Technical Specifications

AC Operating Voltage:
85 – 264 VAC, 48 – 62 Hz

AC Power Supply Certification:
UL, CE, CISPR/FCC Class B

DC Input Operating Voltage:
10 to 30 Volts DC

DC Output Power:
24 Volts DC, 0.5 Amp (for 4-20mA transducers)

12 Volts DC, 1 Amp (intermittent for 5 Watt external radio)

AC Power Consumption:
<0.5 Amp

DC Power Consumption (at 24VDC):
0.1 Amp to 2 Amps depending on active input/output loads and option boards

On-Board AC Input Fuse Rating:
Field replaceable 2 Amp 115 VAC

AC Input Transient Protection:
Yes, 10,000A 120 Joule 150V MOV on board

DC Input Transient Protection:
Yes, field replaceable 2A fuse and 1500W MOV on board

Weight:
1 Pound

Storage Temperature Rating:
-40°C to +85°C

Operating Temperature Rating:
-10°C to +75°C (0 to 40°C for AC-powered version)

Humidity:
15-95% non-condensing

Input/Output Specifications

Analog Input Channels:
2(VS2), 8(VS4)

Analog Input Resolution:
16-bit

Analog Input Signal Type:
0-20 mA grounded, 0-5 Volt DC, 0-10 Volt DC

Analog Input Transducer Power Source:
May be external or use on-board supply

Analog Input Transient Protection:
600W TVS surge and RF filters

Analog Input Transducer On-Board Power Supply:
On-board 24 VDC with AC power or ~1 Volt below DC supply Voltage

Digital Input Channels:
2(VS2), 8(VS4)

Digital Input Channel Signal Type:
Low voltage (5V) contacts or logic level

Digital Input Signal Voltage Required:
None

Digital Input Signal Transient Protection:
600W TVS surge and RF filters

Digital Input Signal Status Indication:
On-board LEDs, one per channel

Digital Input Signal Cable Length:
Maximum 50 feet recommended

Digital Input Signal De-Bounce Time:
Approximately 0.2 seconds

Analog Output Channels:
4(VS4)

Analog Output Resolution:
16-bit

Analog Output Signal Source:
Each output may be driven by any register in the RTU

Analog Output Scaling:
Fully configurable by user

Relay Output Channels:
3(VS2), 4(VS4)

Relay Contact Ratings:
SPDT 10 Amp at 115 VAC, 5 Amp at 30 VDC

Relay Output Signal Indication:
3-4 on-board LEDs, one per channel, show relay states

Relay Control Source:
Any RTU register

Relay Control Method:
Approximately 40 relay control methods

Modbus and Communication

Modbus Slave Connections (Data Retrieval and RTU Config):
RS-232, RS-485, and USB. All may be used simultaneously.

Modbus Master Connections (For Polling Remote Devices):
RS-232, RS-485. Both may be used simultaneously.

Number of Remote Modbus Devices That May Be Polled:
256

Remote Device Baud Rates:
1200, 2400, 4800, 9600, 19200, 38400. Each device may be different.

On-Board Radio Option:
LoRa 900MHz 1 Watt ISM Module plugs in.

Transducer On-Board Power Supply Protection:
Yes, current limited with electronic fuse

Inputs That May Be Monitored by Modbus:
Every analog input, digital input, temperature, and DC voltage

Outputs That May Be Controlled by Modbus:
Every analog output and digital output

SYSTEM CONFIGURATION & PROGRAMMING

The RTU is a very complex device with many settings. It is also a little complex to program, however most applications may still be configured very quickly. In normal operation the RTU continually gathers data from its own signal inputs, such as the analog inputs. This data is stored in volatile (RAM memory, i.e. contents are lost if the power is removed) local memory. The RTU controls the local outputs such as the relays and analog outputs in many different ways. It responds to Modbus commands that arrive via the System serial port, it may poll remote devices for data via the Polling serial port or it may act as a 'slave' device to other Viking Scada devices.

The RTU has thousands of user adjustable non-volatile settings instructing it how to do all of these things, and another 1000 or so volatile registers that contain the result of various inputs or calculations. All of these registers may be read and written to by the user, to effectively configure and take data from the device. All of these registers may be accessed by serial Modbus commands coming into the System serial port. A few communications-related settings may also be accessed by using the WiFi interface using a web browser on a mobile device or computer. The RTU stores all of its settings inside non volatile memory on the PCB where they will remain until the unit is reconfigured. Connection to power is not required to maintain these settings; there are no batteries or similar volatile devices required for configuration storage.

Any general purpose Modbus master program (such as QuickMod by Azeotech) may be used to configure the device. However, because of the complexity, a custom PC program called VikingScada is provided by Viking Scada that allows user-friendly configuration of the RTU; it may also be used to display the current RTU inputs. However the configuration is changed the net result is the same, registers inside the RTU memory will have been changed and the RTU will now operate using the latest settings.

The full Modbus map is provided in a separate section which identifies every register inside the device. All features and functions are primarily described in terms of physical signals and Modbus registers, but some examples of configuration using VikingScada or the WiFi interface may also be provided.

Serial Port Configuration and Data Access

The System communication port has several configuration settings. The baud rate, Modbus Gap Time and Modbus Address may all be changed. Common settings may be entered from the WiFi interface. If any of the Modbus parameters are changed they will become effective immediately, so if a Modbus master is communicating with the RTU it will need to change its own communication parameters to match the RTU's new ones.

The Modbus specification has very strict definitions for the time a slave device should take to respond to commands from a Modbus master and the time that a gap between successive bytes in a packet may be. In ideal circumstances these definitions may work, but in real applications where the RTU may be used with phone modems, data radios, leased line connections, PCs, RS-232 to RS-485 data converters, internet connections etc. they may not always work. The Modbus Gap Time allows adjustments to enable communications via various pieces of equipment that may introduce delays. Modbus RTU messages start and end with a silent interval of at least 3.5 character times, which for a baud rate of 9600 bps is approximately 4 ms. The RTU is capable of reading a Modbus message, acting on the message, formulating a reply, then start transmitting it back to the Modbus master device as soon as this 4 ms time expires. The RTU is also capable of monitoring the Modbus data and detecting a gap between bytes in a message that is 4 ms in length for example.

Normally Modbus messages from a master, such as from a local PC with a built in serial port will usually have correctly formed data packets without any gaps between bytes. However, if there are gaps between bytes of more than 4 ms the RTU units may assume the packet has ended, process it and since the packet is not complete the RTU unit ignores the packet and does not reply. The Modbus master then indicates an error reporting that the RTU device did not reply to a Modbus message, when in fact a legal Modbus packet was not presented to the RTU device. This type of error seldom occurs on modern PC systems with a hardwired local connection; however they will occur when a telephone modem, radio or similar device is between units on a Modbus network. Even short packets of data sent directly between two telephone modems often result in smaller bursts of data at the receiving modem, separated by gaps of several milliseconds. The problem may also occur when some RS-485 interface devices are used that incorporate ‘automatic transmit enable’ circuits; these devices often use simple RC timing circuits to enable the driver output, with the result being they may still be driving the Modbus connection and corrupting data several milliseconds after the bus should have been released, when the RTU unit is trying to send a reply.

To overcome these problems the user may set the gap time in ms. Extending the gap time will delay a response to the Modbus master, so it should not be extended too long, or the Modbus master’s own timeout settings may need to be extended.

⚠ Warning

If this delay is set to very small values Modbus communication problems may occur, especially with remote modems and similar devices. It is possible to completely lose remote Modbus communications, and lose the capability to change the configuration back! If this occurs a direct serial or WiFi connection may be required to regain Modbus control to return modified registers such as the gap time, back to values that allow functionality with connected equipment.

For this reason if an RTU device is accessed within the first 30 seconds after powering up, it will ignore the user's Baud Rate and Gap Time silent time setting and use a Baud Rate of 9600 baud and Gap Time of 3 ms, therefore allowing communication with most standard devices. This gives the user (and the RTU configuration program) a means to restore settings that have been changed to inoperative ones.

Viking Scada devices include an onboard LoRa radio module that enables long distance wireless communication between masters and slaves. When using the radio for communications, it must be configured to transmit and receive on one of 27 channels (channel 0 - 26). By default every RTU using a particular channel receives traffic from every other unit on that channel, similar to how an RS-485 bus works.

If multiple networks of RTUs are operating on the same channel, a Group Address must be configured to prevent each network from receiving traffic from the other. Setting a non-zero Group Address sets the LoRa "Address" field to the Group Address value and enables the LoRa "Fixed Point" transmission mode. This does use the NET ID feature found in some LoRa radios. Instead, all radios use the same address in fixed point mode to achieve isolation. The Group Address can be thought of as a sub-channel, but radios using the same channel with different Group Addresses can still have RF interference. The Group Address is only for filtering the data that each RTU receives from the radio module (the radio still receives all traffic on its configured channel).

Inputs

There are 2 (VS-2) or 4 (VS-4) analog/digital inputs on the RTU. Each may be configured for analog (0-20mA, 0 to 10 Volts DC, 0 to 5 Volts DC), or digital (dry contact closures, switches, relay contacts, open collector transistors, pulses) by moving a jumper located by each input. The RTU firmware is not aware of the actual signal type being used; the jumper location simply changes the load resistance and input type.

The Modbus map is common between several devices (VS-2, VS-4, etc.) and allows for 32 inputs on an RTU. Only the registers that are applicable to a specific RTU type are populated.

Analog Inputs

When the onboard jumpers are set to either of the analog modes the inputs will be constantly read to sample the incoming analog signal levels, and the (16-bit) 'raw' result of these reads are placed directly into 32 Modbus registers starting at address 800. Also, at the end of each sample, a scaling conversion will be performed and the scaled values (engineering units) will be placed into Modbus registers starting at 768.

Typically engineering unit values are more useful than raw input values, for example a tank level of 0 to 20 FT, or a pressure of 500 to 2500 PSI. The RTU may scale the raw values, and the scaled values will then be placed in the scaled analog input registers starting at 768. The operator may then use either raw or scaled (or both) values for various control functions.

To generate scaled values five scaling parameters need to be entered:

- Scaling Input Span Low Value
- Scaling Input Span High Value
- Scaling Output Span Low Value
- Scaling Output Span High Value
- Display Format

The Input Span values correspond to the range of the Analog Input to be scaled. The Output Span values correspond to the range of the Scaled Values. The Low Input Span Value maps to the Low Output Span Value, the High Input Span Value maps to the High Output Span Value, and values in between are scaled linearly. The Display Format is for informational purposes only and does not affect the scaling calculation, but it can be used by users reading the scaled values to know how many decimal places are intended to be in the scaled result. The Display Format must not be set to Pulse Counting mode (value 14) when scaling analog inputs.

This scaling method provides flexibility and makes it easy to correct for scaling errors, for example due to the density of a fluid not being 1, such as saltwater. Whether the inputs are set for 0-20 mA, 0-10 Volts or 0-5 Volts, the scaling process is exactly the same.

Example 1:

Take a 4-20 mA flow transducer that outputs 4 mA at 0 bbl/day and 20 mA at 100 bbl/day. At top of the range (100 bbl/day = 20 mA at Analog Input) the RTU reads a raw analog value of approximately 65535. At the bottom of the range (0 bbl/day = 4 mA at Analog Input), the RTU reads a raw analog value of approximately 9828. Therefore the scaling feature needs to map the 13107 - 65535 raw range to the 0 - 100 bbl/day engineering range. This is set as follows:

- Scaling Input Span Low Value = 13107
- Scaling Input Span High Value = 65535
- Scaling Output Span Low Value = 0
- Scaling Output Span High Value = 100

Example 2:

The Modbus values are all integers and do not contain decimal places. However another set of configuration registers (starting at 6608) contain information on displaying a decimal point or multiplying the result by powers of 10. Take the same transducer from Example 1, but if the user needs to see 99.25 bbl/day instead of 99, this can be taken into account by adjusting the Output Span Values and noting the two decimal places in the Display Format registers. This is set as follows:

- Scaling Input Span Low Value = 13107
- Scaling Input Span High Value = 65535
- Scaling Output Span Low Value = 0
- Scaling Output Span High Value = 10000
- Display Format = x0.01

This means the Output Span Low and High values both are specified in counts of 0.01 instead of counts of 1. Therefore an input value of 39321 (= 12 mA = 50 bbl/day) produces an output value of 5000 (= 50.00 bbl/day).

Digital Inputs

When the onboard jumpers are configured as digital inputs or for pulse counting applications a pullup resistor supplies a current limited 5 Volt signal source to the relevant connector pin. As the input is shorted to ground by an external circuit, switch or dry contact a green LED will illuminate by the input indicating an ‘active’ digital input. When a digitally configured input is open circuit (i.e. ‘OFF’), the LED will be extinguished and the value read for that raw analog input will be approximately 65535, this will drop to approximately zero when the input is shorted or ‘active’.

The raw analog measurement values are scaled just the same as if the onboard jumpers were set to either analog input setting, but typically there is no need to scale the raw values when used as Digital Inputs. To use a Digital Input in pulse counting mode, the Display Format register is set for pulse counting mode, then the scaled value will be a count of the number of pulse edges irrespective of the Input/Output Span Values. The pulse count continues to 4095 then wraps back to zero. (It does not wrap at 65536 to maintain 4G protocol compatibility).

Analog Outputs

There are 0 (VS-2) or 4 (VS-4) analog outputs on the RTU board. Each output is capable of driving a signal from 0 to 20 mA. The RTU provides the source current, so there is no need for an external supply. The negative side of the load is connected to ground at the RTU. The logical signal that drives each analog output may be any Modbus register in the RTU, so the analog outputs may reflect any data that the RTU is aware of.

The digital-to-analog converter chip always outputs 0 mA for a raw value of 0 and 20 mA for a raw value of 65535. The analog outputs may also be scaled, so virtually any field application may be accommodated. The scaling is configured similar to the Analog Inputs, but typically only the Input Span needs to be configured. The Output Span is typically left as the default 0 - 65535. With the Output Span set to the defaults of 0 - 65535, the Input Span can be thought of as the Low Value setting the Analog Output to 0 mA and the High Value setting the Analog Output to 20 mA. To generate scaled values four scaling parameters need to be entered:

- Scaling Input Span Low Value
- Scaling Input Span High Value
- Scaling Output Span Low Value
- Scaling Output Span High Value

Example 1 - Basic Analog Input Mirror:

To set an Analog Output to mirror a 4-20 mA Analog Input, the scaling parameters are set as:

- Scaling Input Span Low Value = 13107

- Scaling Input Span High Value = 65535
- Scaling Output Span Low Value = 0
- Scaling Output Span High Value = 100

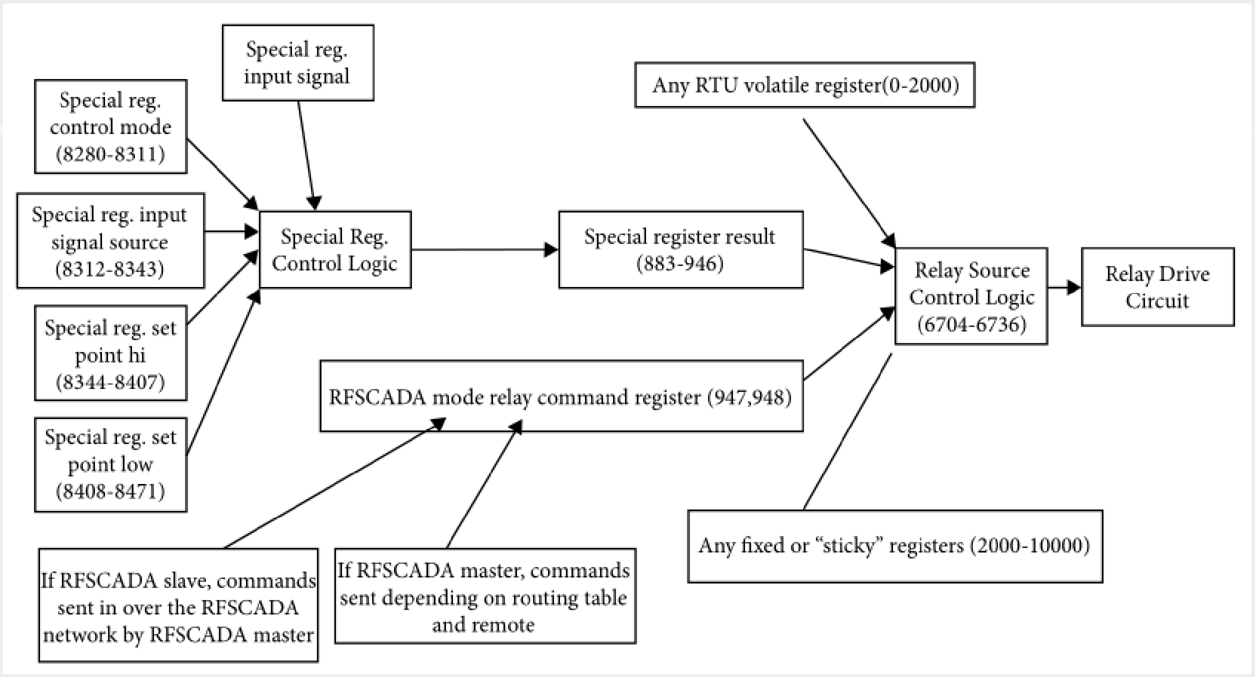
Example 1 - Advanced Analog Output from Modbus Slave:

Take a 4-20 mA flow transducer that is set as a polled Modbus device. The transducer has a Modbus register that provides the present flow from 0 bbl/day to 100 bbl/day as a number 0 to 100. To generate a 4-20 mA signal on an Analog Output for the 0 - 100 bbl/day range, the Source Register is set to point to the Live Value read from the Modbus device, and then Scaling Output Span needs to be set. At top of the range (100 bbl/day), the RTU needs to output 20 mA (Analog Output = 65535). At the bottom of the range (0 bbl/day), the RTU needs to output 4 mA (Analog Output = 13107). This is set as follows:

- Scaling Input Span Low Value = 0
- Scaling Input Span High Value = 100
- Scaling Output Span Low Value = 13107
- Scaling Output Span High Value = 65535

Digital (Relay) Outputs

There are 3 (VS-2) or 4 (VS-4) SPDT relays on the RTU board. There are many ways each relay may be controlled. The diagram below shows the options possible for controlling each of the relays.


Special reg. input signal
Special reg. control mode (8280-8311)
Special reg. input signal source (8312-8343)
Special reg. set point hi (8344-8407)
Special reg. set point low (8408-8471)
Special Reg. Control Logic
Special register result (883-946)
Any RTU volatile register(0-2000)
Relay Source Control Logic (6704-6736)
Relay Drive Circuit
RFSCADA mode relay command register (947,948)
Any fixed or "sticky" registers (2000-10000)
If RFSCADA slave, commands sent in over the RFSCADA network by RFSCADA master
If RFSCADA master, commands sent depending on routing table and remote

Each relay has a relay source control register which points to a memory location controlling when the relay is active. Typically if this unit is to be used as a Viking Scada 4G ‘slave’ each register is aimed at the ‘4G mode register’ which is controlled by the polling master RTU unit or PC. This is the default setting. In this case the relays are driven via a Modbus register that is controlled by the Viking Scada ‘master’. However each relay may also be assigned (locally, in this RTU) to be controlled by any register in this device. So as an example the first 4 relays could be controlled by a polling master but the remaining ones may be controlled by any register in the unit. If the control register that a relay is aimed at contains 0 the relay is inactive (i.e. off, the same state as if the board is not powered), and any other value turns the relay active (on). For very simple applications a relay control register may be aimed directly at a suitable control register. For example, setting register 6709 to contain 8664 means relay 6 will be driven by Modbus register 883, which contains the first volatile user register. If a user writes 1 to 883, the relay turns on. If the user writes 0, the relay turns off. Most relay controls are more complicated than this, and may require setpoints, hysteresis etc. The Viking Scada contains ‘special’ registers that may be configured for complex control purposes; and of course relays may then be aimed at the special registers for applications such as tank level control, relay activation on transducer out of bounds alarms etc. See the section on ‘special registers’ for a full explanation.

Special Registers

The 32 special control registers may be programmed to contain the results of certain parameters. These would typically be used to drive relays but may be used for other applications as well, allowing great flexibility for special control routines such as tank level controls, alarm outputs etc. They may be combined with other control routines such as the toggle functions to provide multiple complex controls that will often replace extensive programming in a typical PLC. Note that the control results will be stored in the special register ‘results’, not assigned to specific relays, so to use them for relay control the respective relay needs to be 'aimed' at the specific special register ‘result’.

There are many modes that can be used, some will require additional register pointers to use data sources, for example a register that may contain a tank level or user set points such as trip levels. First of all the control routines have to be set to an operating mode via the control register; there are over 40 modes available:

- Setting the control register to 0 effectively disables the special register ‘result’ and it will always be turned off.
- Setting the control register to 1 turns on the ‘result’ and it will remain on whenever the RTU is powered up.
- Setting the control register to 2 invokes 16-bit ‘drain level control’ for the respective ‘result’.
  - In this mode a 16-bit Modbus register will be monitored (specified by control source pointers, 8312 to 8343).
  - When this signal level exceeds the ‘high’ trip point (specified by every second register 8344 to 8407) the ‘result’ will be energized.

    - When this signal level is below the ‘low’ trip off point (specified by every second register 8408 to 8471) the ‘result’ will turn off.
    - This is effectively a tank level control system with hysteresis where the ‘result’ controls a pump that drains the tank. Once the level is above the high trip point the pump starts, and remains on until the level drops below the low trip off point. It may of course be used for many other applications such as an alarm when a pressure value is too high.
- Setting the control register to 3 invokes 16-bit ‘fill level control’ for the respective ‘result’.
    - This is identical to drain level control except the ‘result’ is energized when the control level falls below the low trip point, and turns off when the control signal is above the ‘high’ trip point.
    - It is typically used where a pump fills a tank, or an alarm is activated when levels become too low.
- Setting the control register to 4 causes the ‘result’ to be on if the Modbus source register is not equal to zero; it will be off if the source register is any other value.
- Setting the control register to 5 causes the ‘result’ to be off if the Modbus source register is not equal to zero; it will be on if the source register is any other value.
- Setting the control register to 6 through 21 causes the ‘result’ to be activated when a single bit is set in the control register.
    - Bit 0 is checked for a control register value of 6
    - Bit 1 is checked for a control register value or 7
    - ..
    - Bit 15 is checked for a control register value of 21
- Setting the control register to 22 through 37 causes the ‘result’ to be activated when a single bit is clear in the control register.
    - Bit 0 is checked for a control register value of 22
    - Bit 1 is checked for a control register value or 23
    - ..
    - Bit 15 is checked for a control register value of 37
- Setting the control register to 38 invokes 32-bit ‘drain level control’ for the respective ‘result’.
    - This uses the same logic as the 16-bit ‘drain level control’ but uses 32-bit control registers.
- Setting the control register to 39 invokes 32-bit ‘fill level control’ for the respective ‘result’.
    - This uses the same logic as the 16-bit ‘fill level control’ but uses 32-bit control registers.
- Setting the control register to 40 uses the polled device network status to set the ‘result’.
    - The ‘result’ is on if all devices are successfully operating and communicating with the RTU.
    - The ‘result’ is off if no remote devices are communicating with the RTU.

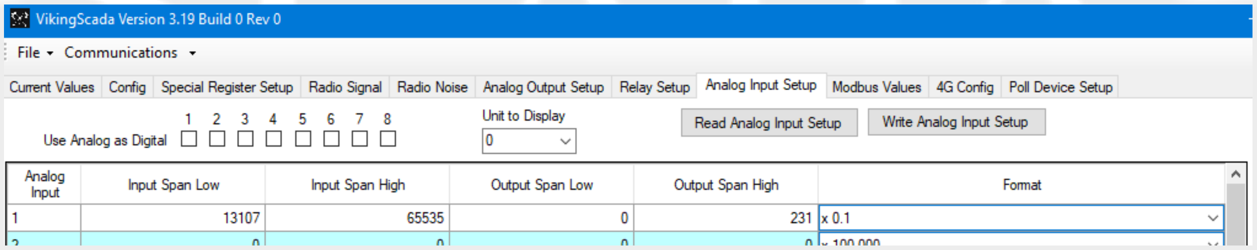
- The ‘result’ slowly toggles between on and off if some but not all remote devices are successfully communicating

Special register configurations also include Trip On Delay and Trip Off Delay parameters. This is used to let the input condition settle before changing the output. The Trip On Delay sets a delay between the control condition going from Off to On and the Live Value changing from Off (0) to On (65535). The Trip Off Delay sets a delay between the control condition going from On to Off and the Live Value changing from On (65535) to Off (0). When one of the delays starts counting because the input condition changed, the count will be restarted if the condition changes back. This prevents noisy or chattering inputs from chattering the outputs.

Example: Special Register Lead / Lag Tank Level Control

This example shows how a common application for tank level control may be set up, with a 'lead' and 'lag' pump output relay. This example will be for pumps that drain the tank; a tank fill version is almost identical with reversed on / off set points. Typically these systems have a 'lead' pump that comes on first when the process level reaches a user preset high 'turn on' trip point. If the fluid continues to rise it may reach a second 'trip point' when the 'lag' pump will turn on. When the fluid level recedes to the respective 'trip off' points each pump will shut down. Although the lead and lag pump typically use the same process variable (fluid level) as the source, each has independent 'trip on' and 'trip off' set points.

First set up the analog input to be used for the level control. We will use a 4 - 20 mA 0 - 10 PSI transducer, which with water will correspond to a full scale height of 23.1 ft. We will display the level in VikingScada with a resolution of 0.1 ft, so the format multiplier will be set to x0.1 with an output span high of 231. The 4 - 20mA transducer will show 65535 as the 20 mA / 23.1 ft input level, and 4.0 mA will correspond to 13107 or 0 ft.

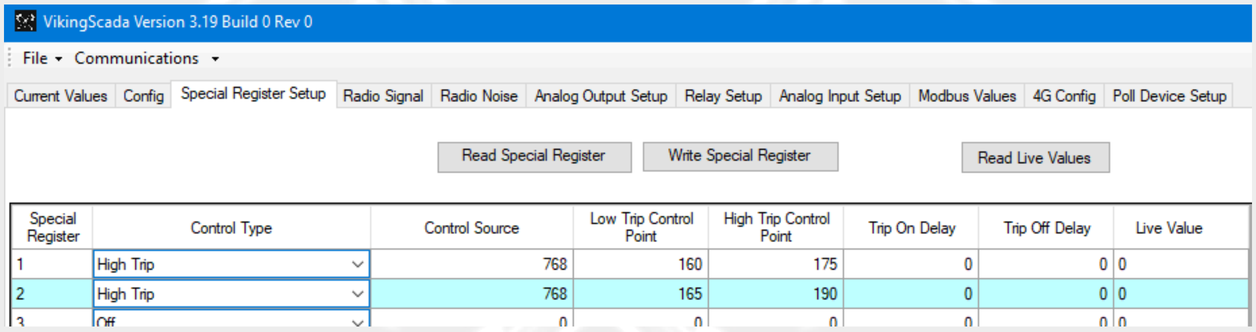


Use Analog as Digital 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐

Unit to Display: 0

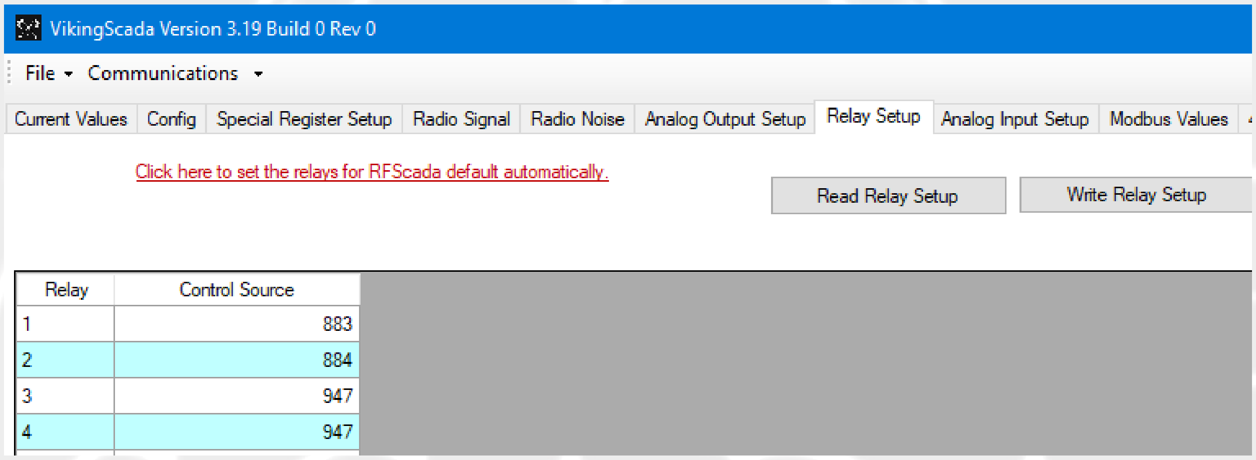
Analog Input	Input Span Low	Input Span High	Output Span Low	Output Span High	Format
1	13107	65535	0	231	x 0.1
2	0	0	0	0	x 100.000

Next we configure the lead (special 1) and lag (special 2) registers. Set them both to high trip for tank drain applications. They will both use a control source of 768, which is the scaled register for analog input 1 (un-scaled is 800). The value in 768 will vary between 0 (0 ft) and 231 (23.1 ft). If the value goes above the high trip control point (17.5 ft for the lead, 19.0 ft for the lag) the respective outputs will turn on, i.e. special registers 1 (883) & 2 (884) will then contain 65535. They will remain that way until the control source drops below the low control trip point (16.0 ft for the lead, 16.5 ft for the lag) when they will revert to containing 0.



Special Register	Control Type	Control Source	Low Trip Control Point	High Trip Control Point	Trip On Delay	Trip Off Delay	Live Value
1	High Trip	768	160	175	0	0	0
2	High Trip	768	165	190	0	0	0
3	Off	0	0	0	0	0	0

Any relays that are aimed at 883 and 884 will then be driven by these special registers as shown for the first two relays. By setting these parameters the level control can quickly be implemented. The high / low trip points may be changed by any local or remote Modbus device, typically a flat panel display.



Relay	Control Source
1	883
2	884
3	947
4	947

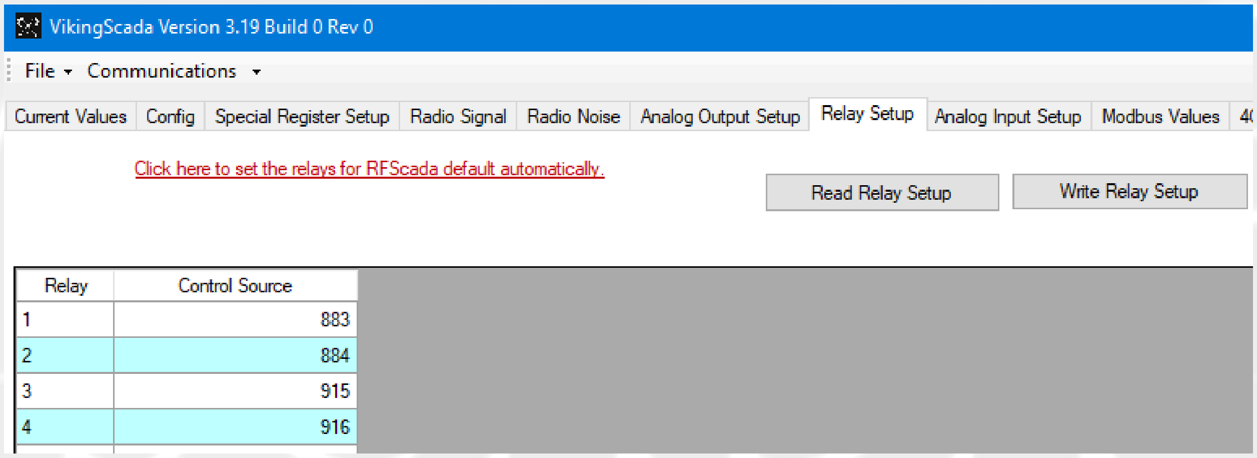
Toggle Registers and Routines

Often it is desirable to toggle certain registers depending on events. This capability is frequently used for dual pumps, where redundancy and equal wear is required. There are 4 sets of toggle functions that may be used, they are all used in a similar manner. The ‘outputs’ will be sent to the toggle results, which are the first 8 of the reserved special registers (915 to 922). The toggle functions operate by copying the contents of any two preset registers in the RTU to two toggle result registers. When the value in the first preset register drops to zero the two toggle result register contents will be ‘swapped’, ie the contents of the first preset will go to the second toggle result, and the contents of the second preset will be copied to the first toggle result. This signal ‘routing’ will remain that way until the preset register 1 contents changes to a non zero value and back to zero again. The toggle registers must be written directly, with 2 registers per toggle function pair (6953, 6954 for the first pair, 6955, 6956 for the second, 6957, 6958 for the third and 6959, 6960 for the fourth). This method allows for easy lead / lag pump swapping, although it may be used for other purposes.

There are 64 special registers (883 - 946). The first 32 (883 - 914) are the ‘results’ of the 32 special register functions. The second 32 (915 - 946) are reserved for general purpose user functions, but the first 8 of these (915 - 922) are used by the toggle function if it is configured. All 64 special registers are in volatile RAM memory so the contents will be lost upon power failure; they will be loaded with zeros upon power up.

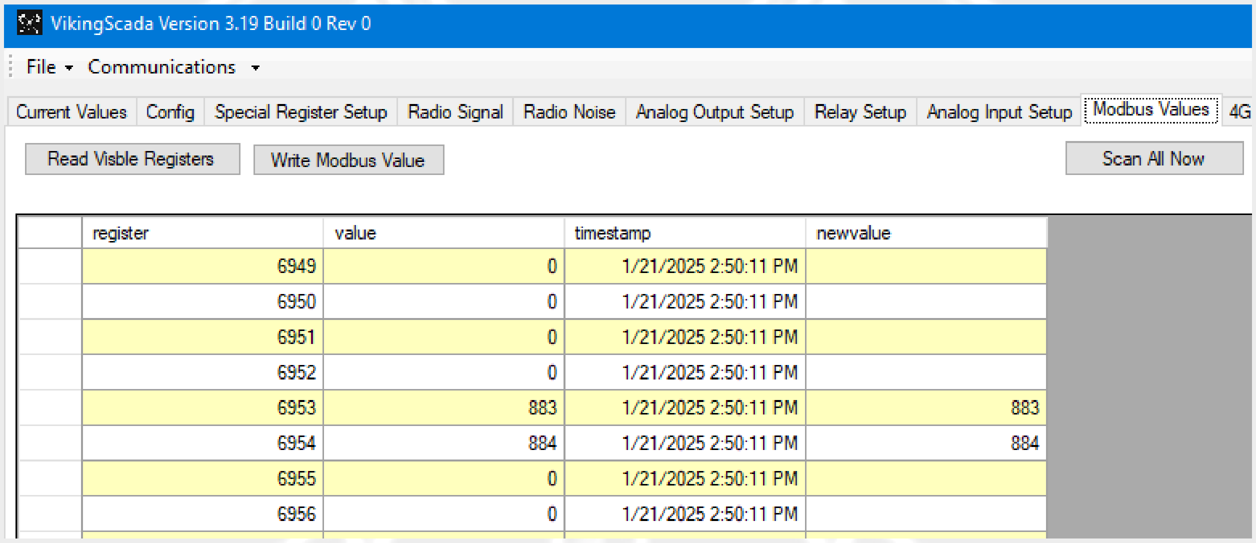
Example: Toggle – Adding pump switching to the Special Register example above

In the above lead / lag example one pump would perform most of the pumping. It is easy to add duplex switching capability by using the toggle function. Assume the lead / lag example above is already set up. We will drive relays 3 and 4 in a lead / lag in sync with the first two relays, however relays 3 & 4 will toggle the lead routing at the end of every lead call, this way the wear and run times will be spread between the two pumps connected to relays 3 and 4. As shown below Relays 3 & 4 will be aimed at the first two toggle output registers 915 & 916.



Relay	Control Source
1	883
2	884
3	915
4	916

Next using the Modbus Values tab change the contents of the toggle lead register (6953) to 883 (special register 1 output, i.e. the lead call) and toggle lag register (6954) to 884 (special register 2 output, i.e. the lag call). Now the toggle function will automatically switch the routing as needed every time a lead call ends, sending the results to the two registers 915 and 916. Since relays 3 & 4 are aimed at 915 and 916 they will perform the same lead / lag output as relays 1 & 2 but with the lead call switching between relays 3 & 4. Note that the toggle routines are called about every second to reduce relay chatter if incorrectly programmed or fast changing signals are applied. To disable the toggle function write a 0 or 65535 to either of the toggle source registers (6953 or 6954 in this example).



register	value	timestamp	newvalue
6949	0	1/21/2025 2:50:11 PM	
6950	0	1/21/2025 2:50:11 PM	
6951	0	1/21/2025 2:50:11 PM	
6952	0	1/21/2025 2:50:11 PM	
6953	883	1/21/2025 2:50:11 PM	883
6954	884	1/21/2025 2:50:11 PM	884
6955	0	1/21/2025 2:50:11 PM	
6956	0	1/21/2025 2:50:11 PM	

Sticky Registers

Sometimes control registers are used that need to retain contents during power cycles; these are called Sticky Registers. For example, a cyclic satellite may transmit periodic commands to the RTU such as a VSD speed control. The satellite typically comes by every 1 ½ hours, and if the RTU power is cycled during that time the most recent speed command should be restored by the RTU upon power up.

There are 256 sticky registers in the RTU that are available for general purpose use, starting at 8664. The registers are stored in non-volatile EEPROM memory so are not dependent upon power. Like all registers they do however have a limited write capability (typically 10,000 to 100,000 writes). The RTU contains intelligent routines that attempt to limit over writing to the memory. When a write is made it is not be saved for a second or so (the blue 4G led flashes quickly) in case the user immediately changes the value. Also the RTU does not write the same value to an existing value in EEPROM (it is simply ignored) so it is safe to continually write the same value to a Sticky Register.

Polling Port Device Setup - Master & Slave Configuration

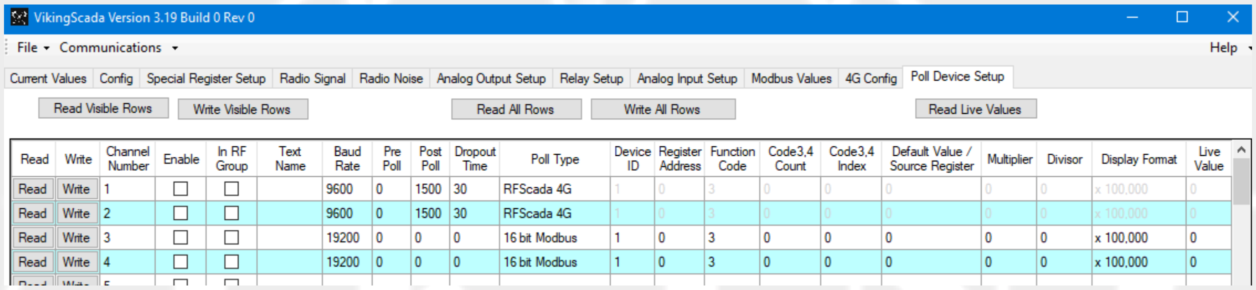
The RTU may be set up in two basic modes, Master or Slave, each with many permutations available. The RTU may also be set up to communicate with other devices using one of three communication modes: RFScada 4G, Modbus, or RFScada Classic. The RFScada 4G and Classic protocols are not exclusive to RFScada devices (Viking Scada devices use both protocols as well); their names have been kept to avoid confusion. The RFScada 4G protocol may be referred to as simply the 4G protocol in this document.

The first main mode is as a Slave device, where other devices may poll this RTU requesting data and sending commands to it. This would typically be used if this RTU were to be accessed via a SCADA system or PC, another Viking Scada RTU that was the system ‘master’, or another third party device that could poll this RTU. In the slave configuration the RTU may be set to respond to Modbus commands or RFScada 4G commands, so can operate in a mixed system. It may also be set to emulate two smaller RFScada units using an older RFScada protocol for compatibility with non-4G legacy RFScada systems.

If configured as a Master the RTU may poll many types of device, such as other Viking Scada units using the 4G protocol and almost any Modbus device.

The descriptions in this section exclusively refer to the VikingScada software since the polling configuration involves many Modbus registers. However, all configuration can also be done using only Modbus registers.

For reference, the relevant tab in the VikingScada software is shown below:



Read	Write	Channel Number	Enable	In RF Group	Text Name	Baud Rate	Pre Poll	Post Poll	Dropout Time	Poll Type	Device ID	Register Address	Function Code	Code3,4 Count	Code3,4 Index	Default Value / Source Register	Multiplier	Divisor	Display Format	Live Value
Read	Write	1	☐	☐		9600	0	1500	30	RFScada 4G	1	0	3	0	0	0	0	0	x 100,000	0
Read	Write	2	☐	☐		9600	0	1500	30	RFScada 4G	1	0	3	0	0	0	0	0	x 100,000	0
Read	Write	3	☐	☐		19200	0	0	0	16 bit Modbus	1	0	3	0	0	0	0	0	x 100,000	0
Read	Write	4	☐	☐		19200	0	0	0	16 bit Modbus	1	0	3	0	0	0	0	0	x 100,000	0
Read	Write	5	☐	☐																

Polled Device Setup - Slave Configuration

In this mode the RTU must be assigned an address that is unique to the ‘network’, ranging from 1 to 255. The protocol used to communicate with the RTU also must also be defined, and typically is set to be RFScada 4G mode or Modbus. Note that 4G and Modbus protocols can co-exist on the same network, so 4G and Modbus devices may have the same ID’s if needed since each only answers to their own protocols. The RFScada Classic mode may also be selected, in this case the RTU emulates 2 RFScada 16 channel boards for compatibility with legacy systems.The communications port baud rate must also be defined, and is typically the same for the whole network, although the baud rates may be mixed.

When the PC software is used to configure the RTU after slave mode has been selected in the Config tab (or read from the connected RTU) several of the tabs not relevant to slave operation for this RTU disappear. When the RTU is set in RFScada slave mode the relays may be placed in RFScada default mode by selecting the default setup button, which assigns the relays and analog outputs to be controlled by commands arriving from the polling port. However, any of the relays or analog outputs may be assigned for local control regardless of the polling commands. If the RTU is set to respond as a Modbus device, then the polling Modbus master is able to query and write any registers in the RTU. The action of the analog outputs and relays are governed by the RTU control settings which may be set to respond to remote Modbus commands.

Polled Device Setup - Master Configuration

The RTU may be configured to poll up to 255 various remote devices, accumulate data from them and send data to certain types of device. The first 32 devices may be configured to communicate using either 4G or Modbus protocols. The remaining devices can only be configured to use Modbus. This is configured using the Poll Type setting. The device number is also referred to as the Channel Number. The master continuously cycles through all enabled polled devices.

Polled Device Setup - General

Some of the Polled Device settings apply to every polled device, regardless of which protocol is used to communicate with it. The remaining settings only apply when communicating with a polled device using Modbus. The common settings are described below.

The Enable setting enables or disables the device. If enabled, the device is polled in order and the Live Value is updated after each successful response or set to the default setting if the Dropout Time has elapsed since the last successful poll. If disabled, the device is never polled and the Live Value is set to the 'default' setting.

The In RF Group setting enables or disables special data handling for a slave that is configured with the same LoRa RF group value as the master. If enabled, the polled command prefixed with the RF Group Address configured in the master. If disabled, the polled command is not modified. If the RF Group Address is configured in the slave and the slave is communicating using the radio, this must be enabled. See the Serial Port Configuration section for more information on RF Group Addressing.

The Text Name is an ASCII label that is stored in the master to help users remember which polled device is which.

The Baud Rate setting is the baud rate used when polling a specific device. Each slave can operate at a different rate. Note that some radio modems (including the optional on board LoRa radio) are configured to operate at a single baud rate, so it may not be possible to communicate with devices over radios if they have different baud rates.

The Pre Poll setting configures a time delay (in milliseconds) between processing a reply from the last command and issuing the next command. It is not normally required but is available if needed when operating for example with older radio modems that take a substantial time to 'key down' after transmitting data.

The Post Poll setting configures a time delay (in milliseconds) after the command has been transmitted before the RTU it inspects the reply (if any) from the remote device. Typically this value may be 50 to 500 ms with directly connected devices; it may be up to several seconds if a radio or internet connected device is being polled.

The Dropout Time setting sets the communication timeout for a device. If a polled device does not respond to data requests for a preset time the RTU will change the polled result in the result table to a default value. This allows alarms or similar actions to occur if communications are lost to a remote device. The timeout is set in 10 second increments up to about 7 days, so a value of 3 is 30 seconds.

The Poll Type setting sets which protocol to use when communicating with the device. The two main options are RFScada 4G and Modbus, but each has further options. The

RFScada 4G protocol can be configured to request 8, 16, 24, or 32 analog inputs from a device. Requesting more than 8 analog inputs requires additional polling commands/responses since only 8 analog inputs are included in a single 4G response. The Modbus protocol can be configured to request a 16-bit or 32-bit register value, where the 32-bit word order (High Low vs. Low High) can also be configured. The RTU always stores the result in High Low order, where the high 16 bits are in the lower address and the low 16 bits are in the higher address. Some Modbus devices store a 32-bit value in Low High order, where the low 16 bits are in the lower address and the high 16 bits are in the higher address.

Polled Device Setup - Modbus

When the Poll Type is set to Modbus, the ‘device’ is actually a single Modbus register in some physical unit which may be accessed via a wired or wireless connection using the polling ports connection. Each device is configured with many individual parameters, so mixed types and brands of equipment may easily be polled. Each Modbus-specific parameter is explained.

The Device ID setting sets the Modbus ID for each device, within the range of 0 to 255. Multiple polled devices may have the same Device ID if multiple registers from the same physical device are required.

The Register Address setting sets the 16-bit Modbus address to be accessed. Note that many Modbus maps show addresses starting at 30001 or 40001; these are usually translated to ‘real’ addresses starting at 0 or 1 by the Modbus host software. The RTU will directly transmit whatever address is programmed without adding or removing any offsets to allow full coverage of the whole address range.

The Function Code setting is the Modbus command that will be used to access the device, typically either command 3 or 4 (read single register or input). Function Codes of 1, 2, 5, 6, 15, and 16 are also supported. For Function Codes that read values (1, 2, 3, 4), if a communication timeout occurs, the Live Value is set to the Default Value setting. For Function Codes that write values (5, 6, 15, 16), the Live Value is not applicable.

The Code3,4 Count and Code3,4 Index settings apply when the Function Code is set to 3 or 4 (which can read multiple registers in a single transaction). Each polled device only stores the result of a single 16-bit or 32-bit Modbus register, but some Modbus slaves do not have individually addressable values. The Code3,4 Count setting sets how many registers the Modbus command reads, and the Code3,4 Index setting sets which register the polled device keeps as its Live Value. For example, a Modbus slave may have 8 analog inputs that can only be accessed by reading 8 register values starting at address 100. To access the third input, the Register Address is set to 100, the Code3,4 Count is set to 8, and the Code3,4 Index is set to 2.

The Default Value / Source Register setting configures the default Live Value for Function Codes that read (1, 2, 3, 4) or the register in the master used as the output data for Function Codes that write (5, 6, 15, 16).

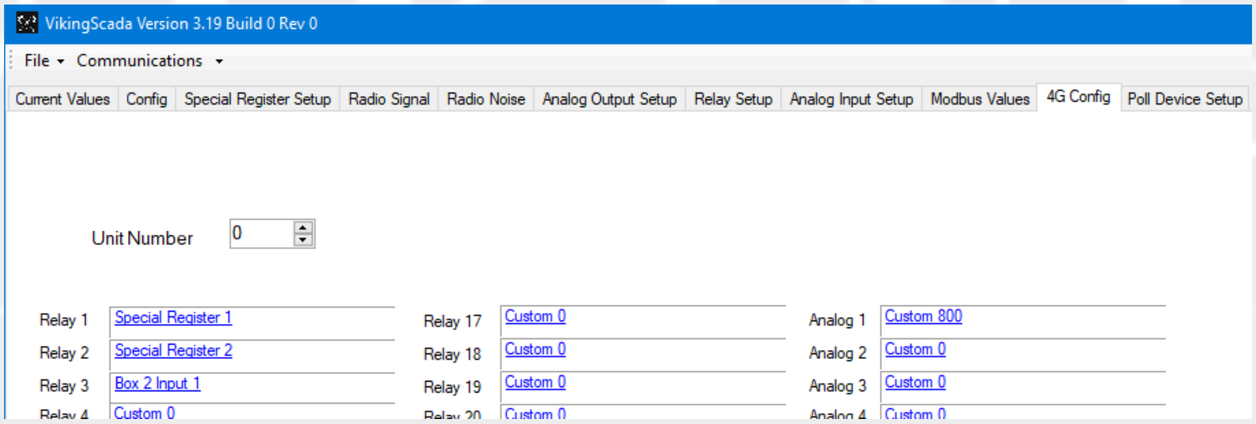
The Multiplier and Divisor settings define the scaling operations performed on the Live Value before storing into the scaled value register for the device. The raw Modbus value is always stored in the Live Value registers 0 - 511. The scaled values (Raw * Multiplier / Divisor) are stored in the Scaled Value registers 512 - 767. The Scaled Values are only stored as a 16-bit value, but the Live Values may be stored as a 32-bit value.

The Display Format setting sets the number of decimal places in the Live Value. This setting does not affect the raw values read from the Modbus device, but only affects what is shown in the Live Value field, in a similar fashion to the scaled Analog Inputs. This is not affected by the Multiplier or Divisor.

Polled Device Setup - RFScada 4G

When the Poll Type is set to 4G, the ‘device’ is another VikingScada or 4G-compatible RFScada device. The 4G protocol receives the raw analog inputs from each polled device and sets the analog and digital (relay) outputs of each polled device based on the configured 4G control parameters. This means there is no protocol configuration beyond the common polled device configuration. However, in order to set each polled 4G slave device’s analog and digital (relay) outputs, a separate 4G configuration is used.

For each 4G device, the master included, the analog and digital outputs are configured individually. As an example below, the master (Unit 0) is configured to have its Relay 1 and 2 controlled by the output Live Value of Special Registers 1 and 2 respectively. Its Relay 3 is controlled by 4G slave ID 2 (Box 2) Input 1, and its Analog Output 1 is controlled by Modbus register 800 in the local (the unit currently connected to VikingScada) master. The 4G protocol is generally used to route an input of any 4G device to an output of 4G device output, but can also be used to control outputs from any Modbus register in the master.



Unit Number 0

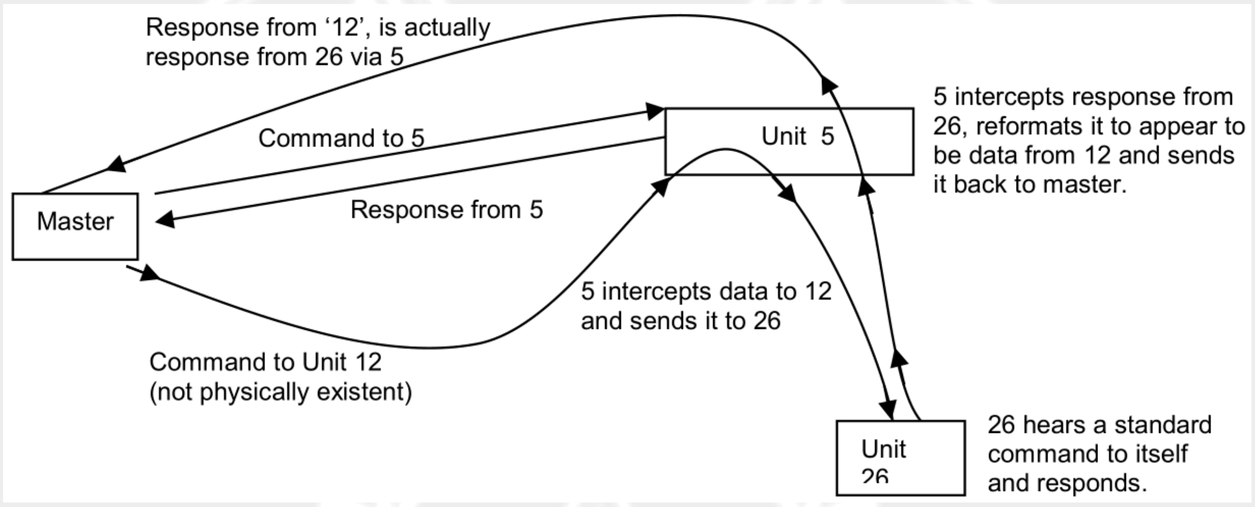
Relay 1	Special Register 1	Relay 17	Custom 0	Analog 1	Custom 800
Relay 2	Special Register 2	Relay 18	Custom 0	Analog 2	Custom 0
Relay 3	Box 2 Input 1	Relay 19	Custom 0	Analog 3	Custom 0
Relay 4	Custom 0	Relay 20	Custom 0	Analog 4	Custom 0

Repeater Mode

In normal installations one Viking Scada device is set up as the master unit, which polls all slave units in turn reading and writing data to them. This means that the master unit must be able to communicate directly with all Viking Scada slave devices in the network. For some installations such as pipelines it may not be possible to obtain reliable communications between the master and all the slave units due to distances or obstructions. In these cases slave units may be configured to act as repeaters, passing the data between the master and other slaves.

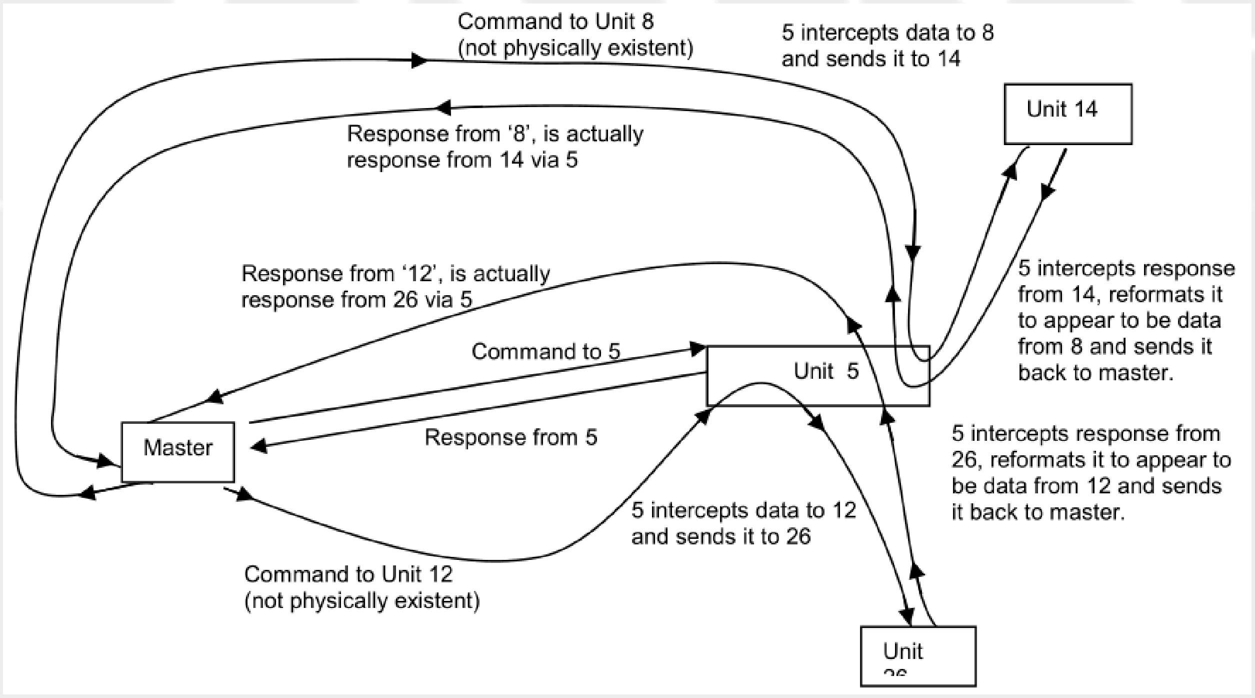
Basic Repeater Operation

The concept is fairly straightforward, example ID's are referenced to make the example easier to follow. A repeater may be any slave unit that is configured to recognize and react to data intended for its own ID (e.g. 5), just like any other slave. However, it is also configured to recognize data intended for the 'target' slave unit (12). When it recognizes this data it translates the data to another target address (26) which the master is not set to communicate with. Unit 5 will then impersonate a master unit by transmitting the data on to the target slave (26), which will then respond back to the repeating slave (5). The repeating slave (5) will take the response from (26), reconfigure it to look like data from (12) and return it to the master unit. The result is that the master is communicating with two slaves, one (5) and the other which it thinks is (12) but is actually the physical unit (26), relayed back via (5). The master unit is not even aware of the existence of a repeater unit.


Response from '12', is actually response from 26 via 5
Command to 5
Unit 5
5 intercepts response from 26, reformats it to appear to be data from 12 and sends it back to master.
Master
Response from 5
5 intercepts data to 12 and sends it to 26
Command to Unit 12 (not physically existent)
Unit 26
26 hears a standard command to itself and responds.

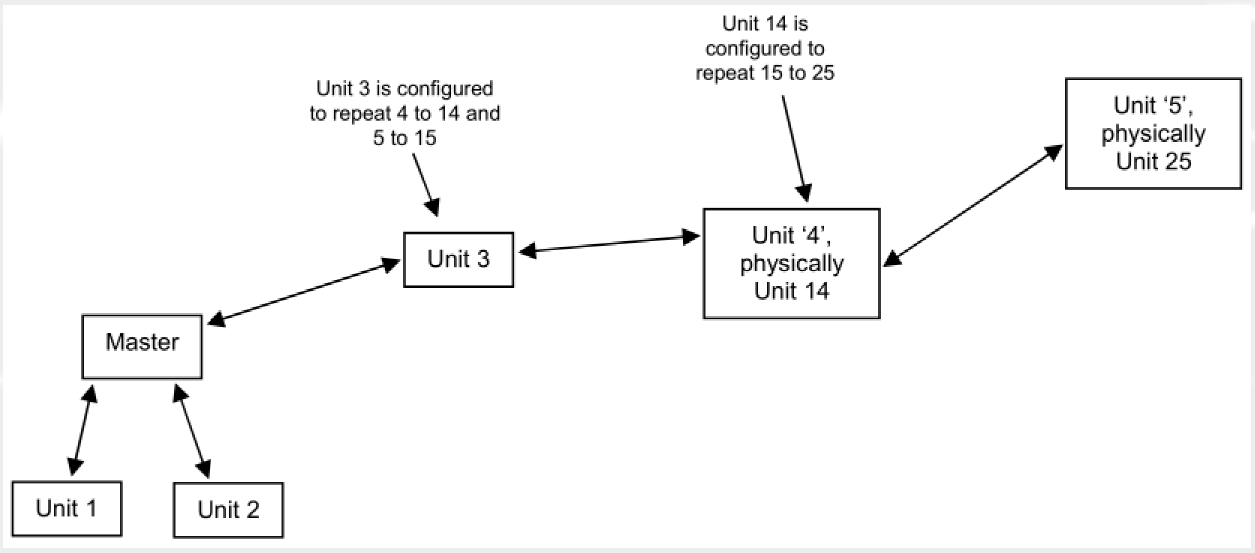
Advanced Repeater Operation

The above example may be expanded by configuring the repeating slave to respond to multiple 'target' units, each corresponding to a physically non existent slave unit. For example, in addition to the above example the repeater could also intercept data to unit 8 and send it to physical unit 14. More than one slave repeater may exist in range of the master provided none of the ID's conflict, so the advanced mode may be expanded.


Command to Unit 8 (not physically existent)
5 intercepts data to 8 and sends it to 14
Unit 14
Response from '8', is actually response from 14 via 5
Response from '12', is actually response from 26 via 5
5 intercepts response from 14, reformats it to appear to be data from 8 and sends it back to master.
Command to 5
Unit 5
Master
Response from 5
5 intercepts response from 26, reformats it to appear to be data from 12 and sends it back to master.
5 intercepts data to 12 and sends it to 26
Command to Unit 12 (not physically existent)
Unit 26

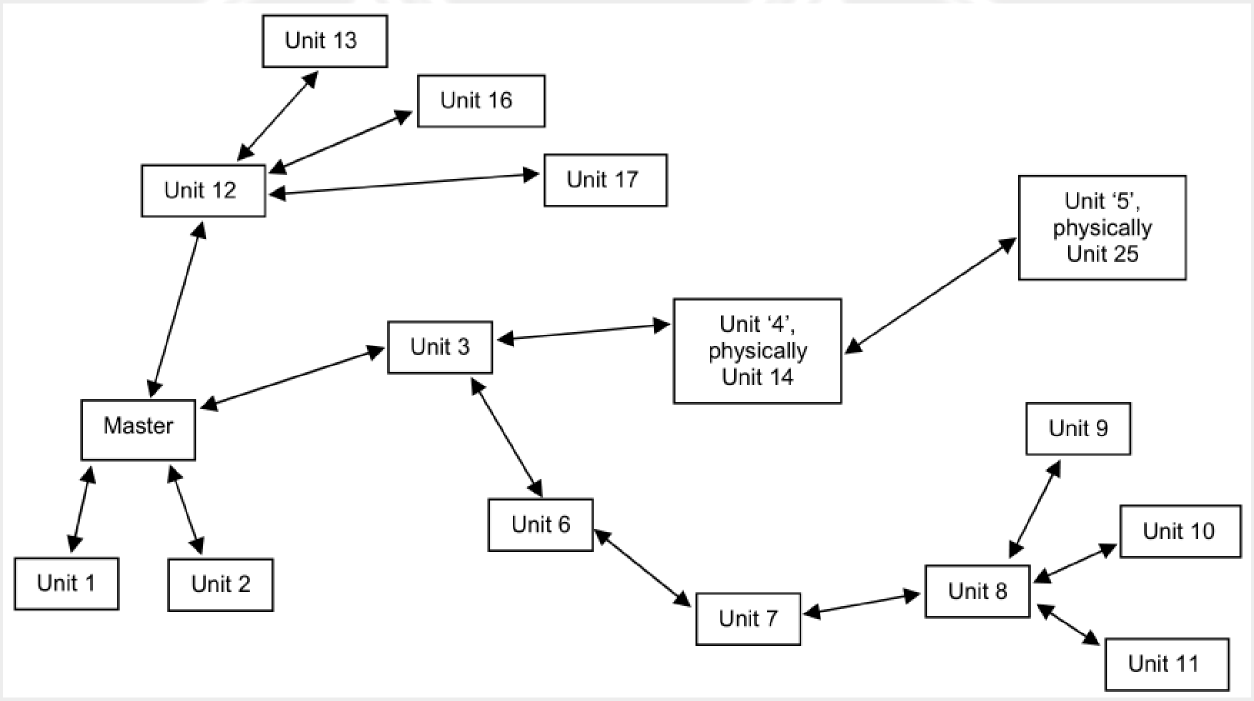
Daisy Chained Repeater Operation

More than one repeater may be placed in series, which may be useful in pipeline and similar applications. In the example below the master is configured to communicate with 5 slaves, ID's 1,2,3,4 and 5. The slaves 1, 2 and 3 are directly in contact with the master. Slave 3 is also a repeater, configured to intercept data intended for units 4 and 5 then translate it to units 14 and 15. The fourth slave, which the master addresses as unit 4, is actually physically configured for address 14 and it too is also a repeater. It translates the data intended for 15 to 25, which is the final slave, physically configured as unit 25 but referred to by the master as 5. So when data is sent from the master to unit 5 it is intercepted by 3, converted to '15' and sent out. It's then intercepted by 14, converted to 25 and sent out. It is then received by 25 which replies. 14 receives the reply from 25, translates it back to appear to be data from 15. Unit 3 picks it up, and in turn translates it to appear to be data from 5. The master receives it and thinks it got a direct response from unit 5. In a similar manner when the master sends a command for unit 4 it is intercepted by unit 3 which translates it to unit 14 data, communicates with unit 14 then returns the response to the master as if it was a response from unit 4. In all of these examples the master unit is not aware that repeaters are in the system. The daisy chained repeaters may be extended for more than two repeaters.


Unit 14 is configured to repeat 15 to 25
Unit 3 is configured to repeat 4 to 14 and 5 to 15
Unit '5', physically Unit 25
Unit 3
Unit '4', physically Unit 14
Master
Unit 1
Unit 2

Complex Repeater Operation

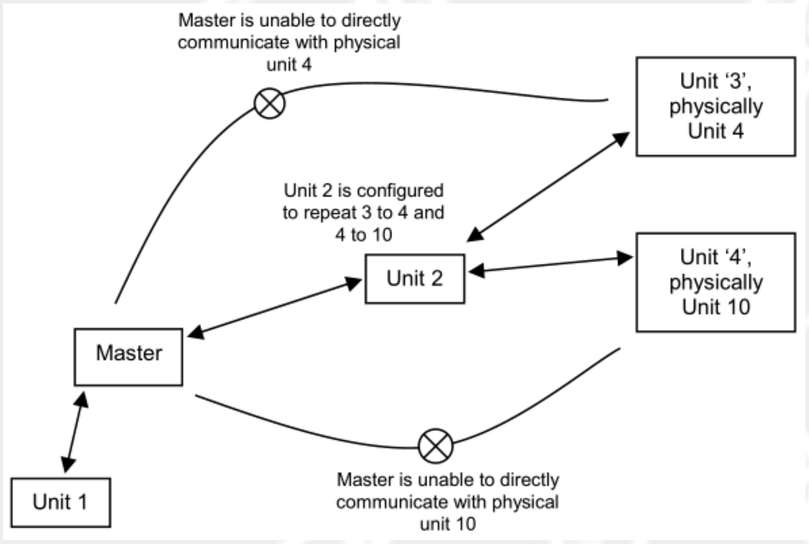
Multiple repeaters may be daisy chained in series, repeaters may be on additional daisy chains branching off the original chain and multiple units may be accessed via any unit in the system for very complex configurations.



Repeater Considerations

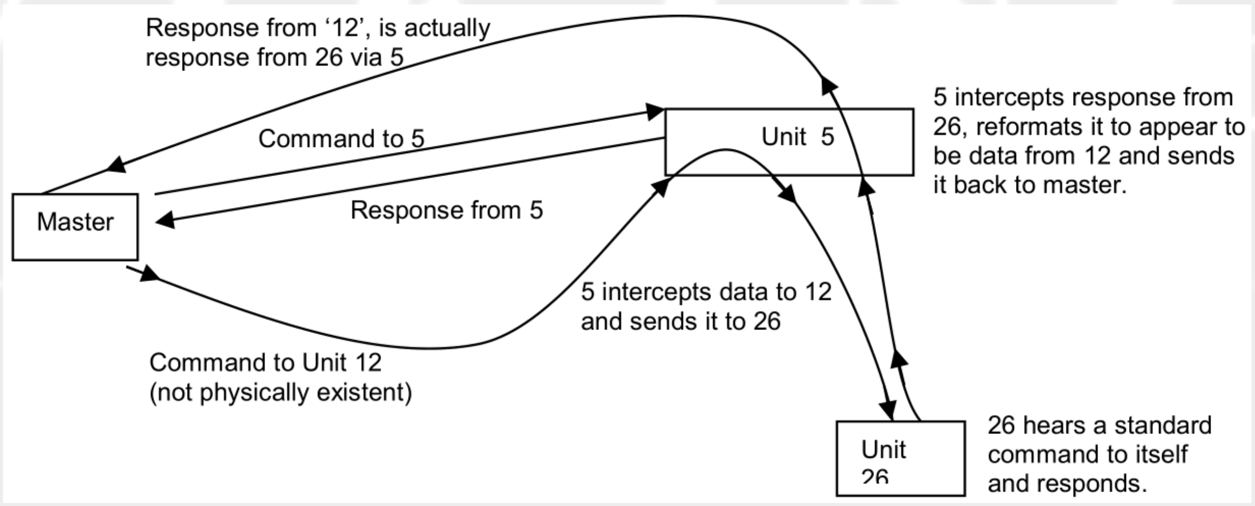
The techniques used for repeater operation depend on repeating slaves translating data intended for physically 'non-existent' slave units. The ID's used for the non–existent and the physical units cannot be the same if there is any possibility that the master may receive transmissions from the remote slave, so to operate a unit via a single repeater actually requires two slave "ID's". Operating a unit via two daisy chained repeaters consumes 3 ID's, and so on. On small systems this is not a problem, but may be a consideration in larger systems if the units have firmware installed before 5.0 that just supports one master and 31 slave units. It is possible to reuse an ID in a repeater to remote slave link provided the master is unable to communicate with the remote slave; this would only be required in cases where all ID's had been consumed. An example is shown below. The master is enabled to communicate with units 1, 2, 3 & 4.

If communications to a repeater are lost then communication to all subsequent units receiving signals via the repeater will be lost too. Another factor to consider when using repeater mode are the time delays associated with transmission of data back and forth between all the repeaters, so timeout settings and scanning rates may need to be adjusted as the repeater depth is increased; this may be done from the 'radio config' settings. The default settings allow for two daisy chained repeaters which would cover most normal operations.


Master is unable to directly communicate with physical unit 4
Unit '3', physically Unit 4
Unit 2 is configured to repeat 3 to 4 and 4 to 10
Unit 2
Unit '4', physically Unit 10
Master
Unit 1
Master is unable to directly communicate with physical unit 10

Repeater Requirements and Configuration

Configuration is straightforward. First assign ID's for units in the system, which will make the process more straightforward. The configuration for the first example will be explained.

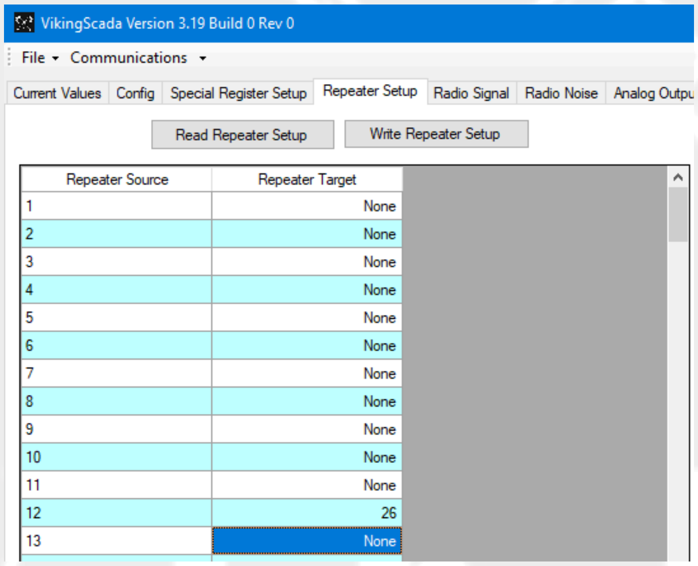

Response from '12', is actually response from 26 via 5
Command to 5
Unit 5
5 intercepts response from 26, reformats it to appear to be data from 12 and sends it back to master.
Master
Response from 5
5 intercepts data to 12 and sends it to 26
Command to Unit 12 (not physically existent)
Unit 26
26 hears a standard command to itself and responds.

Master Config:
Configure it to communicate with 2 slave units, ID's 5 and 12.

Slave Unit 26 (Virtual Slave Unit 12):
This unit will be addressed by the master as 12. It is actually configured with an ID of 26. Configure it as a normal slave with an ID of 26.

Slave Unit 5:

This unit will be the repeater. It has an ID of its own as 5, so set its ID as 5. Any reads or writes the master requests from Unit 5 will come directly to and from this unit. If the unit attached to the VikingScada software is capable of repeater operation (and a slave) a ‘Repeater Mode’ tab is visible to set the repeater configuration. In this example the repeater source of 12 needs to be set to target 26. Change the target from None to 26 and write the configuration. All other units (in this example) should remain as None.



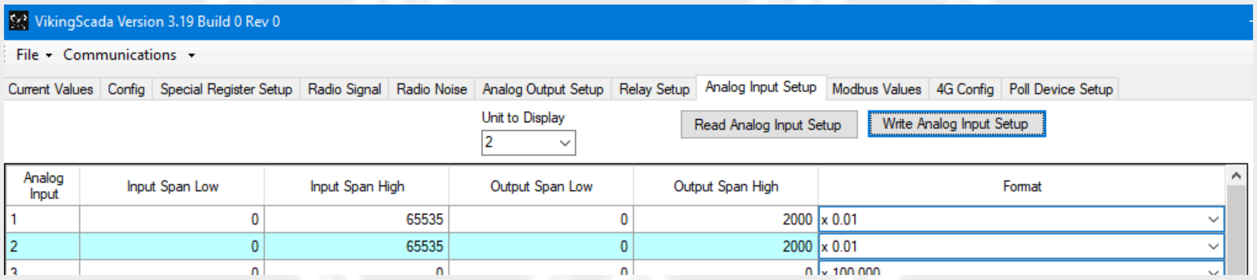
Repeater Source	Repeater Target
1	None
2	None
3	None
4	None
5	None
6	None
7	None
8	None
9	None
10	None
11	None
12	26
13	None

Registers 9200 - 9454

Scaled Slave Analog Input

Scaled Slave Analog Input values can be read from the master when the scale values for each 4G slave are configured in the master.

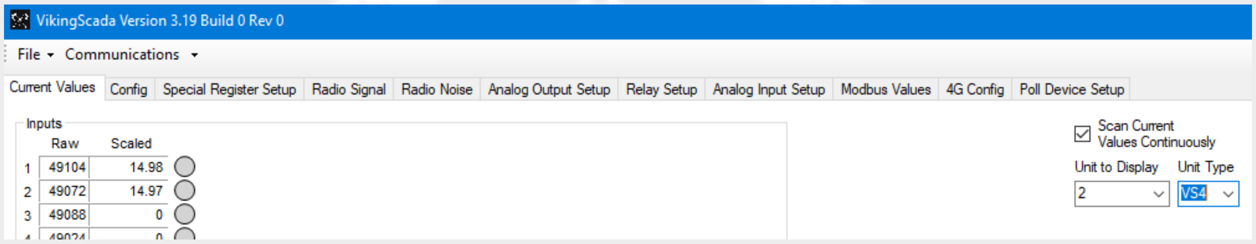
A master unit receives only raw analog inputs from each of its 4G slaves. The analog input scale for each 4G slave can be configured in the master so scaled values can be observed without manual calculation. This configuration exists only in the master and is not read from the slaves nor sent to the slaves. The Analog Input scaling is set for slaves the same way as for the master, but the slave ID indicates which slave is to be scaled. The example below is showing and configuring scaling for slave ID 2.



Unit to Display: 2

Analog Input	Input Span Low	Input Span High	Output Span Low	Output Span High	Format
1	0	65535	0	2000	x 0.01
2	0	65535	0	2000	x 0.01
3	0	0	0	0	x 100.000

These scaling values are used only in the master and not sent to the slaves. The results of the scaled slave inputs can be seen on the Current Values tab or in Modbus registers 15385 - 16376 (768 - 799 for the master's scaled inputs).



Inputs

	Raw	Scaled
1	49104	14.98
2	49072	14.97
3	49088	0
4	49024	0

☑ Scan Current Values Continuously

Unit to Display: 2 Unit Type: VS4

Radio Signal (RSSI) During Communications

The VikingScada LoRa radios measure the Received Signal Strength Indicator (RSSI) for each packet that they receive. The latest value as a percentage of full strength is saved for each unit the device received. The Viking Scada units automatically detect whether an RSSI value is provided with each received packet, so no special configuration is needed. Devices that are connected using a direct serial connection show the RSSI as N/A to indicate that packets are being received but no RSSI is applicable.

Radio Interference (RSSI) for a Channel

Each channel can be scanned for potential radio interference before selecting a channel for a network. The radio module can be put into a channel scanning mode where the Received Signal Strength Indicator (RSSI) is measured around 50 times per second. An algorithm is used to calculate an approximate level of interference (as a percentage) seen on each channel. The highest interference percentage is saved for each channel to help find channels clear of potential interference. The values are cleared by writing a 0 to them. When the scanning mode is enabled, no communications are possible to/from the scanning unit.

Test Registers

Several registers are provided for diagnostic or test purposes.

The firmware operates in an infinite loop that cycles very quickly. A register that measures the performance of this loop (Address 858) shows how many times the loop cycles every second, roughly 6000. This can be a useful diagnostic to verify the RTU is operating as expected.

Sometimes it is useful to use an automatically varying Modbus value as the source of an Analog Output or Special Register to verify correct operation of the configured RTU logic or to check the operation of a separate device. The test registers include values that ramp up or down as follows:

- Slowly increase from 0 to 65535 over about 10 seconds, then wraps back to start at 0 again (Address 852)
- Slowly decrease from 65535 to 0 over about 10 seconds , then wraps back to start at 65535 again (Address 853)
- Very slowly increase from 0 to 65535 over about 2 minutes, then wraps back to start at 0 again (Address 854)
- Very slowly decrease from 65535 to 0 over about 2 minutes, then wraps back to start at 65535 again (Address 855)
- Very quickly increase from 0 to 65535 over about 1 second, then wraps back to start at 0 again (Address 856)
- Very quickly decrease from 65535 to 0 over about 1 second, then wraps back to start at 65535 again (Address 857)

Modbus Map

All registers may be read with Modbus Command 3 or Command 4. The address of 30001 for Command 4, 40001 for Command 3 is the same as 00000. The RTU does not translate the address it receives. It is necessary to use addresses starting at 0000. The registers may also be written using commands 6 (write single register) or 16 (write multiple registers). Note that the RTU will process a maximum of 32 registers in a single command. Registers 0 - 1999 and 15000 - 16663 are volatile and may change back to other values. Registers 2000 - 14999 are non-volatile and are saved across power failures.

Warning

The RTU stores configuration settings in non volatile memory which has a finite lifetime, limited by the number of writes (typically 10,000 to 100,000 writes). Software which writes to the non-volatile RTU configuration registers (2000 – 14999) should prevent continuous writes to the RTU memory. See the Sticky Registers section for more details.

Below is the map of all Modbus Registers. For brevity, many registers are part of a sequential list and are described only once. The beginning and ending of these register lists are noted explicitly with ellipses (...) in between indicating a repeating pattern in between. Any descriptions are only included at the start of the list but apply to each element in the list. For example there may be 256 polled devices so a related group of registers is repeated 256 times in the full map but only registers for polled devices 1, 2, and 256 explicitly shown.

Address	Value	Notes
0	Polled Device 1 Value - high 16 bits	Both hi/lo used for a 32-bit device. Just low 16 bits used for a 16-bit device. Start of volatile RAM register section 1
1	Polled Device 1 Value - low 16 bits	
2	Polled Device 2 Value - high 16 bits	
3	Polled Device 2 Value - low 16 bits	
…	…	
510	Polled Device 256 Value - High 16 bits	
511	Polled Device 256 Value - Low 16 bits	
512	Polled Device 1 Scaled Value 16 bits	POLLSCALED. Made by (raw * mult) / divisor
513	Polled Device 2 Scaled Value 16 bits	
…	…	
767	Polled Device 256 Scaled Value 16 bits	
768	Analog In 1 scaled value 16 bits	ANINSCALED
769	Analog In 2 scaled value as above	
…	…	
799	Analog In 32 scaled value as above	
800	Analog In1 un-scaled raw 16-bit value	Analog In Raw from this unit. ANIN contains all polled units analog ins
801	Analog In2 un-scaled raw 16-bit value	
…	…	
831	Analog In32 un-scaled raw 16-bit value	
832	Analog Output 1 current value, 16-bit integer	ANOUT1VAL
833	Analog Output 2 current value, 16-bit integer	
…	…	
839	Analog Output 8 current value, 16-bit integer	
840	Digital 1 to 16 inputs packed as a 16-bit integer. Lsb = Digital in 1	DIGINIMAGE1
841	Digital 17 to 32 inputs packed as a 16-bit integer. Lsb = Digital in 17	DIGINIMAGE2

842	Image of relays 1 to 16. The current state of the relay outputs 1 to 16 packed as a 16-bit integer. bit0 = relay 1, bit15 = relay 16	RELAYIMAGE
843	Reserved	
…	…	
851	Reserved	
852	Test register. Slow ramp up over several seconds	TESTREG
853	Test register. Slow ramp down over several seconds	
854	Test register. Slow ramp up over several minutes.	
855	Test register. Slow ramp down over several minutes.	
856	Test register. Quickly ramps up.	
857	Test register. Quickly ramps down.	
858	Performance Monitor lower 16 bits	PERFORM Count of main loop excursions per second
859	Comm state summary of all enabled devices.	COMMSTATEGLOBAL. Comm state of all devices. 0 = no response, 1= some OK, 2 = All OK
860	Packed Comm state of devices 1 to 16	COMMSTATE e.g. 0x000E indicates devices 2,3
861	Packed Comm state of devices 17 to 32	
862	Packed Comm state of devices 33 to 48	
863	Packed Comm state of devices 49 to 64	
864	Packed Comm state of devices 65 to 80	
865	Packed Comm state of devices 81 to 96	
866	Packed Comm state of devices 97 to 112	
867	Packed Comm state of devices 113 to 128	
868	Packed Comm state of devices 129 to 144	
869	Packed Comm state of devices 145 to 160	
870	Packed Comm state of devices 161 to 176	
871	Packed Comm state of devices 177 to 192	
872	Packed Comm state of devices 193 to 208	
873	Packed Comm state of devices 209 to 224	
874	Packed Comm state of devices 225 to 240	eg 0x8001 indicates devices 225 and 240 are responding.
875	Packed Comm state of devices 241 to 256	
876	Reserved	
…	…	
881	Reserved	
882	Factory lock. Write various values, nothing stored.	FACTORYLOCK Write FACTORYKEY to unlock locked regs, such as S/N
883	Special register 1	Special register results. These volatile registers 1 to 32 contain the results generated by the special control routines (see regs 8280-8471).
884	Special register 2	COMMSTATE e.g. 0x000E indicates devices 2,3
…	…	
914	Special register 32	

915	Special register 33	Volatile registers available for general use; eg for remote control of relays or analog outputs. Registers 33 to 64 are not assigned to special control routines, but they will always contain 0 upon RTU power up. Registers 33 to 40 may be written to by Toggle() functions.
916	Special register 34	May be written to by Toggle 1 function.
917	Special register 35	May be written to by Toggle 2 function.
918	Special register 36	May be written to by Toggle 2 function.
919	Special register 37	May be written to by Toggle 3 function.
920	Special register 38	May be written to by Toggle 3 function.
921	Special register 39	May be written to by Toggle 4 function.
922	Special register 40	May be written to by Toggle 4 function.
923	Special register 41	
…	…	
946	Special register 64	
947	4G-commanded digital outputs 1-16	DIGCOMMAND1, Written to if we are RFScada slave
948	4G-commanded digital outputs 17-32	DIGCOMMAND2, Written to if we are RFScada slave
949	4G-commanded analog output 1	RFSLAVEANALOG1, Written to if we are RFScada slave
950	4G-commanded analog output 2	
…	…	
956	4G-commanded analog output 8	
957	Reflect input state in RFScada classic mode	RF4GOPTIONS
958	Analog Unit 0 In 1 un-scaled raw 16-bit value	Raw Analog Input Table. Start Unit 0, Input 1 ANIN 32*32 = 1024 length. Table only used if polling any 4G devices
959	Analog Unit 0 In 2 un-scaled raw 16-bit value	
…	…	
989	Analog Unit 0 In 32 un-scaled raw 16-bit value	
990	Analog Unit 1 In 1 un-scaled raw 16-bit value	
991	Analog Unit 1 In 2 un-scaled raw 16-bit value	
…	…	
1021	Analog Unit 1 In 32 un-scaled raw 16-bit value	
…	…	
1980	Analog Unit 31 In 31 un-scaled raw 16-bit value	
1981	Analog Unit 31 In 32 un-scaled raw 16-bit value	Raw Analog Input Table End
1982	Image of relays 17 to 32	RELAY1732IMAGE
1983	Reserved	
1984	Reserved	
1985	Viking Scada type	VS2 = 102, VS4 = 104
1986	Spare RAM	
…	…	
1999	Spare RAM	End of volatile RAM register section 1

2000	Polled Device 1 Baud	Polled Device Config Table. Start of non-volatile EEPROM register section (saved across power cycles).
2001	Polled Device 1 Modbus ID	
2002	Polled Device 1 Modbus Register Addresss	
2003	Polled Device 1 Modbus Function Code	3 or 4
2004	Polled Device 1 Pre-poll time	Milliseconds
2005	Polled Device 1 Post-poll time	Milliseconds
2006	Polled Device 1 Poll type	0 = Modbus 16-bit 1 = Modbus 32-bit H/L 2 = Modbus 32-bit L/H 3 = 4G 8-analogs 4 = 4G 16-analogs 5 = 4G 24-analogs 6 = 4G 32-analogs
2007	Polled Device 1 Dropout timeout	Timeout = value * 10 seconds
2008	Polled Device 1 Default value upper 16 bits	
2009	Polled Device 1 Default value lower 16 bits	
2010	Polled Device 1 Text Name 1	2 ASCII characters
2011	Polled Device 1 Text Name 2	2 ASCII characters
2012	Polled Device 1 Text Name 3	2 ASCII characters
2013	Polled Device 1 Text Name 4	2 ASCII characters
2014	Polled Device 1 Polled value multiplier	
2015	Polled Device 1 Polled value divisor	
2016	Polled Device 1 Modbus 3/4 Count/Index	
2017	Polled Device 1 Spare2	
2018	Polled Device 2 Pre-poll time	
2019	Polled Device 2 Post-poll time	
2020	Polled Device 2 Poll type	
2021	Polled Device 2 Dropout timeout	
2022	Polled Device 2 Default value upper 16 bits	
2023	Polled Device 2 Default value lower 16 bits	
2024	Polled Device 2 Text Name 1	
2025	Polled Device 2 Text Name 2	
2026	Polled Device 2 Text Name 3	
2027	Polled Device 2 Text Name 4	
2028	Polled Device 2 Polled value multiplier	
2029	Polled Device 2 Polled value divisor	
2030	Polled Device 2 Modbus 3/4 Count/Index	
2031	Polled Device 2 Spare2	
…	…	

6590	Polled Device 256 Pre-poll time	
…	…	
6607	Polled Device 256 Spare2	
6608	Analog input 1 display resolution	0 = scaled input * 100000 1 = scaled input * 10000 2 = scaled input * 1000 3 = scaled input * 100 4 = scaled input * 10 5 = scaled input * 1 6 = scaled input * 0.1 7 = scaled input * 0.01 8 = scaled input * 0.001 9 = scaled input * 0.0001 10 = scaled input * 0.00001 11 = scaled input * 0.000001 12 = scaled input * 0.0000001 13 = scaled input * 0.00000001 14 = counted pulses * 1
…	…	
6639	Analog input 32 display resolution	
6640	Analog output 1 span source address	
6641	Analog output 1 span source type	0 = 16-bit source 1 = 32-bit source
6642	Analog output 1 span input high upper 16 bits	
6643	Analog output 1 span input high lower 16 bits	
6644	Analog output 1 span input low upper 16 bits	
6645	Analog output 1 span input low lower 16 bits	
6646	Analog output 1 span output high	
6647	Analog output 1 span output low	
6648	Analog output 2 span source address	
6649	Analog output 2 span source type	
6650	Analog output 2 span input high upper 16 bits	
6651	Analog output 2 span input high lower 16 bits	
6652	Analog output 2 span input low upper 16 bits	
6653	Analog output 2 span input low lower 16 bits	
6654	Analog output 2 span output high	
6655	Analog output 2 span output low	
…	…	
6696	Analog output 32 span source address	
…	…	
6703	Analog output 32 span output low	

6704	Relay 1 source address	If DIGCOMMAND1 or DIGCOMMAND2: Commanded in slave mode If one of ANIN: 4G device input above or below 21000 If any others: register value zero or non-zero
6705	Relay 2 source address	
…	…	
6735	Relay 32 source address	
6736	Reserved	
…	…	
6800	Reserved	
6801	Analog input 1 span input high	
6802	Analog input 1 span input low	
6803	Analog input 1 span output high	
6804	Analog input 1 span output low	
6805	Analog input 2 span input high	
6806	Analog input 2 span input low	
6807	Analog input 2 span output high	
6808	Analog input 2 span output low	
…	…	
6925	Analog input 32 span input high	
6926	Analog input 32 span input low	
6927	Analog input 32 span output high	
6928	Analog input 32 span output low	
6929	Comm1 (System) Mode	
6930	Comm1 (System) Baud	
6931	Comm1 (System) Modbus ID	
6932	Comm1 (System) Gap timeout	
6933	Comm2 (Poll) Mode	
6934	Comm2 (Poll) Baud	
6935	Comm2 (Poll) Modbus ID	
6936	Comm2 (Poll) Gap timeout	
6937	Modbus site name 1	2 ASCII characters
6938	Modbus site name 2	2 ASCII characters
6939	Firmware version	eg. 12 = v1.2
6940	Serial number	
6941	Power on hours	
6942	Reserved	
6943	Reserved	
6944	Comm2 (Poll) Radio channel	
6945	Comm2 (Poll) Comms fail timeout	

6946	Comm2 (Poll) slave poll ID	
6947	Comm2 (Poll) comms fail override	
6948	Reserved	
…	…	
6951	Reserved	
6952	Inputs 1-8 flipped	
6953	Toggle 1 pointer 1	
6954	Toggle 1 pointer 2	
6955	Toggle 2 pointer 1	
6956	Toggle 2 pointer 2	
6957	Toggle 3 pointer 1	
6958	Toggle 3 pointer 2	
6959	Toggle 4 pointer 1	
6960	Toggle 4 pointer 2	
6961	Reserved	
6962	Spare EEPROM	
…	…	
6999	Spare EEPROM	
7000	4G Config device 1 relay 1 source address	
7001	4G Config device 1 relay 2 source address	
…	…	
7031	4G Config device 1 relay 32 source address	
7032	4G Config device 1 analog out 1 source address	
7033	4G Config device 1 analog out 2 source address	
7034	4G Config device 1 analog out 3 source address	
7035	4G Config device 1 analog out 4 source address	
7036	4G Config device 1 analog out 5 source address	
7037	4G Config device 1 analog out 6 source address	
7038	4G Config device 1 analog out 7 source address	
7039	4G Config device 1 analog out 8 source address	
7040	4G Config device 2 relay 1 source address	
7041	4G Config device 2 relay 2 source address	
…	…	
7079	4G Config device 2 analog out 8 source address	
…	…	
8240	4G Config device 32 relay 1 source address	
…	…	
8279	4G Config device 32 analog out 8 source address	
8280	Special register 1 type	

…	…	
8311	Special register 32 type	
8312	Special register 1 source	
…	…	
8343	Special register 32 source	
8344	Special register 1 high threshold upper 16 bits	
8345	Special register 1 high threshold lower 16 bits	
8346	Special register 2 high threshold upper 16 bits	
8347	Special register 2 high threshold lower 16 bits	
…	…	
8406	Special register 32 high threshold lower 16 bits	
8407	Special register 32 high threshold lower 16 bits	
8408	Special register 1 low threshold upper 16 bits	
8409	Special register 1 low threshold lower 16 bits	
8410	Special register 2 low threshold upper 16 bits	
8411	Special register 2 low threshold lower 16 bits	
…	…	
8470	Special register 32 low threshold upper 16 bits	
8471	Special register 32 low threshold lower 16 bits	
8472	Reserved	
8663	Reserved	
8664	Sticky user register 1	
8665	Sticky user register 2	
…	…	
8919	Sticky user register 256	
8920	Spare EEPROM	
…	…	
9199	Spare EEPROM	
9200	Repeater Target for Slave 1	
9201	Repeater Target for Slave 2	
…	…	
9454	Repeater Target for Slave 255	
9455	Slave Ain Input Span High for Slave 1 Input 1	
9456	Slave Ain Input Span Low for Slave 1 Input 1	
9457	Slave Ain Output Span High for Slave 1 Input 1	
9458	Slave Ain Output Span Low for Slave 1 Input 1	
9459	Slave Ain Input Span High for Slave 1 Input 2	
9460	Slave Ain Input Span Low for Slave 1 Input 2	
9461	Slave Ain Output Span High for Slave 1 Input 2	

9462	Slave Ain Output Span Low for Slave 1 Input 2	
…	…	
9579	Slave Ain Input Span High for Slave 1 Input 32	
9580	Slave Ain Input Span Low for Slave 1 Input 32	
9581	Slave Ain Output Span High for Slave 1 Input 32	
9582	Slave Ain Output Span Low for Slave 1 Input 32	
9583	Slave Ain Input Span High for Slave 2 Input 1	
9584	Slave Ain Input Span Low for Slave 2 Input 1	
9585	Slave Ain Output Span High for Slave 2 Input 1	
9586	Slave Ain Output Span Low for Slave 2 Input 1	
9587	Slave Ain Input Span High for Slave 2 Input 2	
9588	Slave Ain Input Span Low for Slave 2 Input 2	
9589	Slave Ain Output Span High for Slave 2 Input 2	
9590	Slave Ain Output Span Low for Slave 2 Input 2	
…	…	
13419	Slave Ain Input Span High for Slave 31 Input 32	
13420	Slave Ain Input Span Low for Slave 31 Input 32	
13421	Slave Ain Output Span High for Slave 31 Input 32	
13422	Slave Ain Output Span Low for Slave 31 Input 32	
13423	Slave Ain Display Precision for Slave 1 Input 1	
13424	Slave Ain Display Precision for Slave 1 Input 2	
…	…	
13454	Slave Ain Display Precision for Slave 1 Input 32	
13455	Slave Ain Display Precision for Slave 2 Input 1	
13456	Slave Ain Display Precision for Slave 2 Input 2	
…	…	
14414	Slave Ain Display Precision for Slave 31 Input 32	
14415	Special Reg 1 Trip On Delay	
14416	Special Reg 2 Trip On Delay	
…	…	
14446	Special Reg 32 Trip On Delay	
14447	Special Reg 1 Trip Off Delay	
14448	Special Reg 2 Trip Off Delay	
…	…	
14478	Special Reg 32 Trip Off Delay	
15000	RSSI Poll Count Total	
15001	RSSI Poll Count Slave 1	
…	…	
15256	RSSI Poll Count Slave 256	

15257	Spare RAM 1	
…	…	
15384	Spare RAM 128	
15385	Scaled Slave Ain for Slave 1 Input 1	These are live values are in RAM (lost after power cycle)
15386	Scaled Slave Ain for Slave 1 Input 2	
…	…	
15416	Scaled Slave Ain for Slave 1 Input 32	
15417	Scaled Slave Ain for Slave 2 Input 1	
15418	Scaled Slave Ain for Slave 2 Input 2	
…	…	
16376	Scaled Slave Ain for Slave 31 Input 32	
16377	LoRa RSSI for Master	
16378	LoRa RSSI Percent for Slave 1	0 = No unit seen, 1 = Unit seen without RSSI, 2-100 = RSSI percent
16379	LoRa RSSI Percent for Slave 2	
…	…	
16632	LoRa RSSI Percent for Slave 256	
16633	LoRa Channel Scan RSSI Enable	
16634	LoRa Channel Scan RSSI Percent Ch 0	
16635	LoRa Channel Scan RSSI Percent Ch 1	
…	…	
16660	LoRa Channel Scan RSSI Percent Ch 26	
16661	LoRa Channel Scan Low Ch	
16662	LoRa Channel Scan High Ch	
16663	LoRa Channel Scan Current Ch	