Modbus Protocol Guide
TG

Senva Sensors
9290 SW Nimbus Ave
Beaverton, OR 97008

154-0033-0A

Rev.	Release Date	By	Description of Change	ECR
0A	3/16/2017	CKB	Initial Release	---

Table of Contents

Access Legend	Readable	R
	Writable	W
	Writable with 0 only (statistics reset, see System Readings)	W0
	Saved in non-volatile configuration memory	NV

Highlighting indicates core registers for quick reference.

See Also:

152-0271	*TG BACnet Installation Instructions*
152-0273	*TG Sensor Replacement Instructions*
154-0031	*TG Series User's Guide*
154-0032	*TG BACnet Protocol Guide*

Configuration

Congratulations on installing your new Senva RS485 TG series toxic gas sensor! The *Modbus Protocol Guide* assumes the first stage of installation is complete, with the TG connected to your local RS485 network and powered. A green status LED indicates the TG is powered and ready. If not, please refer to the separate *Installation Instructions* before continuing.

Each device ships with a default *Slave Address*. To identify this address, add "100" to the last two digits of the unique serial number printed on the label. When installing a TG on a dedicated Modbus network with no other slave devices, the global address 255 (0xFF) may also be used (see R123).

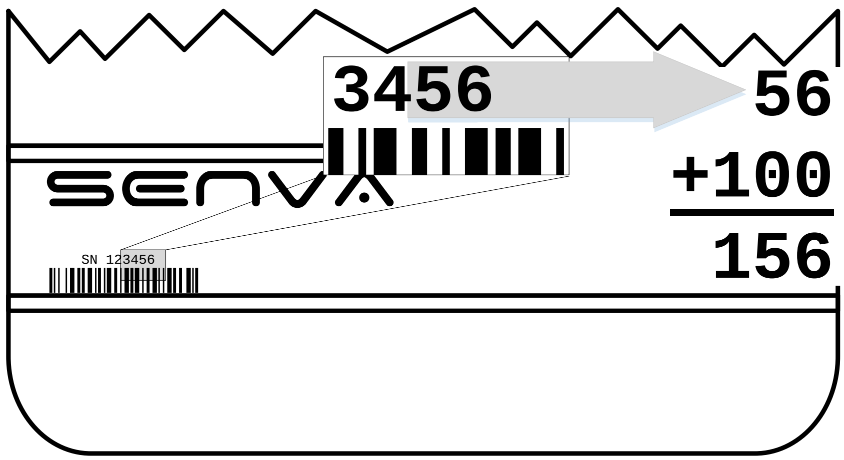

Figure 1: Example Default Address

WARNING: Before connecting a device to an existing network, ensure the default *Slave Address* will not conflict! If necessary, select an alternate address with the *User Menu* interface (see *User's Guide*). Regardless of whether the default *Slave Address*, the global address, or an alternate address is used, the device may be programmed with any supported address after an initial connection is established (see R123).

Leave the TG in the default factory mode for automatic network configuration:

- Automatic *Baud Rate* detection (see R114, R124): 9600 – 115200 baud
- Automatic *Parity* detection (see R116, R126): Even, Odd, No Parity
- Automatic *Data Bits* detection (see R117, R127): 7, 8 Bits
- Automatic *Protocol* detection (see R112, R122): Modbus RTU, ASCII

To begin automatic configuration, simply connect the RS485 terminals to an active Modbus network. An active Modbus network consists of a Modbus master device that regularly polls at least one Modbus slave (whether the slave replies or not).

IMPORTANT: If a Modbus network has devices communicating with multiple baud rates and/or formats, the automatic configuration result is unpredictable. Set the configuration manually with the *User Menu* interface (see *User's Guide*) before connecting the TG in such an environment.

Once connected, the device observes RS485 activity to learn baud rate, serial format, and protocol. Without activity, the TG cannot learn! The device will not interfere with existing network traffic during the observation phase. As configuration proceeds, the *RS485 Status* LED and the LCD display indicate progress with a combination of color and blinking activity patterns, and diagnostic codes (see *Installation Guide)*. Diagnostic mode conditions can help identify the current auto configuration step.

The *RS485 Status* LED turns green after at least one full frame successfully passes a CRC* integrity test. Assuming no conflicts, the master can then use Modbus functions to query or configure registers.

The TG stores any discovered automatic configuration result in non-volatile memory and reloads them whenever the device resets (e.g. after power loss). The automatic configuration can be cleared by using the *User's Guide* to manually set the RS485 configuration parameters. User-configured parameters will not be affected, but the TG must redetect any missing parameters before reestablishing communication.

For permanent installations, the protocol configuration parameters (see R122 – R128) may be set to lock the baud rate, format, and protocol. However, this will prevent the TG from adapting to future changes in the network environment.

The TG supports the following Modbus device functions:

- 0x08 *Diagnostics*
- 0x11 *Report Server ID*

* Cyclical Redundancy Check

Holding Registers

The TG supports the following Modbus functions:

- 0x03 *Read Holding Registers*
- 0x06 *Write Single Register*
- 0x10 *Write Multiple Registers*

In this document, Modbus addresses (beginning with "R") represent raw protocol addresses. Some Modbus conventions offset protocol addresses to form a register ID (e.g. 40001, Modicon notation). Refer to the relevant controller documentation to determine any required programming offset for each installation.

Constructed registers (see Data Types) span multiple Modbus address. The notation RXX/YY specifies a pair of aligned registers. The notation RXX-YY specifies a range of consecutive registers, inclusive.

Unless otherwise specified, changes to RS485 parameters are effective after the response (i.e. a client must maintain the original parameters for the remainder of the current transaction).

System Configuration

R102 Reset Status UINT16 R

Returns a reason determined at the time of the last reset:

1. *Configuration Reset* (see R190)
2. (reserved)
3. *Power Loss*
4. (reserved)
5. *Hardware Watchdog*

R103 Reset Count UINT16 R/NV

Returns a lifetime count of firmware resets for any reason. The device maintains this count in a protected section of non-volatile memory unaffected by *Configuration Reset* (see R190).

R104/05 Up Time UINT32 R

Returns the time since the last device reset, in seconds. To determine the cause of the last device reset read R102.

R111 Identify Device BOOL R/W

Sets the device interface into an easily identifiable state:

0. *Inactive*
1. *Active*

When set to *Active*, all of the status LEDs (see *Installation Guide*) light up, and the LCD screen will blink the current slave address of the device. The device will remain in identify mode until the value is reset to *Inactive* or the any *Button* is pressed.

This feature may be useful if several devices are connected on a single network and the association between discovered device IDs and each physical device is uncertain.

Default: *Inactive* (normal LEDs)

R112 Auto Protocol BOOL R/W/NV

Sets the state of automatic protocol detection:

0. *Inactive*
1. *Active*

When *Active*, the RS485 receiver initially allows frames of any supported protocol. On establishing confidence in a particular protocol (about 10 consecutive frames of the same type), this becomes the preferred *RS485 Protocol* (see R122).

Generally, having a preferred protocol disallows other protocols. This reduces uncertainty in the unlikely event that a particular frame or sequence could be interpreted as more than one protocol. However, should the protocol really change (e.g. by moving the device to a different network), the device will eventually lose confidence in the preferred protocol. After temporarily allowing all protocols, automatic protocol detection will establish a new preference. To avoid the delays associated with changing protocol, set the *RS485 Protocol* to some option that permanently allows multiple protocols (see R122, options 5-8).

When changing this value, the device keeps the current *RS485 Protocol* to avoid communication loss. Setting *Inactive* disables further automatic protocol changes and only allows the protocol(s) specifically set in *RS485 Protocol*.

Default: *Active*

R114 Auto Baud Rate BOOL R/W/NV

Sets the state of automatic baud rate detection:

0. *Inactive*
1. *Active*

When *Active*, the device may automatically change the *Baud Rate* (see R124) in response to RS485 communication errors. When changing this value, the device presents the actual *Baud Rate* to avoid communication loss.

Default: *Active*

R116 Auto Parity BOOL R/W/NV

Sets the state of automatic parity detection:

0. *Inactive*
1. *Active*

When *Active*, the device may automatically change the *Parity* (see R126) in response to RS485 communication errors. When changing this value, the device keeps the current *Parity* to avoid communication loss.

Default: *Active*

R117 Auto Data Bits BOOL R/W/NV

Sets the state of automatic data bits detection:

0. *Inactive*
1. *Active*

When *Active*, the device may automatically change the *Data Bits* (see R127) in response to RS485 communication errors. When changing this value, the device always keeps the current *Data Bits* to avoid communication loss.

Default: *Active*

R118 Auto Stop Bits — BOOL — R/W/NV

Sets the state of automatic stop bits:

0. *Inactive*
1. *Active*

Returns *Active* when *Stop Bits* is *Auto* (see R128).

Default: *Active*

R122 RS485 Protocol — UINT16 — R/W/NV

Sets the communication protocol(s):

1. *Auto*
2. *BACnet*
3. *Modbus RTU*
4. *Modbus ASCII*
5. *BACnet and Modbus RTU*
6. *BACnet and Modbus ASCII*
7. *Modbus RTU and ASCII*
8. *Any Protocol*

If *Auto Protocol* (see R112) is *Active*, returns the preferred automatic protocol (typically *Modbus RTU* or *Modbus ASCII*). Otherwise, returns the user-configured protocol option.

Setting any value other than *Auto* also sets *Auto Protocol* to *Inactive*. Setting *Auto* copies any previously set protocol option to the automatic protocol detector, and sets *Auto Protocol* to *Active*. Setting an option with multiple protocols (5 – 8) reduces the *Auto Protocol* re-detection delay.

Default: *Auto*

R123 Slave Address — UINT16 — R/W/NV

Sets the Modbus slave address, 1 – 254.

Assign unique addresses to each Modbus slave device on a network. The Modbus specification only allows slave addresses 1 – 247. Although the device supports the assignment of reserved addresses 248 – 254, undefined network behavior may result.

In the default configuration, returns the factory default slave address (see Configuration). Otherwise, returns the user-configured slave address. Before setting the address, ensure that the new address will not conflict with any other slave devices on the Modbus network.

In addition to the assigned slave address, the device will respond to Modbus commands addressed to the global address 255 (0xFF). Only use this address if the device is installed on a dedicated Modbus network with no other slave devices.

Default: *Varies*

R124/25 RS485 Baud Rate — UINT32 — R/W/NV

Sets the communication baud rate, 1200 – 460800.

When *Auto Baud Rate* is *Active* (see R114), returns the auto-detected baud rate (see Configuration). Otherwise, returns the user-configured baud rate.

Almost by definition, successfully reading baud rate implies a correct value, with no further action required. However, when transitioning a Modbus network to a new baud rate, it may be useful to remotely configure the new baud rate before transitioning the gateway/controller. Setting any user-configured baud rate also sets *Auto Baud Rate* to *Inactive*.

WARNING: The device provides no facility to revert the baud rate remotely. Once written, the device will lose communication until the client baud rate matches the new configuration.

Default: *Varies*

R126 RS485 Parity UINT16 R/W/NV

Sets the communication parity:

1. *Auto*
2. *No Parity*
3. *Odd Parity*
4. *Even Parity*

If *Auto Parity* is *Active* (see R116), returns the auto-detected parity (typically *Even Parity* for Modbus). Otherwise, returns the user-configured parity option.

Setting any value other than *Auto* sets *Auto Parity* to *Inactive*. Setting *Auto* sets *Auto Parity* to *Active*, but keeps the current parity option to avoid loss of communication.

Default: *Auto*

R127 RS485 Data Bits UINT16 R/W/NV

Sets the communication data bits:

1. *Auto*
2. *7 Bits*
3. *8 Bits*

If *Auto Data Bits* is *Active* (see R117), returns the automatically detected number of data bits (typically *8 Bits* for *Modbus RTU*, *7 Bits* for *Modbus ASCII*). Otherwise, returns the user-configured data bits option.

Setting any value other than *Auto* sets *Auto Data Bits* to *Inactive*. Setting *Auto* sets *Auto Data Bits* to *Active*, but keeps the current data bits option to avoid loss of communication.

Default: *Auto*

R128 RS485 Stop Bits UINT16 R/W/NV

Sets the communication stop bits:

1. *Auto*
2. *1 Bit*
3. *1.5 Bits*
4. *2 Bits*

Always returns the user-configured stop bits option. When set to *Auto*, Modbus dynamically provides the most compatible configuration:*1 Bit* for data receive and *2 Bits* for data transmit.

The *Auto Stop Bits* (see R118) value follows this value. Setting any value other than *Auto* sets *Auto Stop Bits* to *Inactive*. Setting *Auto* sets *Auto Stop Bits* to *Active*.

Default: *Auto*

R133 Temperature Units UINT16 R/W/NV

Sets the preferred units for temperature values:

1. *Degrees Fahrenheit* (°F)
2. *Degrees Celsius* (°C)

Saved statistical values automatically convert when the selected unit changes.

Default: *Degrees Fahrenheit*

R134 Smoothed Temperature Response Time UINT16 R/W/NV

Sets the step response time for smoothed temperature (see R312), in seconds.

Across all groups, the various *Smoothed* values track the instantaneous measurement after the application of a first-order exponential function. This low pass filter attenuates fast changes, such as the inrush current of a large industrial motor. *Smoothed* values may provide a stable baseline measurement but will always lag the instantaneous measurement (see Figure 2A).

Formally, response time sets the time required for a *Smoothed* value to complete 90% of the transition after an ideal step between two stable values (see Figure 2B).

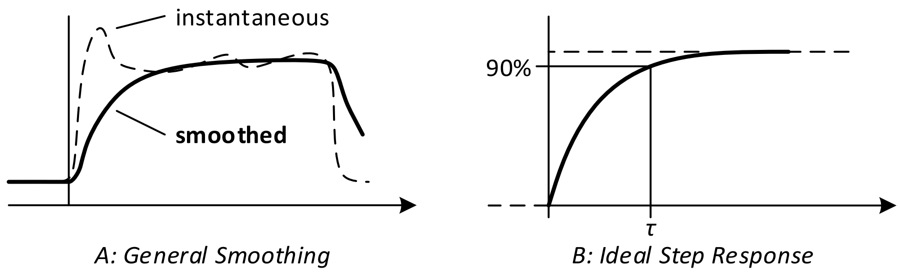

Figure 2: Smoothed Response Time

During periods of invalid measurement, *Smoothed* values return 0 (undefined). The resumption of valid measurements momentarily suppresses the smoothing function while the value stabilizes.

Default: 30 seconds

R136 Smoothed Gas Response Time UINT16 R/W/NV

Sets the step response time for the smoothed gas measurements (see R322 and R332), in seconds.

See R134 for a full explanation of smoothed response times.

Default: 90 seconds

R150 CO Warning Setpoint FLOAT R/W/NV

Sets the CO gas concentration required to enter the *Warning* state, in parts per million. (See R170 for state descriptions)

This object is only present in models with the *CO Sensor* feature.

Default: 25 PPM

R152 CO Alarm Setpoint FLOAT R/W/NV

Sets the CO gas concentration required to enter the *Alarm* state, in parts per million. (See R170 for state descriptions)

This object is only present in models with the *CO Sensor* feature.

Default: 100 PPM

R154 CO Hysteresis FLOAT R/W/NV

Sets how many PPM the CO gas level must fall below a setpoint before the system state transitions back from *Alarm* to *Warning*, or from *Warning* to *Alarm*, in parts per million.

This object is only present in models with the CO Sensor feature.

Default: 0 PPM

R156 CO Sensor Calibration FLOAT R/W/NV

Sets the gas sensitivity calibration of the presently installed CO Sensor, in nanoamperes per PPM.

This object is only present in models with the *CO Sensor* feature.

IMPORTANT: This value must be updated when a sensor is installed or replaced. (See *Sensor Replacement Instructions*)

Default: Varies

R158 CO Sensor Life INT16 R/NV

Returns the remaining life of the installed CO sensor, in days. Typical sensor life is at least 5 years, therefore the starting value for sensor life is 1825 days.

This object is only present in models with the CO Sensor feature.

R160 NO_2 Warning Setpoint FLOAT R/W/NV

Sets the NO_2 gas concentration required to enter a *Warning* state, in parts per million. (See R170 for state descriptions)

This object is only present in models with the *NO2 Sensor* feature.

Default: 1.0 PPM

R162 NO_2 Alarm Setpoint FLOAT R/W/NV

Sets the NO_2 gas concentration at which the TG will enter an *Alarm* state, in parts per million. (See R170 for state descriptions)

This object is only present in models with the *NO2 Sensor* feature.

Default: 3.0 PPM

R164 NO_2 Hysteresis FLOAT R/W/NV

Sets how many PPM the NO_2 gas level must fall below a setpoint before the system state transitions from *Alarm* to *Warning*, or from *Warning* to *Alarm*, in parts per million.

This object is only present in models with the NO_2 Sensor feature.

Default: 0.0 PPM

R166 NO_2 Sensor Calibration FLOAT R/W/NV

Sets the Nanoamp per PPM of the presently installed NO_2 Sensor, in nanoamperes per PPM.

This object is only present in models with the *NO2 Sensor* feature.

IMPORTANT: This value must be updated when a sensor is installed or replaced. (See *Sensor Replacement Instructions*)

Default: Varies

R168 NO_2 Sensor Life INT16 R/NV

Returns the remaining life of the installed NO_2 sensor, in days. Typical sensor life is at least 5 years, therefore the starting value for sensor life is 1825 days.

This object is only present in models with the *NO2 Sensor* feature.

R170 Current System State UINT16 R

Gets the current system state:

1. *Normal*
2. *Warning*
3. *Alarm*
4. *Extended Alarm*

The device determines the current "System State" of the device by using the user defined setpoints. For more information on 'Smoothed PPM' see R322 and R332. For more information on setpoints see R150, R152, R154, R160, R162, and R164. See (Figure 3)

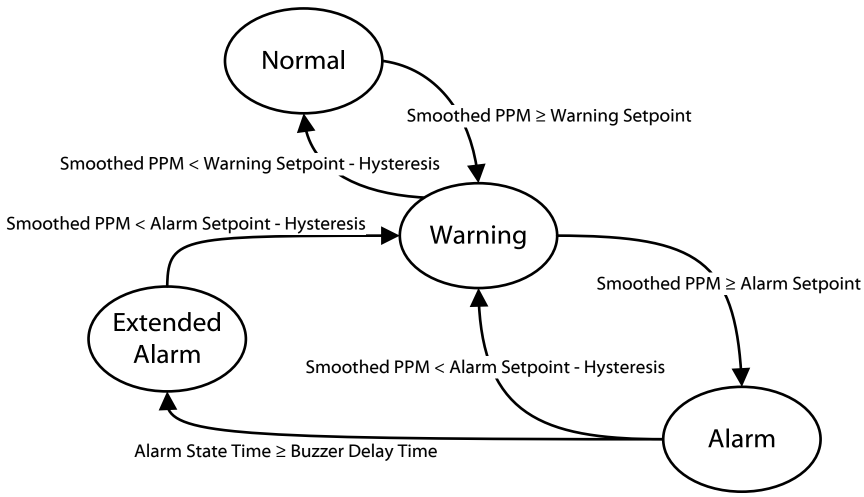

Figure 3: System State Diagram.

R190/91 System Configuration Reset UINT32 W

Always returns 0. Write 9699690 reset the TG to factory defaults.

WARNING: The entire non-volatile configuration will be permanently lost; previous configurations cannot be recovered. This includes the current RS485 serial format and *Slave Address*. However, the sensor calibration and lifetime counter will not be reset. (See R156, R158, R166, and R168.

After a configuration reset, the TG itself will reset and re-learn the current baud rate, serial format, and protocol (see Configuration).

R192/93 System Statistics Reset UINT32 W

Always reads 0. Write 4765089 to reset analog statistics.

R195 Auto Reset Statistics BOOL R/W/NV

Sets the reset mode for statistical measurements (*Minimum*, *Maximum*, *Average*). When *Active*, reading an individual statistical measurement also resets it, as if it had been followed by a write of 0 (see R314-18, R324-28, R334-38, and R384-88).

Auto reset mode may be useful in some low-overhead remote logging installations.

WARNING: When using *Auto Reset Statistics* as part of a periodic process, ensure there are no extra reads generated between logging intervals. Otherwise, the resulting records may reflect only a portion of the intended interval.

Default: *Inactive*

Binary Outputs

R210 Fan Relay BOOL R

Returns the present state of the Fan Relay.

1. *Inactive*
2. *Active*

In *System* mode, the fan relay will activate automatically when the system state is greater than or equal to the *Warning* state (See R170), unless the state of the relay is being overridden by the user (see R211).

R211 Fan Relay Override UINT16 R/W

Overrides the fan relay to be active, inactive, or allow the relay to react naturally to the system state (*System*).

1. *System (Default)*
2. *Inactive*
3. *Active*

R212/13 Fan Maximum Off Time UINT32 R/W/NV

Sets the maximum time the relay may be off before automatically activating the relay, in seconds (see R210). When written to "0", the *Maximum Off Time* is ignored.

The relay will remain active for the relay's *Minimum On Time* (see Figure 4). If the relay's *Minimum Off Time* is greater than the *Maximum Off Time* the relay will remain off for the entire *Minimum Off Time* prior to being forced on by the *Maximum Off Time*.

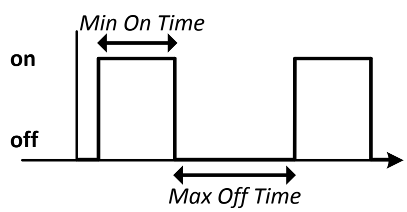

Figure 4: Max Off Time Diagram

Default: 0 seconds

R214/15 Fan Minimum On Time UINT32 R/W/NV

Sets the minimum time the relay must remain on before turning off, in seconds. This helps protect the fan from short cycling.

Default: 60 seconds

R216/17 Fan Minimum Off Time UINT32 R/W/NV

The minimum time the relay must remain off before turning on again, in seconds. This helps protect the fan from short cycling.

Default: 60 seconds

R218/19 Fan State Count UINT32 R/W0

The number of state transitions the relay has experienced since the device's last restart.

R220/21 Fan Total Active Time UINT32 R/W0

Returns the time the fan relay has been active since the device's last restart, in seconds.

R230 Alarm Buzzer BOOL R

Returns the present state of the Fan Relay.

1. *Inactive*
2. *Active*

In *System* mode, the alarm buzzer will activate automatically when the system state is equal to the *Extended Alarm* state, and deactivate when the system state falls below the *Alarm* state (see R170).

R231 Alarm Buzzer Override UINT16 R/W

Overrides the alarm buzzer to be active, inactive, or allow the relay to react naturally to the system state (*System*).

1. *System (Default)*
2. *Inactive*
3. *Active*

R232/33 Buzzer Delay UINT32 R/W/NV

Sets the time the audible buzzer alarm (See R230) waits before activating after the system has entered the *Alarm* state, in seconds. Once activated the buzzer will remain active until the gas concentration has fallen below the alarm setpoint (See R152, R162), or the buzzer is suppressed (See *User's Guide*) by holding down all 3 buttons on the device for 1 second.

Default: 1800 seconds (30 minutes)

R234/35 Buzzer Minimum On Time UINT32 R/W/NV

Sets the minimum time the buzzer must be on before turning off, in seconds.

Default: 0 seconds

R236/37 Buzzer Minimum Off Time UINT32 R/W/NV

Sets the minimum time the buzzer must be off before turning on again, in seconds.

Default: 0 seconds

R238/39 Buzzer State Count UINT32 R/W0

Returns the number of state transitions the buzzer has experienced since the device's last restart.

R240/41 Buzzer Total Active Time UINT32 R/W0

Returns the total time the buzzer has been *Active* since the device's last restart, in seconds.

Miscellaneous

R276-91 Location String ASCII R/W/NV

Null terminated ASCII string to be appended to the Server ID, 0 – 31 characters. Consecutive characters are appended to the fixed portion of the Server ID until the first null (0) is found.

Although registers after the first null won't print, they may still be used for arbitrary storage.

Default: "<location>"

R292-99 User Data UINT16 R/W/NV

Arbitrary data storage, 0 – 65535.

Any value written may be read back later.

Default: 0

System Readings

For statistical purposes, system readings are organized into groups. By convention, the primary value leading each group provides the most accurate, up-to-date reading possible. Secondary values within a group provide statistical measurements updated over time that may support some simple logging with low overhead for setup and bandwidth. Secondary values:

- **Smoothed:** Returns the value after applying a first-order exponential filter (see R134 or R136).
- **Minimum:** Returns the single lowest valid reading taken since last reset.
- **Maximum:** Returns the single highest valid reading taken since last reset.
- **Average:** Returns the average of all valid readings taken since last reset.

Once recorded, there are a few methods to reset statistics records (*Minimum*, *Maximum*, *Average*):

- Manually for a single value, by writing 0 to the register (**W0** in the Access Legend).
- Manually for multiple values, by writing one of the *Statistics Reset* keys (see R192).
- Automatically for single values, by configuring *Auto Reset Statistics* (see R195).

R310/11 Temperature FLOAT R/W0

Returns the approximate internal air temperature of the installed device. The default units are °F (see R133). The TG uses this temperature to compensate gas sensor measurements.

WARNING: The internal air temperature may vary from outside air due to self-heating of the device.

If the temperature is beyond the product's rated specifications, the LCD will indicate the corresponding *Temperature Limit* warning condition (see *User's Guide, Idle Condition Codes*).

The default smoothed value response time is 30 seconds (see R134).

R312/13 Instantaneous • R314/15 Minimum • R316/17 Maximum • R318/19 Average

R320/21 CO Gas Concentration FLOAT R/W0

Returns the measured CO (carbon monoxide) gas concentration in parts per million.

The default smoothed value response time is 90 seconds (see R136).

R322 Smoothed • R324 Minimum • R326 Maximum • R328 Average

R330/31 NO_2 Gas Concentration FLOAT R/W0

Returns the measured NO_2 (nitrogen dioxide) gas concentration in parts per million.

The default smoothed value response time is 90 seconds (see R136).

R332 Smoothed ● R334 Minimum ● R336 Maximum ● R338 Average

R380/81 Power Supply Voltage FLOAT R/W0

Returns the approximate working voltage provided to the power supply input. The voltage reported is an internal voltage after rectification and input protection (typically 1.0 – 2.0 V less than the external supply).

If the supply voltage drops below 10.0 V, the device anticipates power loss and saves the current configuration to non-volatile memory. However, if the power loss is only partial, the device may continue to operate at reduced supply voltages. The LCD will indicate the corresponding *Low Supply Voltage* warning condition (see the *User's Guide, Idle Conditions Codes*).

R382/83 Instantaneous ● R384/85 Minimum ● R386/87 Maximum ● R388/89 Average

Functions

The TG supports the following functions of the *Modbus Application Protocol Specification*, v1.1b3. Examples are intended to be representative; refer to the full Modbus standard for questions or clarification.

Notes:

- The device address 100 (0x64) is arbitrarily selected.
- Unless otherwise specified, examples show Modbus RTU encoding only.
- Refer to the Modbus standard for CRC/LRC calculation procedures.

Data Types

Natively, Modbus holding register functions only support the UINT16 type (2 bytes). The TG constructs additional types from two or more consecutive registers. Client interface software must support the same construction for proper communication:

	# of Registers	Range (hexadecimal)
UINT16	1	0 – 65535 (0xFFFF), unless otherwise noted
BOOL	1	0 – 1
UINT32	2	0 – 4294967295 (0xFFFFFFFF), unless otherwise noted
FLOAT	2	$\pm 3.402823 \times 10^{38}$ (IEEE-754), unless otherwise noted
ASCII	–	Variable length, NULL terminated

UINT32 and FLOAT data always occupies two registers (4 bytes) full network byte order (MSB first). Read and write operations must address both registers, excepting *Write Single Register* (0x06). FLOAT values are treated as a raw sequence of 4 bytes; refer to the IEEE-754 standard to interpret FLOAT contents.

The following examples show UINT32 values encoded in a Modbus PDU beginning at byte [n], register [r]:

Value	Decimal	[n]	[n+1]	[n+2]	[n+3]
0xAABBCCDD	2864434397	0xAA	0xBB	0xCC	0xDD
0x01234567	19088743	0x12	0x34	0x56	0x78
0x00010000	65536	0x00	0x01	0x00	0x00
REGISTER		**[r]**		**[r+1]**	

ASCII data occupies a range of registers, with each register encoding a pair of consecutive characters. Refer to Appendix B for ASCII character encodings. The MSB of each register encodes the first ASCII character of the pair. ASCII values support random access read and write operations on any partial register range. The very last character (the LSB of the last register) must always be ASCII NULL (0). ASCII NULL may also be placed earlier in the value to shorten the printed string. The following example shows the ASCII encoding of "TESTING" in a Modbus PDU beginning at byte [n] (register [r]):

OFFSET	[n]	[n+1]	[n+2]	[n+3]	[n+4]	[n+5]	[n+6]	[n+7]
ASCII	T	E	S	T	I	N	G	NULL
HEX	0x54	0x45	0x53	0x54	0x49	0x4E	0x47	0x00
REGISTER	**[r]**		**[r+1]**		**[r+2]**		**[r+3]**	

0x03 Read Holding Registers

Returns one or more registers in a contiguous block:

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x03
[2]	Starting Address	2	$\boldsymbol{A}$ = 0 to 65535 (0xFFFF)
[3]	Register Count	2	$\boldsymbol{N}$ = 1 to 125 registers
[4]	CRC	2	

Successful reads return the contents of the requested registers:

	Response	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x03
[2]	Byte Count	1	2 * $\boldsymbol{N}$
[3]	Register Data	2 * $\boldsymbol{N}$	
[4]	CRC	2	

Failed reads return an exception code:

Illegal Data	Improperly formed request or Register Count out of range.
Illegal Address	The combination of Starting Address + Register Count exceeds 65536.
Server Failure	Alignment fault: UINT32 and FLOAT reads must request both registers.

This function supports reads from the complete Modbus address space (registers 0 – 65535) without error, including undefined addresses. This facilitates combined reads spanning multiple non-adjacent registers, which may be more efficient in some circumstances. Always discard data read from undefined addresses.

Example 1: Read the current RS485 configuration (see R122 – R128), 6 values total. However, because *Baud Rate* spans two registers, the *Register Count* must be 7 to get all the data.

Request =	**0x**	64	03	00 7A	00 07	2C 24
		[0]	[1]	[2]	[3]	[4]

Response =	**0x**	64	03	0E	00 03	00 64	00 01 2C 00	00 04	00 03	00 00	34 B5
		[0]	[1]	[2]	[3a]	[3b]	[3c]	[3d]	[3e]	[3f]	[4]

[3a] *RS485 Protocol*	= 0x0003	= 3 (*Modbus RTU*)
[3b] *Slave Address*	= 0x0064	= 100
[3c] *Baud Rate*	= 0x00012C00	= 76800
[3d] *RS485 Parity*	= 0x0004	= 4 (*Even Parity*),
[3e] *RS485 Data Bits*	= 0x0003	= 3 (*8 Bits*),
[3f] *RS485 Stop Bits*	= 0x0000	= 0 (*Auto*)

0x06 Write Single Register

Writes a value to a single register:

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x06
[2]	Register Address	2	$A = 0$ to 65535 (0xFFFF)
[3]	Register Value	2	$X = 0$ to 65535 (0xFFFF)
[4]	CRC	2	

Successful writes echo the original request:

	Response	Size	
[0]	Device Address	1	
[1]	Function Code	1	Always 0x06
[2]	Register Address	2	A
[3]	Register Value	2	X
[4]	CRC	2	

Failed writes return an exception code:

Illegal Data	Improperly formed request.
Illegal Address	The requested address A is undefined.
Server Failure	New value X rejected (e.g. out of range).

For convenience, this function supports direct writes to the first register (low address) of UINT32 and FLOAT register pairs. Values are limited to the UINT16 range, and the device expands them internally before storing.

ASCII registers support arbitrary writes to any individual register within the string.

Example 1: Enable *Identify Device* (see R111).

Request = Response = **0x**	**64**	**06**	**00 6F**	**00 01**	**71 E2**
	[0]	[1]	[2]	[3]	[4]

Example 2: Set *Smoothed Gas Response Time* to 30 seconds (see R136).

Request = Response = **0x**	**64**	**06**	**00 88**	**00 1E**	**80 1D**
	[0]	[1]	[2]	[3]	[4]

Example 3: Set *Buzzer Delay* to 60 seconds (see R232; note UINT32 expansion).

Request = Response = **0x**	**64**	**06**	**00 E8**	**00 3C**	**00 1A**
	[0]	[1]	[2]	[3]	[4]

Example 4: Set the *CO Warning Setpoint* to 50.0 PPM (see R150; note FLOAT expansion).

Request = Response = **0x**	**64**	**06**	**00 96**	**00 32**	**E1 C6**
	[0]	[1]	[2]	[3]	[4]

0x08 Diagnostics

Performs miscellaneous device management functions. The Modbus protocol specifications defines many diagnostic sub-functions, but the device only supports the following sub-functions.

0x00 | Return Query Data

Returns response bytes equal to the request (echo):

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x08
[2]	Sub-Function Code	2	Always 0x0000
[3]	Request Data	–	Any data
[4]	CRC	2	

Successful commands echo the original request. This sub-function supports any length of request data up to and including 250 bytes.

Example: Return 4 bytes.

```
Request = Response = 0x 64  08  00 00  04 D2 00 00  10 0C
                       [0] [1]  [2]       [3]        [4]
```

0x01 | Restart Communications

Reinitializes the RS485 transceiver interface and clears all the communications event counters.

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x08
[2]	Sub-Function Code	2	Always 0x0001
[3]	Request Data	2	Always 0x0000
[4]	CRC	2	

Failed commands return an exception code:

Illegal Data Invalid Request Data.

Successful commands echo the original request, unless the device is in *Listen Only Mode* (see sub-function 0x04). In *Listen Only Mode*, the device suppresses the response. After the RS485 transceiver reinitializes, the *RS485 Status* LED indicates the *Waiting For Activity* condition (see *Installation Instructions*).

Example:

```
Request = Response = 0x 64  08  00 01  00 00  B8 3E
                       [0] [1]  [2]    [3]    [4]
```

0x03 | Change ASCII Input Delimiter

Sets a new end-of-message delimiter for future messages (replacing the default LF delimiter). This delimiter only applies to the Modbus ASCII protocol.

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x08
[2]	Sub-Function Code	2	Always 0x0003
[3]	Request Data	1	Encodes the new delimiter (0 – 127)
[4]	Padding	1	Always 0x00
[5]	LRC	2	

Failed commands return an exception code:

Illegal Data Invalid Request Data.

The command may fail if the padding byte is non-zero, or if the new delimiter is out of range. (Because the Modbus ASCII only specifies 7 data bits, delimiter characters greater than 127 cannot be transmitted.) If a command fails, the previously configured delimiter remains unchanged.

Successful commands echo the original request, including the original delimiter.

IMPORTANT: Switch to the new delimiter only at the beginning of the *next* Modbus request.

Example 1: Switch the delimiter to 0x00 (NULL). Encoding is Modbus ASCII.

Request = Response = **0x**	3A	36 34	30 38	30 33	30 30	30 30	39 31	0D	0A
	:	[0]	[1]	[2]	[3]	[4]	[5]	CR	LF

Example 2: Switch the delimiter back 0x0A (LF, the Modbus default).

Request = Response = **0x**	64	08	00 01	00 00	B8 3E
	[0]	[1]	[2]	[3]	[4]

0x04 | Force Listen Only Mode

Disables all Modbus protocol functions except for *Restart Communications* (see sub-function 0x01). This mode isolates the device from other devices on the network, allowing those devices to continue communicating without interruption. In this mode, the device monitors Modbus commands addressed to itself (including broadcast commands) but takes no actions and sends no response.

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x08
[2]	Sub-Function Code	2	Always 0x0004
[3]	Request Data	2	Always 0x0000
[4]	CRC	2	

Failed commands return an exception code:

Illegal Data Request data does not equal 0x0000.

Successful commands return no response.

Example:

Request = **0x**	64	08	00 04	00 00	A8 3F
	[0]	[1]	[2]	[3]	[4]

0x0A | Clear Counters and Diagnostic Register

Resets the diagnostic counters (see sub-functions 0x0B – 0x12).

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x08
[2]	Sub-Function Code	2	Always 0x000A
[3]	Request Data	2	Always 0x0000
[4]	CRC	2	

Failed commands return an exception code:

Illegal Data Request data does not equal 0x0000.

Successful commands echo the original request. NOTE: The diagnostic register is not supported.

Example:

Request = Response = **0x**	64	08	00 0A	00 00	08 3C
	[0]	[1]	[2]	[3]	[4]

0x0B – 0x12 | Return Diagnostic Counters

Returns one of the serial port diagnostic counters. These counters may be useful for performance and error management. Diagnostic counters reset to 0 when the device itself resets, or manually (see sub-function 0x0A).

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x08
[2]	Sub-Function Code	2	0x000B – 0x0012 (described below)
[3]	Request Data	2	Always 0x0000
[4]	CRC	2	

Failed commands return an exception code:

Illegal Data Request data does not equal 0x0000.

Successful commands return the current counter value in the Data field. The Modbus specification defines the following counters:

0x0B | Bus Message Count	Count of messages received, not including errors.
0x0C | Bus Communication Error Count	Count of character overrun, parity, and CRC errors.
0x0D | Bus Exception Error Count	Count of exception responses returned.
0x0E | Server Message Count	Count of messages addressed to and processed by the remote device, including broadcast messages.
0x0F | Server No Response Count	Count of messages addressed to the remote device for which it returned no response (e.g. broadcast messages).
0x10 | Server NAK Count	Count of Negative Acknowledge exception responses. Always returns 0.
0x11 | Server Busy Count	Count of Server Device Busy exception responses. Always returns 0.
0x12 | Bus Character Overrun Count	Count of messages that the device could not handle due to a character overrun (occurs when characters arrive at the port faster than they can be processed).

Example 1: Read the *Bus Message Count*.

Request = **0x** **64 08 00 0B 00 00 98 3C**
[0] [1] [2] [3] [4]

Response = **0x** **64 08 00 0B 01 A0 99 D4**
[0] [1] [2] [3] [4]

[3] *Bus Message Count* = 0x01A0 = 416

0x10 Write Multiple Registers

Writes one or more registers in a contiguous block:

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x10
[2]	Starting Address	2	A = 0 to 65535 (0xFFFF)
[3]	Write Count	2	N = 1 to 123 registers
[4]	Byte Count	1	Always 2 * N
[5]	Write Registers	2 * N	
[6]	CRC	2	

Successful writes echo the *Starting Address* and *Write Count*:

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x10
[2]	Starting Address	2	A
[3]	Write Count	2	N
[4]	CRC	2	

Failed writes return an exception code:

Illegal Data	Improperly formed request or Register Count out of range.
Illegal Address	The combination of Starting Address + Register Count exceeds 65536.
Server Failure	Write value rejected (e.g. out of range), or Write to a read-only register, or Write to an undefined register, or Alignment error: writes to UINT32 and FLOAT must cover both registers.

A *Server Failure* exception indicates that the device rejected a write to one or more registers. Unless exactly one value was written, it is not possible to determine which portion of the request caused the exception. The device attempts to apply each register in the request, regardless of failure result.

Example 1: Reset the System Statistics (see R192, key = 4765089 = 0x0048B5A1):

Request = **0x**	64	10	00 C0	00 02	04	00 48 B5 A1	27 0C
	[0]	[1]	[2]	[3]	[4]	[5]	[6]

Response = **0x**	64	10	00 C0	00 02	48 01
	[0]	[1]	[2]	[3]	[4]

Example 2: Change the location string to "Garage 1A.1" (see R276):

Request = **0x**	64	10	01 14	00 06	0C	47 61 72 61 67 65 20 31 41 2E 31 00	67 3F
	[0]	[1]	[2]	[3]	[4]	[5]	[6]

Response = **0x**	64	10	01 14	00 06	08 06
	[0]	[1]	[2]	[3]	[4]

0x11 Report Server ID

Returns device information in ASCII string format:

	Request	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x11
[2]	CRC	2	

The request always returns a valid response:

	Response	Size	Notes
[0]	Device Address	1	
[1]	Function Code	1	Always 0x11
[2]	Byte Count	1	Total length of fields 3 – 5
[3]	Server ID	1	Always 0x01
[4]	Run Indicator	1	Always 0xFF (run mode)
[5]	Additional Data	–	Varies, see below
[6]	CRC	2	

The *Additional Data* field returns an ASCII string with several concatenated values:

[5a]	Vendor Name	"Senva Sensors"
[5b]	Model Name	"TGW-BCN"
[5c]	Serial Number	varies
[5d]	Firmware Version	varies
[5e]	Location String	varies (see R276)

Example: Read the server ID:

```
Request = 0x 64  11  EB 7C
             [0] [1]  [2]

Response = 0x 64  11  30  01  FF  53 65 6E 76 61 20 53 65 6E 73 6F 72 …
                                … 73 20 54 47 57 2D 42 43 4E 20 33 …
                                … 31 30 30 35 32 20 31 2E 31 2E 30 …
                                … 20 47 61 72 61 67 65 20 31 41 2E 31  0C F9
              [0] [1] [2] [3] [4]               [5]                    [6]
```

[5] *Additional Data* = "Senva Sensors TGW-BCN 310145 1.1.0 Garage 1A.1"

Appendix A: Hex Conversions

HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	LATIN-1	HEX	DEC	LATIN-1
0x00	0	NULL	0x40	64	@	0x80	128	€	0xC0	192	À
0x01	1		0x41	65	A	0x81	129		0xC1	193	Á
0x02	2		0x42	66	B	0x82	130	‚	0xC2	194	Â
0x03	3		0x43	67	C	0x83	131	ƒ	0xC3	195	Ã
0x04	4		0x44	68	D	0x84	132	„	0xC4	196	Ä
0x05	5		0x45	69	E	0x85	133	…	0xC5	197	Å
0x06	6		0x46	70	F	0x86	134	†	0xC6	198	Æ
0x07	7		0x47	71	G	0x87	135	‡	0xC7	199	Ç
0x08	8		0x48	72	H	0x88	136	ˆ	0xC8	200	È
0x09	9		0x49	73	I	0x89	137	‰	0xC9	201	É
0x0A	10		0x4A	74	J	0x8A	138	Š	0xCA	202	Ê
0x0B	11		0x4B	75	K	0x8B	139	‹	0xCB	203	Ë
0x0C	12		0x4C	76	L	0x8C	140	Œ	0xCC	204	Ì
0x0D	13		0x4D	77	M	0x8D	141		0xCD	205	Í
0x0E	14		0x4E	78	N	0x8E	142	Ž	0xCE	206	Î
0x0F	15		0x4F	79	O	0x8F	143		0xCF	207	Ï
0x10	16		0x50	80	P	0x90	144		0xD0	208	Ð
0x11	17		0x51	81	Q	0x91	145	‘	0xD1	209	Ñ
0x12	18		0x52	82	R	0x92	146	’	0xD2	210	Ò
0x13	19		0x53	83	S	0x93	147	“	0xD3	211	Ó
0x14	20		0x54	84	T	0x94	148	”	0xD4	212	Ô
0x15	21		0x55	85	U	0x95	149	•	0xD5	213	Õ
0x16	22		0x56	86	V	0x96	150	–	0xD6	214	Ö
0x17	23		0x57	87	W	0x97	151	—	0xD7	215	×
0x18	24		0x58	88	X	0x98	152	˜	0xD8	216	Ø
0x19	25		0x59	89	Y	0x99	153	™	0xD9	217	Ù
0x1A	26		0x5A	90	Z	0x9A	154	š	0xDA	218	Ú
0x1B	27		0x5B	91	[	0x9B	155	›	0xDB	219	Û
0x1C	28		0x5C	92	\	0x9C	156	œ	0xDC	220	Ü
0x1D	29		0x5D	93	]	0x9D	157		0xDD	221	Ý
0x1E	30		0x5E	94	^	0x9E	158	ž	0xDE	222	Þ
0x1F	31		0x5F	95	_	0x9F	159	Ÿ	0xDF	223	ß
0x20	32		0x60	96	`	0xA0	160		0xE0	224	à
0x21	33	!	0x61	97	a	0xA1	161	¡	0xE1	225	á
0x22	34	"	0x62	98	b	0xA2	162	¢	0xE2	226	â
0x23	35	#	0x63	99	c	0xA3	163	£	0xE3	227	ã
0x24	36	$	0x64	100	d	0xA4	164	¤	0xE4	228	ä
0x25	37	%	0x65	101	e	0xA5	165	¥	0xE5	229	å
0x26	38	&	0x66	102	f	0xA6	166	¦	0xE6	230	æ
0x27	39	'	0x67	103	g	0xA7	167	§	0xE7	231	ç
0x28	40	(	0x68	104	h	0xA8	168	¨	0xE8	232	è
0x29	41	)	0x69	105	i	0xA9	169	©	0xE9	233	é
0x2A	42	*	0x6A	106	j	0xAA	170	ª	0xEA	234	ê
0x2B	43	+	0x6B	107	k	0xAB	171	«	0xEB	235	ë
0x2C	44	,	0x6C	108	l	0xAC	172	¬	0xEC	236	ì
0x2D	45	-	0x6D	109	m	0xAD	173	-	0xED	237	í
0x2E	46	.	0x6E	110	n	0xAE	174	®	0xEE	238	î
0x2F	47	/	0x6F	111	o	0xAF	175	¯	0xEF	239	ï
0x30	48	0	0x70	112	p	0xB0	176	°	0xF0	240	ð
0x31	49	1	0x71	113	q	0xB1	177	±	0xF1	241	ñ
0x32	50	2	0x72	114	r	0xB2	178	²	0xF2	242	ò
0x33	51	3	0x73	115	s	0xB3	179	³	0xF3	243	ó
0x34	52	4	0x74	116	t	0xB4	180	´	0xF4	244	ô
0x35	53	5	0x75	117	u	0xB5	181	µ	0xF5	245	õ
0x36	54	6	0x76	118	v	0xB6	182	¶	0xF6	246	ö
0x37	55	7	0x77	119	w	0xB7	183	·	0xF7	247	÷
0x38	56	8	0x78	120	x	0xB8	184	¸	0xF8	248	ø
0x39	57	9	0x79	121	y	0xB9	185	¹	0xF9	249	ù
0x3A	58	:	0x7A	122	z	0xBA	186	º	0xFA	250	ú
0x3B	59	;	0x7B	123	{	0xBB	187	»	0xFB	251	û
0x3C	60	<	0x7C	124	|	0xBC	188	¼	0xFC	252	ü
0x3D	61	=	0x7D	125	}	0xBD	189	½	0xFD	253	ý
0x3E	62	>	0x7E	126	~	0xBE	190	¾	0xFE	254	þ
0x3F	63	?	0x7F	127		0xBF	191	¿	0xFF	255	ÿ