ROW MODBUS CONNECTIVITY

FOR ROW STARTING FROM FIRMWARE VERSION 6.9

Application Note for integrating ROW into third-party systems using Modbus

This document describes the functioning of Modbus serial protocol in ROW and provides information on connecting ROW to third-party systems and analysis equipment (such as SCADA systems, third-party data loggers, PLCs, etc.), retrieving signal and status information, as well as making minor changes to the parameters.

NB! This document assumes user to have familiarity with ROW, its general settings, wiring specifications, limitations and precautions. Please make sure to study the *ROW User Manual* before attempting any procedures described in this document. In case of any questions, please contact Laser Diagnostic Instrument.

Table of Contents

Introduction 3
Modbus overview 4
Connectivity and network topology 4
Registers and Functions 5
Register Map 8
Modbus exception codes 10
Appendix 1: Reading and Writing Examples 12
Appendix 2: ROW Serial Settings 16
Appendix 3: Rangefinder Support 16
Appendix 4: State and Error Flags 18

Introduction

ROW uses several different interfaces and protocols to communicate with LDI software or third-party equipment. The scope of this document is Modbus RTU protocol over RS-485 interface as applied to ROW.

Modbus RTU is an openly published serial protocol for industrial applications. Modbus RTU in ROW is implemented over RS-485 network and uses the same RS-485 line as the proprietary ROW protocol.

Detailed specification of Modbus, as well as its principles of operation, are out of scope of this document. For general information about Modbus please refer to the official specification at https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

For complete functionality and operational principles of ROW, as well as its electrical characteristics and limitations, please refer to the ***ROW User Manual***.

Modbus implementation in ROW supports following actions:

- Reading current signal and background levels
- Determining whether the device has issued an alarm
- Reading the device debug information (state and errors)
- Reading and writing limited set of parameters
    - Thresholds
    - Distance from ROW to water
    - Alarm delay

Important! Modbus implementation in ROW is intended for run-time day-to-day operations of ROW, primarily for monitoring. While changing certain parameters of ROW is possible using Modbus, any major configuration or troubleshooting should be accomplished using ROW Configurator software. Generally, making any change to the device parameters using ROW Configurator is a safer and more reliable method than writing parameters using Modbus, and should therefore be preferred where feasible.

Modbus overview

Modbus is a serial protocol widely used in industrial applications. It is supported by a large number of control and processing equipment. Generally, it may be considered a standard for digital communication with sensors and industrial automatics.

Modbus protocol has two transmission modes: RTU and ASCII. RTU uses binary codes for its characters and CRC error checking, whereas ASCII uses ASCII characters and LRC error checking. While Modbus ASCII is more human-readable due to being transmitted in plain characters, Modbus RTU requires significantly less bandwidth, is processed faster and uses more efficient error checking mechanism.

ROW implements Modbus RTU transmission mode only!

Modbus is a purely Client-Server protocol. It means that a client device (a controller, computer software, data logger) will have to poll the server device (sensor or actuator), to which the server device responds with either acknowledgement (and data, if any was requested), or an exception when the server cannot execute the client's request.

Originally, Modbus standard defines 21 function and four primary memory types:

- Coil
- Discrete Input
- Input Register
- Holding register

As often the case in recent equipment, ROW only requires and implements a fraction of that functionality. Please see the section *Registers and Functions* below.

Connectivity and network topology

Modbus is implemented in ROW using RS-485 interface. For detailed information on how to connect ROW to RS-485 network electrically, please refer to ***ROW User Manual*** and ***ROW Wiring Schematics.***

As in Modbus specifications, ROW devices are addressed by the master device individually by addresses. **ROW addresses can only be in range of 1 to 99 (0x01 to 0x63)**.

Since RS-485 bus has no natural flow control or collision detection, there can be only one master device on the network/bus. Since ROW proprietary protocol, used by ROW Manager and ROW Configurator, utilizes the same RS-485 interface as Modbus, they cannot be used at the same time.

Please make sure that only one master device is connecting to ROW at any given time. Do not use Modbus monitoring and ROW Manager/ROW Configurator simultaneously. Make sure to disengage monitoring when making changes to ROW parameters (either using Modbus or ROW Configurator).

Registers and Functions

While Modbus specification describes multiple registers and functions, most modern devices limit their Modbus functionality for simplicity of design and operation. This is the case with ROW.

ROW supports only one data type defined by Modbus: 16 bit Holding Register, and only 2 Modbus functions:

- **Function 3** (0x03): Read Multiple Holding Registers
- **Function 16** (0x10): Write Multiple Holding Registers

Function 3 is used to read data from ROW, while function 16 is used to write parameters. Both functions allow for writing multiple registers at the same time.

Registers in Modbus are 16 bit long. Since some values returned by ROW have 32-bit Float type, a single value may be stored in two consecutive registers. As traditional for Modbus devices, the smaller address features the most significant bit. For more detailed information on values and data types, see *Register Map* below.

Both functions have to pass starting register and the number of registers to read/write as 16-bit values. Any request and any response will be trailed by 2 bytes of CRC checksum.

Please note that all 16-bit and 32-bit values except for CRC are Big-endian (the most significant byte is in the earlier register with smaller address). CRC, on the other hand, is Little-endian.

Function 3

Function 3 reads values from one or more 16-bit holding registers. To read a 32-bit floating point value, two consecutive registers should be read at the same time.

Function 3 can be applied to all the registers supported by ROW (see *Register Map*). It is intended to read both ROW signal values and parameters.

Function 3 request from the master device to ROW consists of 8 bytes. The structure is as follows:

Byte	Field	Values (hex)	Values (decimal)
1	ROW Address	0x01 to 0x63	1 to 99
2	Function	0x03	03
3	Starting Register Address MSB	0x00	00
4	Starting Register Address LSB		
5	Register Count MSB	0x00	00
6	Register Count LSB		
7	CRC sum LSB		
8	CRC sum MSB		

Function 3 response from ROW to the master device consists of variable number of bytes. This packet is composed of 3 *header bytes*, data bytes (determined by number of registers requested, calculated as register count multiplied by 2, since each register is two bytes), and 2 bytes of CRC. Exact number of *data bytes* can be read from the Byte Count field (see below). The structure is as follows:

Byte	Field	Values (hex)	Values (decimal)
1	ROW Address	0x01 to 0x63	1 to 99
2	Function	0x03	03
3	Byte Count	Determined by number of registers requested	
4	Register Value 1 MSB		
5	Register Value 1 LSB		
…….			
3 + (N×2) - 1	Register Value N MSB		
3 + (N×2)	Register Value N LSB		
3 + (N×2) + 1	CRC sum LSB		
3 + (N×2) + 2	CRC sum MSB		

Function 16

Function 16 writes values to one or more 16-bit holding registers. To write a 32-bit floating point value, two consecutive registers should be written at the same time.

Function 16 can be applied only to parameter registers (0x0010 to 0x0016, see Register Map). It is intended to write certain parameters to ROW.

When using Function 16, please pay extra attention to formatting the data. There are safeguards on ROW end to prevent memory corruption, but to avoid any unexpected behavior, please make sure the values passed to ROW are of the correct type.

Function 16 request from master device to ROW consists of variable number of bytes. The packet is composed of 7 header bytes, data bytes (determined by number of registers passed, calculated as register count multiplied by 2), and 2 bytes of CRC. Exact number of data bytes should be written to the Byte Count field. The structure is as follows:

Byte	Field	Values (hex)	Values (decimal)
1	ROW Address	0x01 to 0x63	1 to 99
2	Function	0x10	16
3	Starting Register Address MSB	0x00	00
4	Starting Register Address LSB		
5	Register Count MSB	0x00	00
6	Register Count LSB		
7	Byte Count	Determined by register count	
8	Register Value 1 MSB		
9	Register Value 1 LSB		

……			
7 + (N×2) - 1	Register Value N MSB		
7 + (N×2)	Register Value N LSB		
7 + (N×2) + 1	CRC sum LSB		
7 + (N×2) + 2	CRC sum MSB		

Function 16 response from ROW to master device consists of 8 bytes. The structure is as follows:

Byte	Field	Value (hex)	Value (decimal)
1	ROW Address	0x01 to 0x63	1 to 99
2	Function	0x10	16
3	Starting Address MSB	0x00	00
4	Starting Address LSB		
5	Register Count MSB	0x00	00
6	Register Count LSB		
7	CRC sum LSB		
8	CRC sum MSB		

For the examples of data exchange between a Modbus master and ROW, please see the

Appendix 1: Reading and Writing Examples.

Register Map

Here is the map of all available registers for ROW

DATA REGISTERS				
ADDRESS	**FIELD**	**DIRECTION**	**COMMAND**	**DATA TYPE**
0x0000	Signal MSB	READ	03	32-bit float
0x0001	Signal LSB			
0x0002	Background MSB			32-bit float
0x0003	Background LSB			
0x0004	Simple State			16-bit unsigned int
0x0005	Data Counter			16-bit unsigned int
0x0006	Device State			16-bit word
0x0007	Device Errors			16-bit word
PARAMETER REGISTERS				
ADDRESS	**FIELD**	**DIRECTION**	**COMMAND**	**DATA TYPE**
0x0010	Threshold Low MSB	READ/WRITE	03, 16	
0x0011	Threshold Low LSB			
0x0012	Threshold High MSB			
0x0013	Threshold High LSB			
0x0014	Alarm Delay			16-bit unsigned int
0x0015	ROW Distance			16-bit int
0x0016	Rangefinder Distance			16-bit int

Data registers

These registers contain signal and background data acquired by ROW. They are read-only.

- Signal
  - A 32-bit float value (format 4321, same as other float values in ROW) containing average signal over the latest batch of measurements by ROW. Reflects the fluorescent signal intensity
- Background
  - A 32-bit float value containing average background over the latest batch of measurements by ROW. Reflects outside (parasitic) light intensity
- Simple State
  - A 16-bit unsigned integer value (21 format) containing a simplified device state. Currently it can be one of the following values:
    - 0: Online and OK. The device is functioning normally, and alarm has not been raised
    - 10: Alarm. There is an oil spill detected by the device
    - 100: Service Required. The LED light intensity has dropped to the point where the device is no longer operational and should be serviced by LDI.

- Data Counter
    - A 16-bit unsigned integer value (21 format) containing a data counter. It increments every time ROW reads and processes another batch of data and updates the signal, background and state. This value can go up to 65535, and then begins again from zero.
- Device State & Device Errors
    - A 16-bit bitfield (word) containing more detailed device state in form of flags. There are used for troubleshooting and are not required to be handled during normal operations. Please see *Appendix 4: State and Error Flags* for further information.

Parameter Registers

- Threshold Low
    - A 32-bit float value containing lower alarm threshold. Can assume values from 0 to 10 000.
- Threshold High
    - A 32-bit float value containing upper alarm threshold. Can assume values from 0 to 1 000 000.
- Alarm Delay
    - A 16-bit unsigned int value containing alarm delay time. Can assume values from 1 to 100.
- ROW Distance
    - A 16-bit int value containing ROW distance to water. Distance should be in **centimeters**. Can assume values from 31 to 2000 cm.
- Rangefinder Distance
    - A 16-bit int value containing Rangefinder (Senix) distance to water. Distance should be in **centimeters**. Can assume values from 31 to 1524cm, or zero to turn off the water level monitoring. Please refer to ***ROW & Senix Rangefinder Manual*** for setting this value.

Modbus exception codes

In case something goes wrong with the request (either on client or server side), following outcomes are possible:

- Request is not received by ROW.
    - This happens when ROW has not received the request at all due to communication line issue, or due to ROW being out of service
- Request is ignored by ROW
    - This happens when ROW has received only part of the request (due to communication line issue or collisions on the bus), when the address is incorrect (does not match ROW's own address or broadcast address) or when the request CRC is incorrect
- An exception code is returned
    - This happens when the request itself is valid, but there is a user input error (such as an unsupported function, wrong data address or illegal data value)

The exception response returned by ROW follows the Modbus standard and has the following packet structure:

Byte	Field	Values
1	Address	ROW address, same as in the request
2	Function + Error Marker	Function code from the request + error marker.
3	Error Code	See the list of error codes below
4	CRC sum LSB	
5	CRC sum MSB	

Byte 2 of the exception response is an error marker, which is generated by setting the highest bit of the function in the request to 1 (or adding 0x80 to the function code).

For example:

```
REQUEST:  01 04 00 00 00 01 31 CA

RESPONSE: 01 84 01 82 C0
```

The request contains the function 04. Function 04, while a valid Modbus function, is not supported by ROW. Therefore, ROW will return the Illegal Function exception. Byte 2 of the response will now be 0x84, which is 0x04 (original function) + 0x80 (error marker). It can also be visualized as binary 1000 0100.

Byte 3 contains an error code, which defines which error had place. ROW can return following error codes:

- **Error code 0x01 – Illegal Function**
    - Is returned when the request contains an unsupported function, or any other than 0x03 or 0x10.
- **Error Code 0x02 – Illegal Address**
    - Is returned when the request tries to access an address which either does not exist (is out of bounds of the register map) or is not supported by current function (trying to write into a register 0x0000 to 0x0007).
- **Error Code 0x03 – Illegal Data Value**
    - Is returned when the request tries to write a value into a parameter that cannot assume this value. For example, trying to set a threshold to a negative number will return this error. This error will also be returned if the byte count in the request does not correspond with the provided register count.

Appendix 1: Reading and Writing Examples

Here are a few examples of using Modbus to communicate with ROW.

For these examples, we are using ROW with an address 01.

Reading Signal value from ROW

In this example we are reading a single 32-bit float value (Signal), which is contained in two registers: 0x0000 and 0x0001.

```
REQUEST:  01 03 00 00 00 02 C4 0B

RESPONSE: 01 03 04 43 B4 BD 0F 9E C5
```

REQUEST	**01 03 00 00 00 02 C4 B8**		**8 bytes**
Byte	**Field**	**Value**	**Description**
1	ROW Address	0x01	Sending request to ROW with an address 1
2	Function	0x03	Function 03: read holding registers
3	Starting Register	0x00	Starting to read from register with an address 0, which holds the signal
4		0x00	
5	Register Count	0x00	Reading 2 registers: a single 32-bit value.
6		0x02	
7	Checksum	C4	Calculated checksum: 0x0BC4, in Little-Endian format.
8		0B	
RESPONSE	**01 03 04 43 B4 BD 0F 9E C5**		**9 bytes**
Byte	**Field**	**Value**	**Description**
1	ROW Address	0x01	Receiving reply from ROW 1. Should be the same as in request.
2	Function	0x03	Function should be the same as in the request
3	Byte Count	0x04	Received 4 data bytes (2 16-bit registers)
4	Register 0x0000	0x43	Received the first part of the 32-bit float value
5		0xB4	
6	Register 0x0001	0xBD	Received the second part of the 32-bit float value
7		0x0F	
8	Checksum	9E	Received Checksum 0xC59E. Take a note that unlike all other numbers, checksum is Little-endian
9		C5	

Registers 0x0000 and 0x0001 together contain a value of 0x43B4BD0F = 361.477.

The signal value of 361.477 has been received from ROW.

Reading Simple State value from ROW

In this example we are reading a single 16-bit integer value (Simple State), that is contained in the register 0x0004.

```
REQUEST:  01 03 00 04 00 01 C5 CB

RESPONSE: 01 03 02 00 0A 38 43
```

REQUEST	**01 03 00 04 00 01 C5 CB**		**8 bytes**
Byte	**Field**	**Value**	**Description**
1	ROW Address	0x01	Sending request to ROW with an address 1
2	Function	0x03	Function 03: read holding registers
3	Starting Register	0x00	Starting to read from register with an address 4, which holds the Simple State
4		0x04	
5	Register Count	0x00	Reading 1 registers: a single 16-bit value.
6		0x01	
7	Checksum	C4	Calculated checksum: 0x0BC4, in Little-Endian format.
8		0B	
RESPONSE	**01 03 02 00 0A 38 43**		**7 bytes**
Byte	**Field**	**Value**	**Description**
1	ROW Address	0x01	Receiving reply from ROW 1. Should be the same as in request.
2	Function	0x03	Function should be the same as in the request
3	Byte Count	0x02	Received 2 data bytes (a single 16-bit register)
4	Register 0x0004	0x00	Received the 16-bit value
5		0x0A	
6	Checksum	38	Received Checksum 0x4338.
7		43	

The register 0x0004 contains a value of 0x000A = 10. The Simple State 10 means there is an alarm.

Reading parameter values from ROW

In this example we are reading all parameter values that are available by Modbus: Threshold Low, Threshold High, Alarm Delay, ROW Distance, Rangefinder Distance, registers from 0x0010 to 0x0016.

```
REQUEST:  01 03 00 10 00 07 05 CD

RESPONSE: 01 03 0E 44 7A 00 00 47 43 50 00 00 03 00 64 00 00
9A B5
```

- **Request (8 bytes)**
  - Address: 0x01
  - Function: 0x03
  - Starting register: 0x0010
  - Register count: 0x0007
  - Checksum: 0xCD05
- **Response (19 bytes)**
  - Address: 0x01
  - Function: 0x03
  - Byte count: 0x0E (14 bytes, 7 registers: 2 *32-bit float* + 3 *16-bit int/uint*)
  - Threshold Low (32bit float): 0x447A 0000 (decimal 1 000)
  - Threshold High (32bit float): 0x4743 5000 (decimal 50 000)
  - Alarm Delay (16bit uint): 0x0003 (alarm delay 3 seconds)
  - ROW Distance: 0x0064 (decimal 100: ROW distance 100 cm)
  - Rangefinder Distance: 0x0000 (Rangefinder monitoring off)
  - Checksum: 0xB59A

NB! Not all master devices are capable of reading and processing mixed-type registers (like 32-bit float and 16-bit integer as in this example) in a single channel. For some devices, separating float and integer values into different channels and communication sessions might be necessary.

Writing thresholds ROW

In this example we set the ROW Lower Threshold to 1500, and Upper Threshold to 75 000.

```
REQUEST:  01 10 00 10 00 04 08 44 BB 80 00 47 92 7C 00 F3 14

RESPONSE: 01 10 00 10 00 04 C0 0F
```

REQUEST	**01 10 00 10 00 04 08 44 BB 80 00 47 92 7C 00 F3 14**		**17 bytes**
Byte	**Field**	**Value**	**Description**
1	ROW Address	0x01	Sending request to ROW with an address 1
2	Function	0x10	Function 16: write holding registers
3	Starting Register	0x00	Starting to write from register with an address 0x0016, which holds the Threshold Low
4		0x10	
5	Register Count	0x00	Writing four registers, or two 32-bit values: Threshold Low and Threshold High
6		0x04	
7	Byte Count	0x08	Writing 8 bytes, or 2 32-bit values
8	Register 0x0016	0x44	Threshold Low: 1500 decimal = 0x44BB 8000 float Two 16-bit registers
9		0xBB	
10	Register 0x0017	0x80	
11		0x00	
12	Register 0x0018	0x47	Threshold High: 75 000 decimal = 0x4792 7C00 float Two 16-bit registers
13		0x92	
14	Register 0x0019	0x7C	
15		0x00	
16	Checksum	0xF3	Calculated checksum: 0x14F3, in Little-Endian format.
17		0x14	
RESPONSE	**01 10 00 10 00 04 C0 0F**		**8 bytes**
Byte	**Field**	**Value**	**Description**
1	ROW Address	0x01	Receiving reply from ROW 1. Should be the same as in request.
2	Function	0x10	Function should be the same as in the request
3	Starting Register	0x00	Starting to write from register with an address 0x0016.
4		0x10	
5	Register Count	0x00	Writing four registers, or two 32-bit values.
6		0x04	
7	Checksum	0xC0	Received Checksum: 0x0FC0
8		0x0F	

Since ROW responded with a confirmation packet, the settings were successfully written and applied. In case of a wrong input, ROW will either ignore the packet (checksum mismatch, timeout), or return an exception code (unsupported function, illegal data address, illegal data value).

Appendix 2: ROW Serial Settings

Here are the serial settings used to connect to ROW. Those in the "Changeable" section can be changed using ***ROW Configurator*** only.

CHANGEABLE SETTINGS		
Setting	**Value range**	**Default value**
Address	1 to 99	1 as shipped 99 after parameter reset
Baud rate (bps)	4800 to 115200	57600
STATIC SETTINGS		
Setting	**Value**	
Data bits	8 bits	
Stop bits	1	
Parity bit	None	

Please refer to *ROW User Manual* for more general information concerning serial communications, wiring and precautions.

Appendix 3: Rangefinder Support

ROW supports Senix ToughSonic series rangefinders. These rangefinders communicate with ROW using Modbus.

In communication between ROW and rangefinder, ROW serves as a master (client), and the rangefinder as a slave (server). This may cause issues, especially in a network with larger number of ROWs, since communications from master controller may collide with rangefinder communications.

The principal points of communications between ROW and the rangefinder are as follows:

- There can be only one rangefinder on a single RS-485 bus. This rangefinder should have an address set to 101. This address is set for all Senix rangefinders shipped with ROW by LDI. A packet with any other address will not be interpreted by ROWs as a rangefinder packet, and therefore will not be processed as such.
- When Rangefinder Distance parameter is non-zero, ROW with address 1 will send a Modbus request to rangefinder to fetch data, unless such message has been already sent by some other source (ROW Manager, for instance) during this interval.
- When Rangefinder Distance parameter is non-zero, any ROW in the network will receive and read a rangefinder response, should such appear, and adjust their distance calculation accordingly.

This way, only one device in the network (either ROW or master) will request distance info from the rangefinder, but all ROWs in the network will receive the response from the rangefinder.

While the probability of a collision between master-ROW and ROW-rangefinder communication sessions is relatively low, it still persists, especially in the long run. Therefore, it might be reasonable to handle rangefinder polling from the Modbus master side, so that it can be synchronized with the rest of communication sessions on the bus.

To do this, master can send a Modbus command to the rangefinder at a regular interval (recommended interval: 5 seconds). Since there can be only one rangefinder in the network and the address of any rangefinder is fixed to 101, the command will be the same for any network:

```
0x65 0x03 0x02 0x08 0x00 0x01 0x0C 0x54
```

Byte	Value	Description
1	0x65	Address of the Senix rangefinder (101)
2	0x03	Function to read holding registers (03)
3	0x02	Starting Register (0x0208 Distance Field)
4	0x08	
5	0x00	Register Count (1 register)
6	0x01	
7	0x0C	CRC (0x540C)
8	0x54	

This command will request distance data from register 0x0208, where distance measurement is stored in Senix rangefinder.

For further information on how to acquire and analyze distance data received from the rangefinder, please refer to the ***ToughSonic Family Sensors – Installation & Operating Instructions*** by Senix.

Appendix 4: State and Error Flags

The Device State and Device Errors registers provide user with more detailed information concerning ROW state and any errors that occur in the device. These are mostly used for troubleshooting purposes and are typically not required to be handled during normal operations.

Bit positions here are presented starting from the earliest in the bit field. Bit positions in a big-endian number would be reversed.

```
Binary:   1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0

Bit:      1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
```

Device State		
Bit	**Field**	**Description**
1	State OK	Normally should be 1
2	Image OK	Firmware loaded correctly. Normally should be 1
3	Software reset	Device was reset internally, for example, by pressing a “Reset” button in ROW Configurator, or from a firmware update
4	Watchdog reset	The device was forced to reset itself due to an error. May be an indication of abnormal behavior.
5-10	Not in use	
11	Signal too high	Signal exceeds Threshold High
12	No Alarm Delay	Waiting to disengage the alarm
13	Alarm Delay	Waiting to engage the alarm
14	Reflection	An increase in background coincided with signal increase. Alarm will not be generated.
15	Signal in thresholds	Signal is within thresholds for an alarm generation
16	Alarm	Alarm has been activated
Device Errors		
Bit	**Field**	**Description**
1-4	Not in use	
5-8	Internal Error	Please reset the device. If the problem persists, please contact LDI
9-11	Not in use	
12	LED Failure	Possible LED failure. Please contact LDI
13,15	ADC Underflow	Something caused abnormally low signal. If problem persists, please contact LDI.
14,16	ADC Overflow	Something caused abnormally high signal. This may happen while the device is exposed to bright external light (for example, sunlight). If problem persists without external factors, please contact LDI.