serial connections such as RS485 or RS232 because it is the most compact and therefore the most efficient.
- ASCII format is required for any kind of wireless serial communications because it eliminates the message timing requirements needed for RTU mode. Message timing can be erratic over a wireless link.
- IP formatted messages can be used when the messages are transported using a secondary transport protocol such as TCP/IP. In this case the secondary transport protocol ensures that all of the packet bytes are transported correctly. Additionally, this format provides for packet sequence numbering because the secondary transport layer eliminates the synchronous Master/Slave nature of the serial RTU/ASCII formats.

5.3 Protocol Overview

Modbus provides message structures to read and write data to/from a device. The protocol also provides for the extension of the protocol to permit customization of the message structures. The instrument manufacturer adheres to the standard read/write message structures in its implementations. Modbus does not provide as part of the standard, a suitable file transfer message structure. For this function, the manufacturer has used a protocol extension to satisfy the downloading of data files from the device.

5.4 General Message Formats

The general message format consists of a device address, a function code, a data payload and a message validity checksum. The message format for each of the three modes changes as described below.

5.5 RTU Message Format

The RTU message format allows the transmission of bytes of data encompassing the full range of values per byte 0-255. There are no characters indicating the start or end of the packet. The end of packet is signaled by a time delay equivalent to a 3 byte transfer time on the communication link without any data transmitted.

5.5.1 Computer (Master) Message Format

Device Address	Function Code	Data Payload	CRC

- Device Address: 1 byte field with a value ranging from 1 to 247. Broadcast address is 0.
- Function Code: 1 byte field with a value range 1-127 representing the standard or extended function code. The function code tells the device what operation is to take place (i.e. Read/Write).
- Data Payload: 0-N bytes with information required to complete the requested function code operation.
- CRC: 2 bytes with a value computed mathematically from the message bytes. This value can be computed on both sides of the link and verified to ensure that the bits of the message were transmitted and received correctly.

5.5.2 Device (Slave) Response Format

Device Address	Function Code	Data Payload	CRC

- Device Address: Echo of device address sent in the message to the device. A broadcast message will not generate a response.
- Function Code: Echo of the function code sent to the device in the message packet. If an error occurs, the top bit of the byte is set and the data payload is the 1 byte error code from the device.
- Data Payload: 0-N bytes with response data from the device. For an error response, the payload will be a 1 byte value 1-255.
- CRC: 2 bytes with a value computed mathematically from the message bytes. This value can be computed on both sides of the link and verified to ensure that the bits of the message were transmitted and received correctly.

5.6 ASCII Message Format

The ASCII formatted message is almost identical in content to the RTU formatted message with the addition of the Start Of Packet (SOP) and End Of Packet (EOP) characters. The SOP character is a ':' and the EOP is the combination carriage return <CR> (0x0D) followed by a linefeed <LF> (0x0A).

The contents of the packet are converted to 2 byte hex characters 0-9 and A-Z. For example, the 1 byte device address 25 would be two bytes 0x3235 where 0x32 is the ASCII character '2' and 0x35 is the ASCII character '5'. This ensures that the message contents never contain SOP or EOP characters.

The check value uses a different mathematical algorithm and is called an LRC.

5.6.1 Computer (Master) Message Format

Start Packet	Device Address	Function Code	Data Payload	LRC	End Packet

- Start Packet: the ':' character signals the start of an ASCII packet.
- Device Address: 2 byte field containing the device address 1-247 in hex characters.
- Function Code: 2 byte field with a value range 1-127 representing the standard or extended function code in hex characters.
- Data Payload: 0-N bytes with information required to complete the requested function code operation. Each data byte is represented in its two byte hex character format.
- LRC: 2 bytes represented in hex characters with a value computed mathematically from the message bytes. This value can be computed on both sides of the link and verified to ensure that the bits of the message were transmitted and received correctly.
- End Packet the <CR><LF>characters.

5.6.2 Device (Slave) Response Format

Start Packet	Device Address	Function Code	Data Payload	LRC	End Packet

- Start Packet: the ':' character signals the start of an ASCII packet.
- Device Address: 2 byte field containing the device address 1-247 in hex characters.
- Function Code: 2 byte field with a value range 1-127 representing the standard or extended function code in hex characters.
- Data Payload: 0-N bytes with response data from the device in hex characters. For an error response, the payload will be a 1 byte value 1-255.
- LRC: 2 bytes represented in hex characters with a value computed mathematically from the message bytes. This value can be computed on both sides of the link and verified to ensure that the bits of the message were transmitted and received correctly.
- End Packet: the <CR><LF> characters.

5.7 IP Message Format

The IP message format is based on the RTU format but eliminates the CRC because TCP/IP will ensure that the message bytes are transmitted correctly. The instrument does not support IP format. However, the protocol can be used through a connected telemetry device.

TCP/IP is an asynchronous protocol. The significance is that the device must send its response to the address of the sender (computer). This differs from the serial RTU and ASCII formats where the protocol assumes only a single master device (computer) which does not have an address. For this reason, the IP format has a different header than the RTU serial formatted message.

5.7.1 Computer (Master) Message Format

Xac ID	Protocol ID	Msg Length	Device Address	Function Code	Data Payload

- Xac ID: 2 byte transaction id to provide for asynchronous messages.

- Protocol ID: 2 byte field – always 0.
- Length: 2 byte field represents number of bytes following.
- Device Address: 1 byte field with a value ranging from 1 to 247. Broadcast address is 0.
- Data Payload: 0-N bytes with information required to complete the requested function code operation.

5.7.2 Device (Slave) Message Format

Xac ID	Protocol ID	Msg Length	Device Address	Function Code	Data Payload

- Xac ID: transaction id echoed from message.
- Protocol ID: 2 byte field – always 0.
- Length: 2 byte field represents number of bytes following.
- Device Address: Echoed from message. No response for broadcast.
- Data Payload: 0-N bytes with information required to complete the requested function code operation.

5.8 Data Addresses (Registers)

As shown in the section pertaining to message formats, the function code from the Master's message tells the Slave device what operation to perform. There are a number of standard function codes defined by Modbus. These functions typically operate on atomic chunks of data historically and generically called registers. A register is quite simply a data address. A Modbus device will always have a published "Register Map" that defines the numerical addresses of data values that can be accessed in the device.

Note: Register Maps typically use a 1-based numbering system whereas the protocol requires the data address/register number passed to a device to be 0-based. In this document, the register maps are 1-based.

Data addresses are associated with two atomic sizes of memory, 1 bit and 2 bytes. These are divided into Read-Only Bits, Read/Write Bits, Read-Only Words and Read/Write Words each with their own associated name as follows:

- Discretes: Read Only Bits
- Coils: Read/Write Bits
- Input Register: Read Only Word
- Holding Register: Read/Write Word

All Manufacturer Registers are Holding Registers.

5.9 Function Codes

Function codes in a message packet tell the device what operation to perform. The function code is always in the byte following the device address. Some of the standard function codes are as follows:

- 01: Read Coil
- 02: Read Discrete
- 03: Read Holding Registers
- 04: Read Input Registers
- 05: Write Coil
- 06: Write single Holding Register
- 15: Write multiple Coils
- 16: Write multiple Holding Registers
- 17: Read slave device ID

- 22: Mask write Holding Register

The slave device ID is an implementation defined response that will vary with manufacturer and/or device. Function codes from 65 to 72 and 100 to 110 can be used as custom function codes.

5.10 Standard Message Formats

In-Situ has implemented a sub-set of the standard function codes in every Modbus-enabled instrument. These message formats are defined in this section.

5.10.1 Read Holding Registers

This command reads one or more registers from a device.

Message (8 bytes):			**Response (5 + N bytes):**		
Address	1 Byte	1-247	Address	1 Byte	1-247
Function Code	1 Byte	0x03	Function Code	1 Byte	0x03
Data Address	2 Bytes	0 to 0xFFFF	Byte Count	1 Byte	0 to 0xFA
Register Count	2 Bytes	0 to 0x7D	Data Payload	N Bytes	
CRC	2 Bytes		CRC	2 Bytes	

Where Byte Count is the #bytes in the Data Payload (does not include CRC bytes).

Byte Count = 2 * Register Count.

5.10.2 Write Holding Register

This command sets a **single** register in a device.

Message (8 bytes):			**Response (8 bytes – message echo):**		
Address	1 Byte	1-247	Address	1 Byte	1-247
Function Code	1 Byte	0x06	Function Code	1 Byte	0x06
Data Address	2 Bytes	0 to 0xFFFF	Data Address	2 Bytes	0 to 0xFFFF
Data Payload	2 Bytes	0 to 0xFFFF	Data Payload	2 Bytes	0 to 0xFFFF
CRC	2 Bytes		CRC	2 Bytes	

5.10.3 Write Holding Registers

This command sets one or more registers in a device.

Message (9 + N bytes):			**Response (8 bytes):**		
Address	1 Byte	1-247	Address	1 Byte	1-247
Function Code	1 Byte	0x10	Function Code	1 Byte	0x10
Data Address	2 Bytes	0 to 0xFFFF	Data Address	2 Bytes	0 to 0xFFFF
Register Count	2 Bytes	1 to 0x7B	Register Count	2 Bytes	0 to 0x78
Byte Count	1 Byte	2 to 0xF0	CRC	2 Bytes	
Data Payload	N Bytes				
CRC	2 Bytes				

Where Byte Count is the #bytes in the Data Payload (does not include CRC bytes). Byte Count = 2*Register Count.

The register count is limited to a single data format field. If an attempt is made to write a data field with an incorrect register count, the device will return a Modbus exception response with error code 0x80.

5.10.4 Mask Write Register

This command will set and/or clear one or more bits in a single register.

Message :			**Response:**		
Address	1 Byte	1-247	Address	1 Byte	1-247
Function Code	1 Byte	0x16	Function Code	1 Byte	0x16
Data Address	2 Bytes	0 to 0xFFFF	Data Address	2 Bytes	0 to 0xFFFF
And_Mask	2 Bytes	0 to 0xFFFF	And_Mask	2 Bytes	0 to 0xFFFF
Or_Mask	2 Bytes	0 to 0xFFFF	Or_Mask	2 Bytes	0 to 0xFFFF
CRC	2 Bytes		CRC	2 Bytes	

Register = (Register Value AND And_Mask) OR (Or_Mask AND (NOT And_Mask))

And_Mask: 0 = bits to change, 1 = bits to leave unchanged.

Or_Mask: 0 = bits to clear, 1 = bits to set.

Example: - set bit 1 (LSB), clear bit 2, leave remaining bits unchanged

Old Value:	0x007E	0000 0000 0111 1110
And_Mask:	0xFFFC	1111 1111 1111 1100
Or_Mask:	0x0001	0000 0000 0000 0001
New Value:	0x007D	0000 0000 0111 1101

This command is useful in a bit mapped register where the Master wants to set some bits of a register that are mapped to a feature without disturbing the other bits of the register that might be mapped to a different feature.

5.10.5 Report Slave ID

This command query's a device for ID information.

Message (4 bytes):			**Response (N bytes):**		
Address	1 Byte	1-247	Address	1 Byte	1-247
Function Code	1 Byte	0x11	Function Code	1 Byte	0x11
CRC	2 Bytes		Byte Count	1 Byte	0 to 0xFF
			Slave ID	1 Byte	0 to 0xFF
			Run Status	1 Byte	0 to 0xFF
			Data Payload	N Bytes	
			CRC	2 Bytes	

- The Data Payload layout is defined in the Slave ID Format section.
- The Byte Count field is the number of bytes from the Slave ID field to the end of the Data Payload (excludes the 2 byte CRC).
- Run Status – must be 0x00 or 0xFF.

Slave ID Layout Example

Byte Offset	Field Description	Type	Value
0	Device Address	byte	1-247
1	Function Code	byte	0x11
2	Byte Count	byte	23
3	Slave ID	byte	0x49 ('I')
4	Run Status Indicator	byte	0x00 = Off, 0xFF = On
Device Specific Info			
5	Slave ID Format Version ID	byte	0x01
Format Version 1 ID Block			
6-7	Manufacturer ID	ushort	0x5349 ('SI')
8-9	Device ID	ushort	
10-11	Application Firmware Version	ushort	Version * 100
12-13	Boot Code Firmware Version	ushort	Version * 100
14-15	Hardware Version	ushort	

Byte Offset	Field Description	Type	Value
16-17	Register Map Template ID	ushort	
18-21	Device Serial Number	ulong	
22-23	Max Message/Response Size (bytes)	ushort	
24-25	Max Baud Rate ID	ushort	
26-27	CRC		

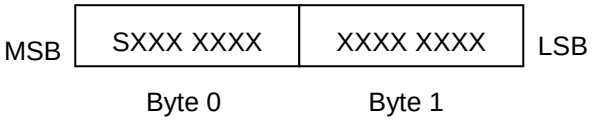
The specific field values such as device ID, baud ID, etc. are documented in the sections that follow.

5.11 Instrument Manufacturer Data Types

Modbus defines all I/O in terms of 2 byte blocks called registers. Modbus does not formally define blocks for floating point values or strings. In the manufacturer implementation, these fundamental types and others are handled by combining two or more registers. The manufacturer data type implementations are defined in the following sections.

5.11.1 Short

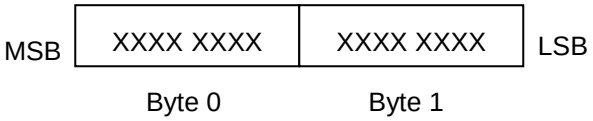
A 2-byte signed integer contained in a single register data address. IEEE standard.



Where S = sign bit

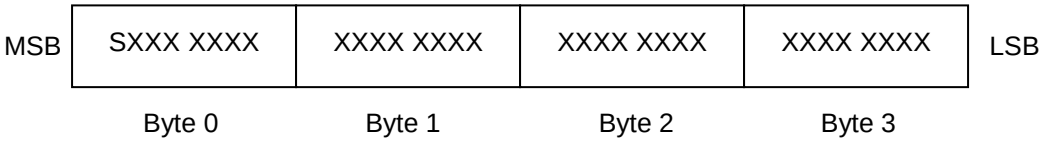
5.11.2 Unsigned Short

A 2-byte unsigned integer contained in a single register data address. IEEE standard.



5.11.3 Long

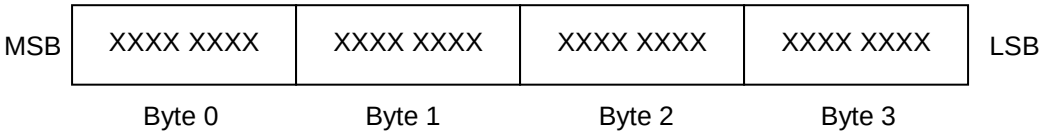
A 4-byte signed integer contained in two register data address's. IEEE standard.



Where S = sign bit

5.11.4 Unsigned Long

A 4-byte unsigned integer contained in two register data address's. IEEE standard.



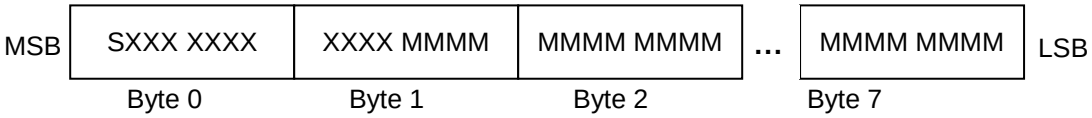
5.11.5 Float

IEEE 4-byte numeric standard – 1 sign bit, 8-bit exponent, 23-bit mantissa.

Where S = sign bit, X = exponent bits and M = mantissa bits.

5.11.6 Double

IEEE 8-byte numeric standard – 1 sign bit, 11-bit exponent, 64-bit mantissa.



Where S = sign bit, X = exponent bits and M = mantissa bits.

5.11.7 Character

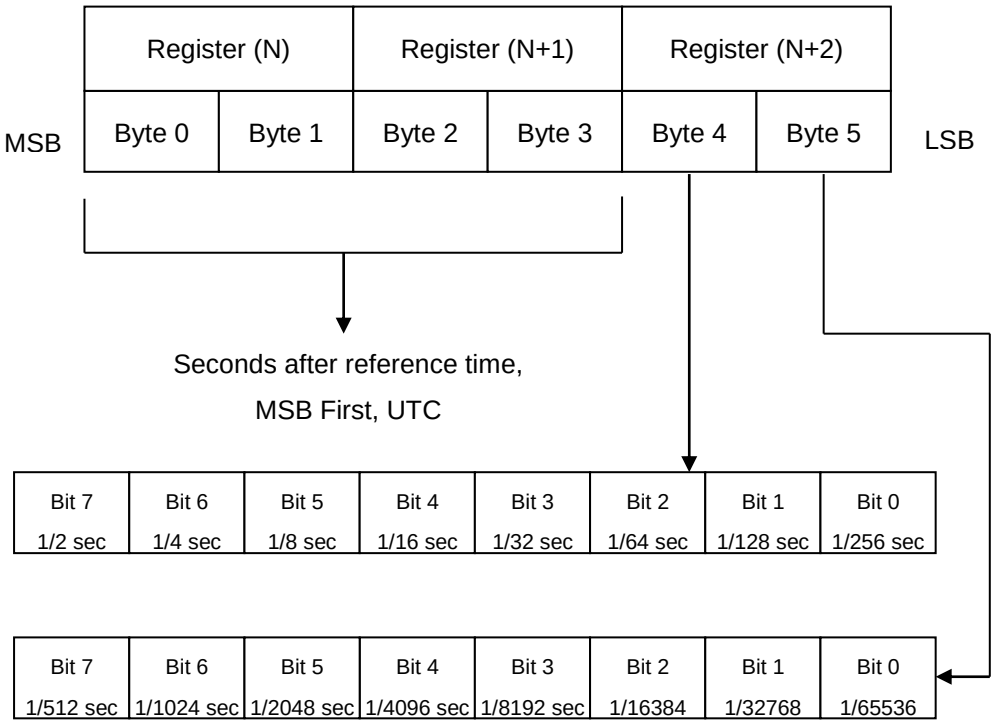
A 2 byte Unicode character contained with a single register data address.

5.11.8 String

The number of consecutive registers in the register map will represent the maximum string length in Unicode characters excluding any termination characters. For example, a 6-register string can have 6 Unicode characters. When reading/writing a string, all register values must be transmitted. If the string to be written does not require the full defined field length, the extra bytes must be padded with the value 0x0000 to reach full length. A string termination character is not required. All registers may contain a character. Failure of the Master device to transmit or request all registers of a string field will be rejected by the Slave device with the Modbus exception error code 0x80.

5.11.9 Time

Time is represented by a 6-byte (3 register) number. The first 4 bytes represent the number of seconds since 00:00:00 January 1, 1970 UTC, MSB first, not adjusted for DST. The 5th and 6th bytes are fractions of a second represented by the bits in powers of 2 starting with the MSB. If a device does not have the ability to support the full fractions of a second resolution available in the time format, unused bits must be set to 0.



Time Example: For a time value of 0x001A5E00C000, the bytes 0x001A5E00 represent the whole number of seconds from the reference time. The bytes 0xC000 represent the additional fractional number of seconds as shown in the diagram above. In this example, the whole number of seconds represents 20 days and the fractional seconds represents 750 ms.

5.12 Exception Codes

The instrument manufacturer supports the standard Modbus exception codes but also provides additional exception codes to assist with troubleshooting problems.

Code	Name	Description
1	Illegal Function	The function code received in the query is not an allowable action for the slave. If a Poll Program Complete command was issued, this code indicates that no program function preceded it.
2	Illegal Data Address	The data address received in the query is not an allowable address for the slave.
3	Illegal Data Value	A value contained in the query data field is not an allowable value for the slave.
4	Slave Device Failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.
5	Acknowledge	The slave has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the master. The master can next issue a Poll Program Complete message to determine if processing is completed. **This exception code may not be supported by the devices**
6	Slave Device Busy	The slave is engaged in processing a long–duration program command. The master should retransmit the message later when the slave is free. **This exception code may not be supported by the devices**
8	Memory Parity Error	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server (or slave) attempted to read record file, but detected a parity error in the memory. The client (or master) can retry the request, but service may be required on the server (or slave) device. **This exception code may not be supported by the devices**
0x0A	Gateway Path Unavailable	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.

Code	Name	Description
		This exception code may not be supported by the devices.
0x0B	Gateway Target Device Failed To Respond	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network. **This exception code may be used by controllers, but is not used by probes.**

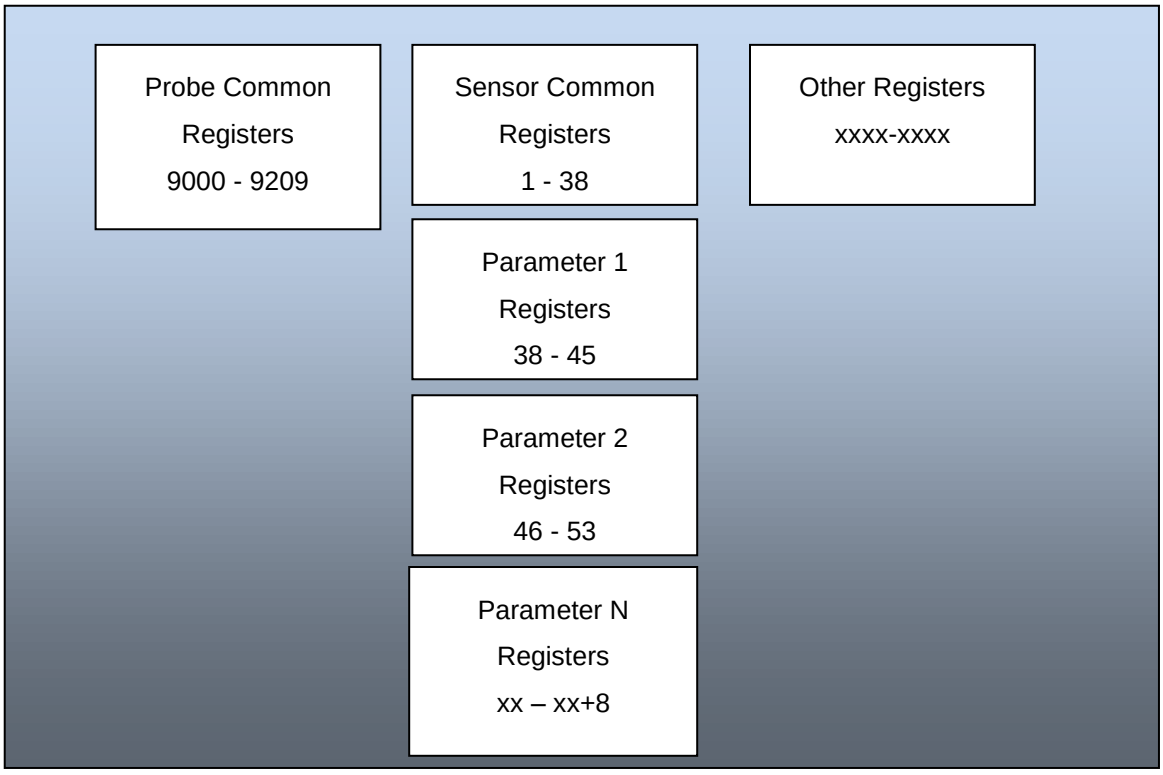
5.13 Extended Exception Codes

Code	Name	Description
0x80	Field Mismatch	Mismatch between register number, count and field size.
0x81	Write Only Register	Attempting to read a Write Only register.
0x82	Read Only Register	Attempting to write a Read Only register.
0x83	Access Level	Attempting to Read/Write a register with invalid access level.
0x84	Write Value	Attempting to write an illegal field value.
0x85	Command Sequence	Invalid device command register sequence.
0x86	File Sequence	Invalid file command register sequence.
0x87	File Command	Invalid file command.
0x88	File Number	Invalid file number.
0x89	File Size	Invalid file size.
0x8A	File Data	The file data block transferred to device is invalid.
0x8B	File Interval	Invalid file interval.
0x90	Gateway Error	Invalid probe command on controller gateway.
0x91	Sensor Sequence	Invalid sensor command register sequence.
0x92	Sensor Mode	Invalid change to sensor mode, or attempting to read/write with an invalid sensor mode set.
0x93	Sensor Config	Attempting a sensor operation on a sensor port that is not part of the current configuration.
0x94	Sensor Missing	Attempting a sensor operation on a sensor port with no sensor connected.
0x95	Sensor Invalid	Attempting a sensor operation on a sensor port that has a sensor that is not compatible with the sensor that was originally configured.

Code	Name	Description
0x96	Sensor Firmware	Attempting a sensor operation on a sensor with no application code.
0xA0	Data Log Register	Attempting to write a register that is read-only during logging.
0xA1	Data Log Memory	Data log memory is full.
0xA2	Data Log Directory	Data log directory is full.
0xA3	Data Log Edit	Log configuration cannot be edited.
0xA4	Data Log Sequence	Invalid data log command sequence.

5.14 Probe Register Map Layout

Each probe and sensor will follow a common layout pattern which provides consistency and improves code reuse between platforms. **All Instrument registers are Holding Registers.** The common probe register map layout is as follows:



All registers in this document are 1-based. This means the actual packets sent to the devices must have a data address 1 less than what the register number is in this document.

5.15 Probe Common Registers

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9000	1	R	ushort	Register Map Template Version (SIS Level)
9001	1	R/W	ushort	Device ID
9002-003	2	R/W	ulong	Device Serial Number
9004-006	3	R/W	time	Manufacture Date
9007	1	R	ushort	Firmware version * 100
9008	1	R	ushort	Boot Code version * 100
9009	1	R	ushort	Hardware version
9010	1	R	ushort	Max Data Logs
9011-012	2	R	ulong	Total Data Log Memory (bytes)
9013-014	2	R	ulong	Total Battery Ticks
9015-017	3	R/W	time	Last Battery Change
9018	1			Reserved
9019-050	32	R/W	string	Device Name
9051-082	32	R/W	string	Site Name
9083-086	4	R/W	double	Latitude Coordinate (degrees) (negative = east)
9087-090	4	R/W	double	Longitude Coordinate (degrees) (negative = south)
9091-094	4	R/W	double	Altitude Coordinate (meters)
9095-096	2			Reserved
9097-099	3	R/W	time	Current time (UTC)
9100-101	2	R	32 bits	Device Status
9102-103	2	R/W	ulong	Used Battery Ticks
9104-105	2	R	ulong	Used Data Log Memory (bytes)

5.16 Communication Control Registers

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9200	1	R/W	ushort	Device Address (1-247, default = 1)
9201	1	R/W	ushort	Serial Communication Configuration
9202	1	R/W	ushort	EOM timeout (1000-15000 ms, default = 1000)
9203	1	R/W	ushort	EOS timeout (5000-60000 ms, default = 5000)
9204	1	R	ushort	Max allowed baud rate id (0-7)
9205	1	R	ushort	Max Message/Response size (bytes)
9206-207	2	R/W	ulong	Good message counter
9208	1	R/W	ushort	Bad message counter
9209	1	R/W	ushort	Exception response counter

5.17 Probe Connection Registers

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9297	1	R1	ushort	Max Probe Connections (1-32)
9298-299	2	R1	32 bits	Probe Connection Status

5.18 Sensor Connection Registers

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9300	1	R1	ushort	Max Sensor Connections (1-32)
9301-302	2	R1	32 bits	Sensor Connection Status
Sensor Map Registers				
9303	1	R1	ushort	Connection 1 sensor id
9304	1	R1	16 bits	Connection 1 sensor status
9305	1	R1/W2	ushort	Connection 1 sensor command
9306	1	R1	ushort	Connection 1 sensor data register map version

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9307	1	R1	ushort	Connection 1 sensor data register offset
9308-462	5 x 31			Connections 2-32 patterned after connection 1
9463	1	R1/W3	ushort	Sensor Data Cache Timeout (0-60000 msec)

5.19 Current Loop Configuration Registers

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9501	1	R1/W3	ushort	Current loop sensor number
9502	1	R1/W3	ushort	Current loop parameter number
9503-504	2	R1/W3	float	Current loop 20 mA setpoint (I_{20})
9505-506	2	R1/W3	float	Current loop 4 mA setpoint (I_4)
9507	1	R1/W3	ushort	Current loop enable (0 = off, 1 = on)

5.20 Logged Record Registers

Register XXXX	Size (registers	Mode (R/W)	Data Type	Description
9600-601	2	R1	ulong	Number of logged records
9602-603	2	R1/W3	ulong	Requested log record number
9604-606	3	R1	time	Time stamp for record
9607-608	2	R1	float	Parameter 1 measured value
9609	1	R1	ushort	Parameter 1 data quality id
96010-696	29 x 3	R1	---	Parameters 2 – 30 (30 is max allowed #log parameters)

5.21 Register Map Template ID

This is a number that represents the version of the reserved register map supported by the device. This provides for the modification of the reserved register map at a future date and detection of this difference by the Master. This register is commonly referred to as the System Interface Specification Level or (SIS Level).

5.22 Device ID

Device ID for RDO PRO = 12

Device ID for RDO Titan = 19

5.23 Device Serial Number

This is a 6-digit serial number engraved on the device. Serial numbers for devices in this system will range from 000001 to 999999.

5.24 Manufacture Date

This is the date and time of manufacture in the Time format.

5.25 Firmware, Boot Code, Hardware Versions

The firmware and boot code versions will be the floating point version multiplied by 100 to create an integer. For example, version 1.32 will be stored as 132. The hardware version will be a non-scaled integer that represents the circuit board version. The hardware version will be calculated by the firmware based on parameters determined from the circuit board components.

5.26 Device Name

This is a general-purpose 32 character string representing a user-defined device name or identifier.

5.27 Site Name

This is a 32 character string that represents the location where the instrument is recording data.

5.28 Coordinates

These registers are used by the computer to store the coordinates of a device when taking measurements. The device expects coordinates in meters and degrees.

5.29 Current Time

This represents the time in the device in UTC.

5.30 Device Status

The device status register holds general status information. Each set bit represents a status value. There are a limited number of standardized predefined status values that all devices will support. These predefined status values are contained in the lower register. The upper register is reserved for device specific status values.

Bit	Category	Description
0	Alarm	Sensor high alarm
1	Warning	Sensor high warning
2	Warning	Sensor low warning
3	Alarm	Sensor low alarm
4	Warning	Sensor calibration warning
5	Alarm	Sensor malfunction
6-7	N/A	Reserved
8	Status	Power management disabled.
9	Status	Device off line
10	Alarm	Device hardware reset occurred
11	Alarm	Device malfunction
12	Status	No external power.
13	Warning	Low battery – battery capacity < 5%
14	Warning	Low memory – data log memory capacity < 5%
15	N/A	Reserved
16-31	N/A	Available for device-specific status

- Bits 0-7 of the device status register are reserved for sensor status. These bits are the logical OR of bits 0-7 of the sensor status register in each sensor connection.
- Bits 8-15 of the device status register are reserved for common device status. Any bit in this range that is not applicable to a device will be set to zero.
- Bits 16-31 of the device status register are available for device-specific status. Any bit in this range that is not utilized by a device will be set to zero.

5.31 Serial Communication Configuration

The 16 bits in this register are mapped to the communication parameters. The bits are mapped as follows:

Bits	Description
0	Modbus Transmission Mode 0 = RTU (default) 1=ASCII
1,2 & 3	Baud Rate ID 0 = 9600 (mandatory) 1 = 19200 (default) 2 = 38400 3 = 57600 4 = 115200 5 = 128000 6 = 230400 7 = 256000
4	Data Bits 0 = 7 data bits 1 = 8 data bits (default)
5,6	Parity Bits 0 = Even (default) 1 = Odd 2 = None
7	Stop Bits 0 = 1 Stop Bit (default) 1 = 2 Stop Bits
8-15	Unassigned

When the communication configuration register is changed, the Modbus response will be sent to the Master at the current configuration (mode, baud, parity, data bits…etc). After the response has been sent to the Master, the device will switch to the new settings.

The Master software must switch communications to the new settings after receiving a positive Modbus response to the write message. The Master software should confirm the new settings by reading back the device address and communication configuration register.

5.32 Baud Rates

The instrument supports 9600 through 57600 baud rates. A device will support all baud rates from 9600 up to and including the maximum baud rate as specified by the Max Baud Rate ID register.

- If the Master attempts to set the baud rate of a device to a non-supported value, the device will respond with a Modbus exception error code 3 (Illegal Data Value).
- Baud rates will be referenced in this document by the ID 0-7.

5.33 RTU Settings

Every device supporting serial communications will implement Modbus RTU. The device will at a minimum support the standard settings defined below.

- 1 Start Bit
- 8 Data Bits
- 1 Parity Bit
- 1 Stop Bit
- Even Parity

Note: If an attempt is made to write RTU mode with 7 data bits, the device will return an exception with error code 0x84 (Write Illegal Value). RTU communication does not support 7-data-bit settings.

5.34 ASCII Settings

When implementing Modbus ASCII, the device will at a minimum support the standard settings defined below.

- 1 Start Bit
- 7 Data Bits
- 1 Parity Bit
- 1 Stop Bit
- Even Parity

5.35 Max Message/Response Size

This register indicates to the Master the largest message or response the device can accept. This may vary based on the hardware configuration of the device.

5.36 Message Counters

The message counter registers are to provide diagnostic information for troubleshooting communication problems. The counters will not roll over and can only be cleared using the configuration software.

- There are 3 message counter registers allocated as follows:
- Good Message Counter – count of number of properly formatted messages received that are addressed to this device
- Bad Message Counter – count of number of improperly formatted messages received (i.e. bad CRC). Bad messages might not be associated with this device because it is impossible to determine if a bad message was addressed to the device or not.
- Exception Response Counter – count of the number of messages received that were rejected with a Modbus exception response.

5.37 Probe Connection Registers

5.37.1 Max Probe Connections

Each controller can have attached to it up to 32 probes. For the attached probes, the controller acts as a gateway, passing through messages to the connected probes. The controller maps unique addresses to each connected probe starting at the next higher Modbus address than the controller (if controller is at address 5, first probe is at address 6). Messages sent to the controller for a probe will be remapped to the protocol and address of the attached probe. Responses from a probe to the Modbus master are protocol and address converted before being sent back to the master.

5.37.2 Probe Connection Status

Each bit, when set, indicates that the controller is actively connected to a probe on the corresponding connection/port. The bit position 0-31 is used to determine the Modbus device address offset (from the controller) of the attached probe. For example, if bit 5 is set and the controller is at address 5, then the probe on the 6th port is at address 11.

5.37.3 Controller/Probe Addressing

Since controllers will map virtual device addresses to the connected probes, the controller's Modbus address range must be less than the max 247 by the number of allowed probes. For example, if a controller supports two probes, the maximum allowed address for the controller is 245 where address 246 is mapped to the first probe and address 247 is mapped to the second probe.

5.38 Sensor Connection Registers

5.38.1 Max Sensor Connections

Each probe or controller can present up to 32 sensor images to the outside world. These sensors may be real sensors plugged into physical ports, sensors on the internal PCB or it may be virtual sensors created from derived information. This register tells how many sensor images are being presented by the probe or controller at the time it is read.

5.38.2 Sensor Connection Status

Each bit when set indicates that the probe or controller is actively presenting a sensor image on the corresponding port. The bit position 0-31 is used to determine the sensor map register group to use to control the sensor or access its data block.

5.39 Sensor Map Registers

5.39.1 Sensor ID Registers

These registers duplicate the Sensor ID register provided in each sensor's Sensor Data Header Register block. If a sensor connection is open, the probe returns a zero. If an attempt is made to access a Sensor ID register that is not mapped to a sensor (one that exceeds the maximum number of sensors supported by the probe), the probe will return an exception response with error code 0x02 (illegal data address).

5.39.2 Sensor Status Register

These registers duplicate the sensor status register provided in each sensor's Sensor Data Header Block. If a sensor connection is open, the probe returns a zero. If an attempt is made to access a sensor status register that is not mapped to a sensor (one that exceeds the maximum number of sensors supported by the probe), the probe will return an exception response with error code 0x02 (illegal data address).

5.39.3 Sensor Command Register

These registers are used to send commands to a sensor. The behavior of the sensor command register and the commands available is different from SIS-1&2 and SIS-3. The sensor command register is primarily used to calibrate the sensor. Failure of the command is returned in the response exception.

If an attempt is made to access a sensor command register that is not mapped to a sensor (one that exceeds the maximum number of sensors supported by the probe), the probe will return an exception response with error code 0x02 (illegal data address).

ID	Name	Description
0xE000	Calibration Mode On	Put sensor in the calibration mode.
0xE001	Calibration Update	Commit new calibration to memory.
0xE002	Calibration Mode Off	Put sensor in the normal operating mode.
0xE003	Restore Cal Defaults	Restore factory default calibration for sensor only.
0xE004	Restore sensor defaults	Restore factory defaults for calibration, parameters and units, sentinel values, etc. for sensor only.

5.39.4 Sensor Data Register Map Version Registers

These registers specify the version of the sensor data register map pointed to by the Sensor Data Register Offset registers. If a sensor connection is open, the probe shall return a zero. If an attempt is made to access a sensor map version register that is not mapped to a sensor (one that exceeds the maximum number of sensors supported by the probe), the probe will return an exception response with error code 0x02 (illegal data address).

5.39.5 Sensor Data Register Map Offsets

These registers specify the register number of the first register in each sensor's Sensor Data Header Block. These registers assume the 4x (holding register) reference ID and therefore will not contain the reference ID as part of the value (i.e. 40001 will be stored as 1).

If an attempt is made to access a sensor status register that is not mapped to a sensor (one that exceeds the maximum number of sensors supported by the probe), the probe returns an exception response with error code 0x02 (illegal data address).

If a sensor connection is open (the sensor is not plugged in or is not part of the current configuration) the probe returns 0. This permits a valid block read of the mapped sensor status registers regardless of the current configuration of the probe and sensors.

If an attempt is made to access a sensor connection register that is not mapped to a sensor (one that exceeds the maximum number of sensors supported by the probe), the probe will return an exception response with error code 0x02 (illegal data address).

5.39.6 Sensor Data Cache Timeout

Sensors shall measure all of their parameters whenever a parameter value register is read. The parameters are recorded in a data cache and a cache timeout is started. If any subsequent parameter value from the sensor is read within the specified cache timeout, the device shall return the value recorded in the cache instead of making a new measurement. The default value of the cache timeout is device specific. If the cache timeout value is set to 0 milliseconds, each Modbus message to read one or more parameters will cause a new sensor reading to be taken. Logs running on a probe will also populate the data cache when a measurement is taken.

The end-of-session timeout supersedes the cache timeout – if an end of session timeout occurs, the cache for all sensors shall be cleared.

5.40 Current Loop Configuration

Attempting to access current loop configuration registers 9501 through 9507 will result in an exception response with exception code 0x02 (illegal data address).

5.41 Last Logged Record Registers

These registers are not supported by the device since data logging is not supported. Attempting to access these registers will result in an exception response with exception code 0x02 (illegal data address).