ISAFE GATEWAY

Complete Technical Architecture & Modbus Manual | Firmware 1.0-20260126a1

1. Modbus Register Map

The device memory is strictly divided into two blocks. The **40000 Block** is purely for reading live data. The **41000 Block** is purely for writing new configuration.

READ-ONLY (LIVE)	READ/WRITE (SETUP)	PARAMETER NAME	VALUE FORMAT & EXPLANATION
40001	**41001**	**Slave ID**	`1` to `247`. Sets the Modbus RTU Slave Identity. Default is `1`.
40002	**41002**	**Baud Rate**	`0` =9600, `1` =19200, `2` =38400, `3` =57600, `4` =115200 (Default)
40003	**41003**	**Stop Bits**	`1` or `2`. Default is `1`.
40004	**41004**	**Parity**	`0` =None (Default), `1` =Even, `2` =Odd
40005	**41005**	**Live Status**	`0` = Inactive, `1` = Active
40006 - 40009	*Read Only*	**Station IP Address**	4 Registers. Live IP provided by the connected router. E.g., `192`, `168`, `1`, `100`.
40010	*Read Only*	**Station Port**	Default Modbus TCP port is `502`.
40011 - 40014	*Read Only*	**AP IP Address**	4 Registers. The IP of the module's own broadcast network. Default is `192.168.4.1`.
40015	*Read Only*	**AP Port**	Default Modbus TCP port is `502`.
40016 - 40021	*Read Only*	**MAC Address**	6 Registers. Exact hardware bytes (e.g., `236`, `218`, `59` ...)
40022 - 40036	**41022 - 41036**	**Client SSID**	15 Registers. 1 ASCII letter per register. Maximum 15 characters.

READ-ONLY (LIVE)	READ/WRITE (SETUP)	PARAMETER NAME	VALUE FORMAT & EXPLANATION
40037 - 40051	**41037 - 41051**	**Client Password**	15 Registers. 1 ASCII letter per register. Maximum 15 characters.
40052	**41052**	**WiFi Mode**	`0` =OFF, `1` =AP, `2` =Client, `3` =Both (Default)
40053	**41053**	**Retry Interval**	Seconds between connection attempts. Default is `600` .
40054	**41054**	**Retry Limit**	Max attempts before 1-hour pause. Default is `6` .
40055 - 40069	**41055 - 41069**	**AP SSID**	15 Registers. Default: `iSAFE-[Last 4 MAC]`
40070 - 40084	**41070 - 41084**	**AP Password**	15 Registers. Default: `isafe@dm1n`
40085 - 40099	**41085 - 41099**	**Web Login Pass**	15 Registers. Default: `isafe@[Last 4 MAC]`
40100	**41100**	**Data Bits**	`7` or `8` . Default is `8` .
40101 - 40109	**41101 - 41109**	**Reserved Buffer**	9 empty registers kept at `0` .
Read Only	**41110**	**Commit Trigger**	**Write `1` to this register to commit all changes to flash memory and reboot!**

> **String (Text) Data Encoding:** To prevent Endianness (byte-swapping) issues between different PLCs and Master software, the iSAFE firmware stores exactly **ONE character per 16-bit register**. E.g., if Register 40022 reads `105`, that is the ASCII code for the letter 'i'. Unused spaces are populated with `0`.

2. System Logic & Architecture Breakdown

This section explains the core C++ algorithms that make the iSAFE gateway industrial-grade, crash-proof, and highly reliable.

1. Dual-Modbus Mirroring Engine

The device initializes two completely separate Modbus libraries: `ModbusRTU` (assigned to Serial pins 16/17) and `ModbusTCP` (assigned to WiFi Port 502). By default, these two systems do not talk to each other. To solve this, the firmware uses a **Mirroring Callback System**.

Every writable register (41001-41100) has an attached callback function. If a Master writes to the RTU register, the callback instantly grabs that value and silently pushes it to the TCP register bank (and vice versa). A boolean flag (`isMirroring`) ensures an infinite loop does not occur while they sync.

2. The "Commit Trigger" Staging System

Writing directly to flash memory on an ESP32 takes time. If a Modbus master tries to write 15 characters of an SSID sequentially and the device saves to flash after every single letter, the device will crash or timeout.

To solve this, the `41000` block acts strictly as RAM (Random Access Memory). The Master can write data as slowly or as quickly as it wants without interrupting the device. The data is only converted, checked, and burned into the permanent NVS Flash memory when the Master writes a `1` to register **41110**. This acts as a hardware "Save & Reboot" switch.

3. Non-Blocking WiFi State Machine

Standard Arduino WiFi code uses a `while(WiFi.status() != WL_CONNECTED)` loop. In an industrial environment, if the router turns off, this standard loop will freeze the entire processor, meaning the Fire Relay and Modbus RS485 will stop working.

The iSAFE firmware uses a **Switch-Case State Machine** (`wState`) attached to a `millis()` timer. It checks the WiFi status for just 1 microsecond during each loop. If the connection fails, it increments a counter. If it hits the retry limit (e.g., 6 times), it switches to a `WS_LONG_WAIT` state and completely suspends WiFi connection attempts for exactly 1 hour. During this entire process, the core loop continues running at maximum speed, ensuring the Fire Relay can trigger instantly at all times.

4. Auto-Healing Memory & Random MAC Fallback

If a power surge interrupts a flash-write cycle, the ESP32 memory can become corrupted. If it reads a Parity value of "999", the serial port will fail to boot.

The `loadConfig()` function includes an **Auto-Repair Module**. As soon as variables are pulled from flash, they are clamped to mathematical limits (e.g., if Parity > 2, force it to 0). Furthermore, if the physical WiFi antenna is damaged and returns a MAC address of `00:00:00:00:00:00`, the system will use an internal randomizer to generate a stable, locally-administered MAC address, save it to flash, and use it to construct the default Passwords so you are never locked out of the device.

5. AP Auto-Timeout Logic

According to the PDF specifications, if the device is in "Both AP and Client" mode (Code 3), the Access Point must disable itself to save power and security bandwidth after 20 minutes.

The system records `apStartMillis` at boot. The main loop constantly compares the current time against this marker. Once the difference exceeds `1,200,000 milliseconds` (20 minutes), the firmware executes `WiFi.softAPdisconnect(true)`, killing the broadcast network while leaving the Client network flawlessly intact.

6. Manual Ignite Override & Relay Control

The relay logic is executed on every single loop iteration via a highly optimized ternary operator:
`digitalWrite(PIN_RELAY, (digitalRead(PIN_SENSOR) == HIGH || manualAlarm) ? HIGH : LOW);`

This means the Fire Relay will instantly activate if the physical sensor pin goes HIGH **OR** if the virtual `manualAlarm` variable is triggered via the Web UI "Ignite Alarm" button. The relay cannot be blocked by any web loading or Modbus querying.