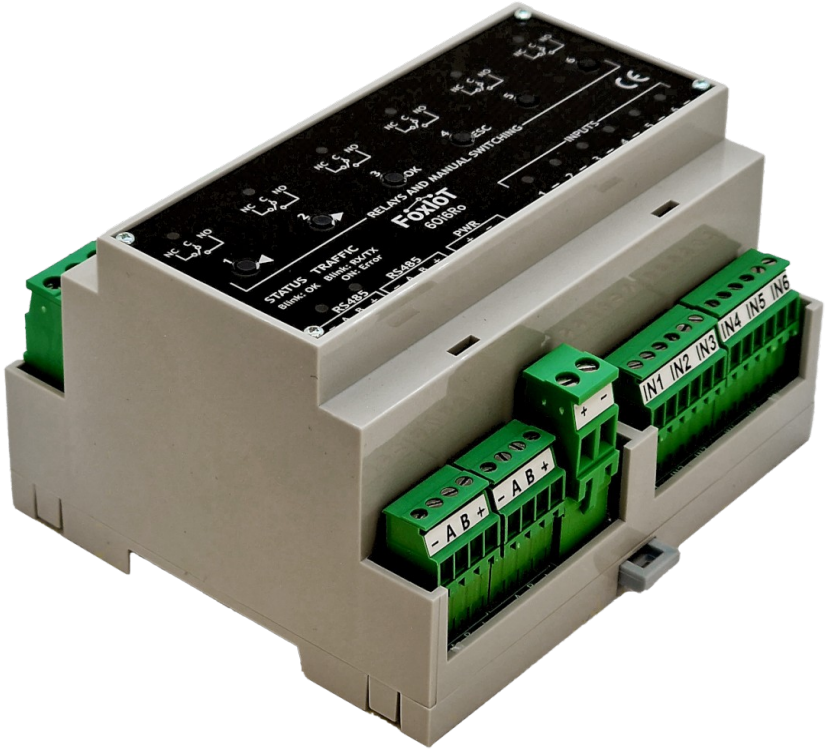
FoxIoT 6Oi6Ro Modbus Integration Guide

Register Map, Rule Configuration, Firmware Update, and Example Packets



FoxIoT OÜ | www.foxiot.eu | support@foxiot.eu

Software Version 1.05
Document Version: 1.0

Modbus Integration Guide

This guide provides detailed information for integrating the FoxIoT 6Oi6Ro controller with Modbus-based systems. It includes a complete register map with descriptions, data types, default values, scaling factors, and access levels.

In addition, the guide outlines how to configure Modbus settings, use the built-in rule engine via Modbus, perform firmware updates, and construct Modbus RTU request/response examples for reading and writing registers.

For general information about the Modbus protocol, visit the official Modbus website:
(http://www.modbus.org)

Data Types Description

Data Type	Description
INT16	16-bit signed integer
UINT16	16-bit unsigned integer.
UINT32	32-bit unsigned integer.
UINT64	64-bit unsigned integer.
FLOAT	32-bit floating-point number.

Access Level Description

Access Level	Description
RO (Read-Only)	This access level indicates that the register can only be read and not modified.
RW (Read-Write)	Registers with this access level can be both read and modified by the user.
W (Write-Only)	Registers with this access level can only be write by the user.

Holding Registers and Applicable Function Codes

All registers listed in the table below are holding registers.

Function Code	Description
Function Code 03 (0x03)	Read Holding Registers. This function code is used to read the contents of one or more holding registers.
Function Code 06 (0x06)	Write Single Register. This function code is used to write a value into a single holding register.
Function Code 16 (0x10)	Write Multiple Registers. This function code allows writing values to multiple holding registers in a single command.

Understanding the Modbus Register Map

Column name	Description
Register Address	The starting Modbus register offset. For example, address 0 corresponds to register 40001 in Modbus convention. (Note: The register numbers in this document refer to Modbus addresses, which begin at 0.)
Register Type	Indicates the type of Modbus register (e.g., Holding Register).
Name/Label	A human-readable identifier describing the data point (e.g., Serial Number, Input Voltage).
Data Type	Specifies how the data is encoded (e.g., INT16, UINT16, UINT32, UINT64, or FLOAT). The data type also determines how many 16-bit registers are used to represent the value.
Access Level	Indicates whether the register is Read-Only (RO) or Read/Write (RW).
Default Value	The initial value set at factory or during device boot-up, where applicable.
Value Range	The valid range of values. Useful for validation.
Units of Measure	Specifies the physical unit of the value (e.g., Volts, Seconds).
Scaling Factor	Multiplier applied to the raw register value to obtain the actual value (e.g., a factor of 0.01 means the register value must be multiplied by 0.01)
Size	Number of 16-bit Modbus registers used for this data item.
Description	Provides additional details for understanding or using the register value.

Register Address	Register Type	Name/Label	Data Type	Access Level	Default Value	Value Range	Units of Measure	Scaling Factor	Size	Description
Readable Registers										
0	Holding	Serial Number	UINT64	RO	-	-	-	1	4	Device's unique identifier
4	Holding	Software Version	UINT16	RO	-	-	-	1	1	Major.Minor format; MSB=Major, LSB=Minor. E.g., 0x0104 = v1.4
5	Holding	Hardware Version	UINT16	RO	-	-	-	1	1	Major.Minor format; MSB=Major, LSB=Minor. E.g., 0x0101 = v1.1
6	Holding	Platform Number	UINT16	RO	34	-	-	1	1	Unique identifier (34 for this product) for distinguishing among different platforms/products within the same family. Used for compatibility checks.
8	Holding	Uptime	UINT32	RO	-	0 to 4294967295	Seconds	1	2	Controller uptime since last boot in seconds.
10	Holding	Input Voltage	FLOAT	RO	-	0 to 32.0	Volts	1	2	Input Voltage in Volts, float representation.
12	Holding	Input Voltage	UINT16	RO	-	0 to 3200	Volts	0.01	1	Input Voltage in Volts, UINT16 scaled by 0.01 for decimal representation.
48	Holding	Input 1 - Digital	UINT16	RO	-	0 to 1	-	1	1	Represents the state of Input 1. See Note *1 for active/inactive state description
49	Holding	Input 2 - Digital	UINT16	RO	-	0 to 1	-	1	1	Represents the state of Input 2. See Note *1 for active/inactive state description.
50	Holding	Input 3 - Digital	UINT16	RO	-	0 to 1	-	1	1	Represents the state of Input 3. See Note *1 for active/inactive state description.
51	Holding	Input 4 - Digital	UINT16	RO	-	0 to 1	-	1	1	Represents the state of Input 4. See Note *1 for active/inactive state description.
52	Holding	Input 5 - Digital	UINT16	RO	-	0 to 1	-	1	1	Represents the state of Input 5. See Note *1 for active/inactive state description.
53	Holding	Input 6 - Digital	UINT16	RO	-	0 to 1	-	1	1	Represents the state of Input 6. See Note *1 for active/inactive state description
54	Holding	Input 1 - Digital Customized	UINT16	RO	-	-	-	1	1	Complex custom functionality, including click types, with real-time input tracking to ensure all changes are captured. See Note *2 for a detailed explanation and examples
55	Holding	Input 2 - Digital Customized	UINT16	RO	-	-	-	1	1	Complex custom functionality, including click types, with real-time input tracking to ensure all changes are captured. See Note *2 for a detailed explanation and examples.
56	Holding	Input 3 - Digital Customized	UINT16	RO	-	-	-	1	1	Complex custom functionality, including click types, with real-time input tracking to ensure all changes are captured. See Note *2 for a detailed explanation and examples.
57	Holding	Input 4 - Digital Customized	UINT16	RO	-	-	-	1	1	Complex custom functionality, including click types, with real-time input tracking to ensure all changes are captured. See Note *2 for a detailed explanation and examples.
58	Holding	Input 5 - Digital Customized	UINT16	RO	-	-	-	1	1	Complex custom functionality, including click types, with real-time input tracking to ensure all changes are captured. See Note *2 for a detailed explanation and examples.
59	Holding	Input 6 - Digital Customized	UINT16	RO	-	-	-	1	1	Complex custom functionality, including click types, with real-time input tracking to ensure all changes are captured. See Note *2 for a detailed explanation and examples.
Readable & Writable Registers										
128	Holding	Boot Flag	UINT16	RW	1	0 to 1	-	1	1	This flag is set to 1 when the system starts and may need to be reset to 0 by the user for custom input logic. See Note *2 for a detailed explanation and examples.
129	Holding	Relay 1	UINT16	RW	0	0 to 1	-	1	1	Controls Relay 1: 0 for OFF, 1 for ON.
130	Holding	Relay 2	UINT16	RW	0	0 to 1	-	1	1	Controls Relay 2: 0 for OFF, 1 for ON.
131	Holding	Relay 3	UINT16	RW	0	0 to 1	-	1	1	Controls Relay 3: 0 for OFF, 1 for ON.
132	Holding	Relay 4	UINT16	RW	0	0 to 1	-	1	1	Controls Relay 4: 0 for OFF, 1 for ON.
133	Holding	Relay 5	UINT16	RW	0	0 to 1	-	1	1	Controls Relay 5: 0 for OFF, 1 for ON.
134	Holding	Relay 6	UINT16	RW	0	0 to 1	-	1	1	Controls Relay 6: 0 for OFF, 1 for ON.
135	Holding	Relay Timer Register 1	UINT16	RW	0	1 to 65536	Seconds	1	1	Controls Relay 1 by time: - Turns the relay ON for a specified period. - Example: Writing 300 turns the relay on for 300 seconds.
136	Holding	Relay Timer Register 2	UINT16	RW	0	1 to 65536	Seconds	1	1	Controls Relay 2 by time: - Turns the relay ON for a specified period. - Example: Writing 300 turns the relay on for 300 seconds.
137	Holding	Relay Timer Register 3	UINT16	RW	0	1 to 65536	Seconds	1	1	Controls Relay 3 by time: - Turns the relay ON for a specified period. - Example: Writing 300 turns the relay on for 300 seconds.

Register Address	Register Type	Name/Label	Data Type	Access Level	Default Value	Value Range	Units of Measure	Scaling Factor	Size	Description
138	Holding	Relay Timer Register 4	UINT16	RW		1 to 65536	Seconds	1	1	Controls Relay 4 by time: - Turns the relay ON for a specified period. - Example: Writing 300 turns the relay on for 300 seconds.
139	Holding	Relay Timer Register 5	UINT16	RW	0	1 to 65536	Seconds	1	1	Controls Relay 5 by time: - Turns the relay ON for a specified period. - Example: Writing 300 turns the relay on for 300 seconds.
140	Holding	Relay Timer Register 6	UINT16	RW	0	1 to 65536	Seconds	1	1	Controls Relay 6 by time: - Turns the relay ON for a specified period. - Example: Writing 300 turns the relay on for 300 seconds.
191	Holding	Reboot	UINT16	RW	0	0 to 65536		1	1	Reboot the controller Writing any non-zero value triggers an immediate reboot.
192	Holding	Rule0_RW0		RW				1	1	Runtime register 0 for Rule 0 See 'Rule Definitions' sheet for details.
193	Holding	Rule0_RW1		RW				1	1	Runtime register 1 for Rule 0 See 'Rule Definitions' sheet for details.
194	Holding	Rule0_RW2		RW				1	1	Runtime register 2 for Rule 0 See 'Rule Definitions' sheet for details.
195	Holding	Rule0_RW3		RW				1	1	Runtime register 3 for Rule 0 See 'Rule Definitions' sheet for details.
		RuleX_RWn								Remaining rules' runtime registers follow the same 4 register layout. Address = 192 + (X × 4 + n) where X = rule number (0–7), n = register index (0-3). A total of 8 rules are supported. See 'Rule Definitions' sheet for details.
Configuration Registers										
256	Holding	Modbus Address	UINT16	RW	1	1 to 247	-	1	1	Sets the Modbus network address of the controller.
257	Holding	Baud Rate	UINT16	RW	6	1 to 10	-	1	1	Sets the communication baud rate. Options: 1 - (300 bps) 2 - (600 bps) 3 - (1200 bps) 4 - (2400 bps) 5 - (4800 bps) 6 - (9600 bps) 7 - (19200 bps) 8 - (38400 bps) 9 - (57600 bps) 10 - (115200 bps)
258	Holding	Character Framing	UINT16	RW	1	1 to 4	-	1	1	Configures Modbus character framing: **1** - (8E1): 8 bits, Even parity, 1 stop bit. **2** - (8N2): 8 bits, No parity, 2 stop bits. **3** - (8N1): 8 bits, No parity, 1 stop bit. **4** - (8O1): 8 bits, Odd parity, 1 stop bit.
264	Holding	Relay Coil Voltage Reduction	UINT16	RW	1	0 to 1	-	1	1	Configures the operational mode of Relay Coil Voltage Reduction: 0 - Coil reduction OFF (Relay coil operates at full voltage) 1 - Coil reduction ON (Reduces relay coil voltage after activation to save power)

Register Address	Register Type	Name/Label	Data Type	Access Level	Default Value	Value Range	Units of Measure	Scaling Factor	Size	Description
265	Holding	Buttons Mode	UINT16	RW	1	0 to 3	-	1	1	Configures the operational mode of interface buttons: **0** - Disabled **1** - Enabled (each button toggles its corresponding relay) **2** - Offline Mode (each button toggles its corresponding relay if no Master query for 60s) **3** - Relay Disable Mode - When activated via button, the corresponding relay turns off, and its LED starts flashing. - The relay cannot be operated via Modbus while in this state. - Pressing the button again restores the relay to the state defined in the relay register. - Note: After a controller reboot, any active relay disablements will be cleared.
Rule Configuration Registers										
1000	Holding	Rule0_Type	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1001	Holding	Rule0_Param0	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1002	Holding	Rule0_Param1	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1003	Holding	Rule0_Param2	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1004	Holding	Rule0_Param3	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1005	Holding	Rule0_Param4	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1006	Holding	Rule0_Param5	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1007	Holding	Rule0_Param6	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1008	Holding	Rule0_Param7	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
1009	Holding	Rule0_Param8	UINT16	RW	0	0 to 65536	-	1	1	See 'Rule Definitions' sheet
		RuleX_Type / RuleX_ParamN	UINT16	RW	0	0 to 65536	-	1	1	**Remaining rules follow same structure every 10 registers. Address = 1000 + (X × 10) where X = rule number (0–7).** **A total of 8 rules are supported.**
Rule Configuration Registers										
9000	Holding	FW_Block_Write	BYTE[130]	W			-	1	65	See 'Firmware Update' sheet

Note *1: **Active/Inactive State Description for Digital Inputs**
- Active State: Minimum guaranteed activation voltage is above 5 V.

Note *2: **Custom Input Logic:**
Bit 0: Input State
- Represents the current real-time physical state of the input (0 = inactive, 1 = active).

Bits 1-4: Activation Counter
- Increments each time the input transitions to the active state.

Bit 5: Input Change Flag
- Set to 1 for 30 seconds following any change in input state, indicating recent activity.

Bit 6: Boot State Indicator
- Reflects the boot register (reg 128) state (1 = device has rebooted).

Bits 7-9: Single Click Counter
- Increments each time a single click is identified. The system evaluates the input after a 350ms window to determine if a single click occurred within this period.

Bits 10-12: Double Click Counter
- Increases for two quick presses: the first within 350ms, followed by a second in another 350ms window. Double-click registered if both fit this timing

Bits 13-15: Long Push Counter
- Increments when the input is held in the active state for longer than 2 seconds, indicating a long press.

Internal Rule Engine Configuration

Rule Type Summary

Rule Type	Name	Description
0	Disabled	Rule is not active
1	Toggle Relay Rule	Toggles relay via corresponding input
2	Flexible Switch Control Rule	Bitmask-based input/relay control

⚠ Configuration Note (Important)

When configuring or modifying a rule, always write to the RuleX_Type register last.
- Writing a non-zero value to RuleX_Type activates the rule
- Writing 0 disables the rule
- Parameter changes are only applied if the rule is disabled before modification and re-enabled afterward

Rule Type 1 - Toggle Relay Rule

This rule links selected input channels to their corresponding relays (e.g., input 1 toggles relay 1), with control logic defined by switch type.

Configuration Parameters

Parameter Index	Name	Description
Type		Must be set to **1**
Param0	Input Enable Bitmask	Each bit enables an input (bit 0 = input 1, bit 1 = input 2, etc.)
Param1	Switch Type	0 = Toggle switch, 1 = Push button

Example: Toggle Relay Rule (Rule Slot 0)

Use case: Toggling a wall switch connected to Input 1 changes the state of Relay 1 (like a standard light switch).

Register	Name	Value	Description
1000	Rule0_Type	1	Toggle Relay Rule
1001	Rule0_Param0	1	Enable Input 1 (bit 0)
1002	Rule0_Param1	0	Toggle switch mode

Behavior:

Relay 1 switches ON or OFF each time the wall switch is toggled.

Rule Type 2 – Flexible Switch Control Rule

This rule enables multiple inputs to control one or more relays, with independent configuration for switch type and optional timer-based reset.

Configuration Parameters

Parameter Index	Name	Description
Type		Must be set to **2**
Param0	Toggle Input Bitmask	Each bit enables a toggle input (bit 0 = input 1, bit 1 = input 2, etc.)
Param1	Push Button Input Bitmask	Each bit enables a push-button input (bit 0 = input 1, bit 1 = input 2, etc.)
Param2	Relay Output Bitmask	Each bit defines a controlled relay (bit 0 = relay 1, bit 1 = relay 2, etc.)
Param3	Countdown Time	In seconds. If zero, relays toggle OFF on second press. If non-zero, relays turn OFF automatically after time elapses. **If a new input activation occurs while the countdown is active, the timer restarts from the original value.**

Example: Flexible Switch Control (Rule Slot 1)

Use case: Two wall switches (connected to Input 2 and Input 3) control Relay 2.

Each switch acts as a toggle. When the relay turns ON, it automatically turns OFF after 60 seconds.

Register	Name	Value	Description
1010	Rule1_Type	2	Toggle Relay Rule
1011	Rule1_Param0	6	Inputs 2 & 3 as toggles (bitmask: bits 1 and 2)

1012	Rule1_Param1	0	No push buttons
1013	Rule1_Param2	2	Relay 2 (bit 1)
1014	Rule1_Param3	60	Countdown 60s

Behavior:

Toggling either wall switch changes the state of Relay 2. If the relay is turned ON, a 60-second countdown starts. If a new switch activation occurs during the countdown, the timer restarts from 60 seconds. Once the countdown ends, Relay 2 turns OFF automatically.

Firmware Update via Modbus RTU

Firmware updates are performed by sending 128-byte blocks to a dedicated Modbus register.
Each block must be exactly 130 bytes:
- 2 bytes for the block offset (high byte + low byte)
- 128 bytes of firmware data

The block offset defines the write position of the firmware data in flash memory. Offset 0 means the first 128 bytes of the firmware file, offset 1 means the second 128 bytes, and so on.

The controller writes each block to flash as it is received. Once the final block is received, the controller will reboot automatically after a 5-second delay.

Address	Name	Description	Data Type	Access
9000	FW_Block_Write	Firmware update block write interface	BYTE[130]	Write-only

Block Format

Byte Index	Purpose
0	Offset High Byte (block offset >> 8)
1	Offset Low Byte (block offset & 0xFF)
2-129	128 bytes of firmware data

⚠ Important Notes

- Each firmware block must be exactly 130 bytes in total.
- If the final firmware block is smaller than 128 bytes, it must be padded with 0x00 or any other value.
- CRC validation is handled by the Modbus protocol itself; no additional checksum is needed.
- The controller will automatically reboot 5 seconds after receiving the final block.

This process ensures a reliable and compact mechanism to perform firmware upgrades in the field using the standard Modbus RTU protocol.

Practical Examples of Modbus RTU Communications

Here are some practical examples of Modbus RTU communications, detailing the packet structure necessary for reading from and writing to the FoxIoT controller's registers. These illustrate how to construct the packets for:

- Reading the state of a digital input from Input 2 (UINT16).
- Setting the state of Relay 1 and Relay 2.
- Setting the Modbus address of the controller.

Each Modbus RTU packet includes a device address, function code, data payload (which encompasses register addresses and values), and a Cyclic Redundancy Check (CRC) for error checking.

Example 1: Reading Digital Input from Input 2

To retrieve the digital state of input 2, we use the following Modbus RTU packet:

Request:
This request packet in bytes is:
01 03 00 31 00 01 D5 C5
- Device Address: **01** (The Modbus device address)
- Function Code: **03** (Function code for reading holding registers)
- Register Address: **0031** (The address of the digital input register, 49 in decimal)
- Quantity of Registers: **0001** (Number of registers to read)
- CRC: **D5C5** (The error-checking CRC value)

Response:
This response packet in bytes is:
01 03 02 00 01 79 84
- Device Address: **01** (The Modbus device address)
- Function Code: **03** (Function code for reading holding registers)
- Byte Count: **02** (Number of data bytes to follow)
- Data: **0001** (The data retrieved from the input register)
- CRC: **7984** (The error-checking CRC value)

The response data 0x0001 from Register 49 (0x31) indicates that input 2 is in an active state.

Example 2: Controlling Relay 1 and Relay 2 States

To simultaneously control the state of Relay 1 and Relay 2, we use the following Modbus RTU packet:

Request:
This request packet in bytes is:
01 10 00 81 00 02 04 00 01 00 00 6B C3
- Device Address: **01** (The Modbus device address)
- Function Code: **10** (Function code for writing multiple holding registers)
- Register Address: **0081** (The starting address for the operation, corresponding to Relay 1's register, which is 129 in decimal)
- Quantity of Registers: **0002** (Indicates that two registers will be written to, covering Relay 1 and Relay 2)
- Byte Count: **04** (Indicates 4 bytes of data for 2 registers (2 bytes per register))
- Data for Relay 1: **0001** (Data to set the state of Relay 1. In this example, 01 sets Relay 1 to the ON state)
- Data for Relay 2: **0000** - (Data to set the state of Relay 2. In this example, 00 sets Relay 2 to the OFF state)
- CRC: **6BC3** (The error-checking CRC value)

Response:
This response packet in bytes is:
01 10 00 81 00 02 11 E0
- Device Address: **01** (Identifies the Modbus device address)
- Function Code: **10** (Confirms the operation was to write to multiple holding registers)
- Register Address: **0081** (Echoes the starting address from the request, corresponding to Relay 1's register)
- Quantity of registers: **0002 (**Confirms that two registers were targeted, as per the request, affecting Relay 1 and Relay 2**)**
- CRC: **11E0** (The error-checking CRC value)

The response packet confirms the successful execution of the command to write to multiple holding registers, ensuring that Relay 1 and Relay 2 are set as requested.