

IS4350: I2C Modbus TCP/IP Server

65,500 Holding Registers

Applications

- Industrial Sensors and Transmitters (Temperature, Pressure, Flow, Level)
- Actuator and Motor Controllers
- Remote I/O Modules
- Energy Meters and Power Monitors
- Building Automation and HVAC Controllers
- Data Acquisition and Test Equipment

Main Advantages

- No TCP/IP or Modbus Stack to Develop, Test, or Maintain
- Simple Integration: Standard I2C Reads and Writes, No Libraries Required
- Lower Microcontroller Load: No Ethernet ISRs, Timers, or Stack Memory
- Only Two Microcontroller Pins (SDA and SCL)

Modbus and I2C Characteristics

- 65,500 Holding Registers available
- Implemented Function Codes:
 - 03 – Read Holding Registers
 - 06 – Write Single Register
 - 16 – Write Multiple Registers
- I2C speeds: 100 kHz, 400 kHz, and 1 MHz
- 3 configurable I2C addresses: 24, 25, 26

Safety

- IS4350 can be updated with the latest security patches
- Optional I2C Watchdog Timer (I2C-WDT)
- Optional I2C CRC Protection (I2C-CRC)
- Fail-Safe Power-Up

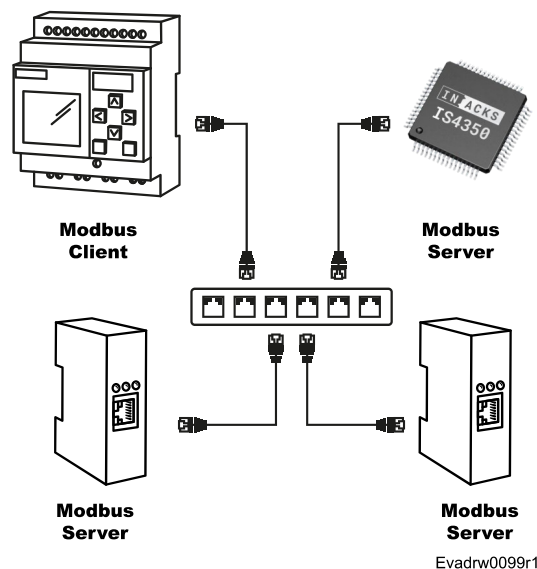
General Description

The IS4350 is an integrated circuit that implements a Modbus TCP/IP server stack. It provides 65,500 Holding Registers of internal memory for application use, accessible both by your microcontroller via I2C and by any Modbus client (PLC, computer, etc.) over the network. The IS4350 interfaces with Ethernet through an external LAN8742A PHY transceiver.

The IS4350 reduces the engineering time and cost associated with implementing and testing Ethernet and Modbus TCP/IP communication, offering a reliable, ready-to-use solution that shortens product time-to-market. It requires no specialized libraries; communication is handled using standard I2C read and write commands.

The IS4350 internal firmware can be updated to keep it current with the latest security patches, and it implements safety mechanisms such as I2C CRC protection (I2C-CRC), an I2C watchdog timer (I2C-WDT), and fail-safe power-up.

The device operates at 3.3 V and is housed in an LQFP64 (10×10 mm) package.



1 Ordering Information

Standard configuration

The standard configuration refers to the device variant that is mass-produced and offered at the most cost-efficient price.

Orderable Device	Status	Package		Pins	Temperature Range
IS4350-L64I www.inacks.com/is4350	ACTIVE		LQFP64 (10×10 mm)	64	I (Industrial) -40 °C to 85 °C
IS4350-M1I www.inacks.com/is4350-m1	ACTIVE		Module (20×62 mm)	36 + Ethernet Connector	I (Industrial) -40 °C to 85 °C

Status Description:

ACTIVE: Product recommended for new designs.

SPECIAL: Available only for high-volume pre-orders. Contact sales.

NRND: Device is in production to support existing customers, but it is not recommended for use in new designs.

OBSOLETE: Production of the device has been discontinued.

2 Revision History

This datasheet and the IS4350 may be revised without notice. Users are responsible for obtaining and reviewing the latest version of this document before purchasing the product or designing it into any application. The most recent datasheet revision available on the company's website (www.inacks.com) shall be considered the only valid and authoritative source of product information.

2.1 Document Revision

Date	Revision Code	Description
September 2026	ISDOC139C	- Initial Release
June 2026	ISDOC139B	- Preliminary Review
June 2026	ISDOC139A	- Preliminary Release

2.2 Firmware Revision

Firmware Revision can be found in the `FW_REV` register in the memory map.

Date	Revision Code	Description
June 2026	0	- Preliminary Release

3 Specifications

Absolute Maximum Ratings

Parameter			Min	Max	Unit
Input Voltage	VDD, SPD, ADR, NRST, RMII_TXEN, MAC_SDA, MAC_SCL pins		-0.3	4	V
	I2C_CRC, SCL, SDA, DEF, INT, PROV_RX, PROV_TX, RMII_MDC, RMII_REFCLK, RMII_MDIO, RMII_CRS, RMII_RXD0, RMII_RXD1, RMII_TXD0, RMII_TXD1 pins		-0.3	6	
Current Sourced/Sunk by any I/O and Control Pin			—	±20	mA
Total output current Sourced/Sunk by sum of all I/Os and control pins			—	±140	
Temperature	Operating Temperature	IS4350-L64I (Industrial)	-40	+85	°C
	Maximum junction temperature	T _J	—	+105	
	Storage Temperature		-65	+150	
Electrostatic Discharge (TA = 25 °C)	Human-body model (HBM), Class 2		—	2000	V
	Charged-device model (CDM), Class C2a		—	500	

Exceeding the specifications outlined in the Absolute Maximum Ratings could potentially lead to irreversible harm to the device. It's important to note that these ratings solely indicate stress limits and don't guarantee the device's functionality under such conditions, or any others not specified in the Recommended Operating Conditions. Prolonged exposure to conditions at or beyond the absolute maximum ratings might compromise the reliability of the device.

Recommended Operating Conditions

Parameter	Pins	Symbol	Min	Nom	Max	Unit
Supply Voltage	VDD	V _{DD}	2.7	3.3	3.6	V
I/O Input Voltage	SCL, SDA, I2C_CRC, DEF, INT, PROV_RX, PROV_TX, RMII_MDC, RMII_REFCLK, RMII_MDIO, RMII_CRS, RMII_RXD0, RMII_RXD1, RMII_TXD0, RMII_TXD1	V _{I/O-IN}	-0.3	3.3	5.5	
	SPD, ADR		-0.3	1.65, 3.3	3.6	
	RMII_TXEN, NRST, MAC_SCL, MAC_SDA		-0.3	3.3	3.6	
Source/Sink Current	Any I/O Pins	I _{I/O-SS}	—	—	±8	mA

Electrical Characteristics

All I/Os are CMOS and TTL compliant.

Parameter		Symbol	Min	Nom	Max	Unit
Current Consumption	T _J = 25 °C	I _{DD}	37	—	90	mA
	T _J = 85 °C		65	—	118	
Input Voltage	Logical High-Level	V _{IH}	0.7 x V _{DD}	—	—	V
	Logical Low-Level	V _{IL}	—	—	0.3 x V _{DD}	

Electrical Specifications Revision A

Flash Memory Cell Characteristics

This memory stores all the non-volatile memory map registers and the IS4350 program itself.

Parameter		Min	Nom	Max	Unit
Write Cycles ¹		10,000	—	—	Cycles
Data Retention ¹	T _A = 85 °C	30	—	—	Years
Write Time		—	2.8	—	ms

¹ This information is evaluated by characterization, not tested in production.

Modbus Characteristics

Parameter	Characteristics
Implemented Function Codes	• FC03 – Read Holding Registers • FC06 – Write Single Register • FC16 – Write Multiple Registers
Available Holding Registers	65,500
Modbus Unit Identifier	0xFF (Responds to any, as defined in the standard)
Simultaneous Client Connections	1

Modbus Performance

Parameter		Min	Nom	Max	Unit
Requests per Second ²	FC03 – Read Holding Registers	Qty of Registers = 1	—	267	Hz
		Qty of Registers = 60	—	261	
		Qty of Registers = 125	—	248	
	FC06 – Write Single Register	Qty of Registers = 1	—	270	
		Qty of Registers = 1	—	259	
		Qty of Registers = 60	—	241	
	FC16 – Write Multiple Registers	Qty of Registers = 1	—	259	Hz
		Qty of Registers = 60	—	241	
		Qty of Registers = 123	—	216	

² These results depend significantly on network performance and the performance of the Modbus client. Therefore, the values shown should be considered a guide and are not deterministic, as they may be affected by external factors beyond the control of the IS4350. Actual performance may be better or worse than the values described here. These tests were performed on a low-performance office network using a Python-based Modbus client running on a computer with an Intel Core i5 processor.

Ethernet Characteristics

Parameter	Characteristics
Ethernet PHY	Requires an external LAN8742A
Modbus Server IP	DHCP / Static (default: DHCP)
Modbus Server Port	502
Ethernet Speed	10/100 Mbps
Ethernet MAC address configuration	Locally Administered
	Universally Administered

4 Description

What It Is and What It Does

The IS4350 is an integrated circuit operating at 3.3 V that implements a Modbus TCP/IP server stack. It interfaces with your microcontroller via I2C, requiring no specialized libraries, only standard I2C read and write operations. This makes it easy and fast to develop Modbus TCP/IP products.

Utility

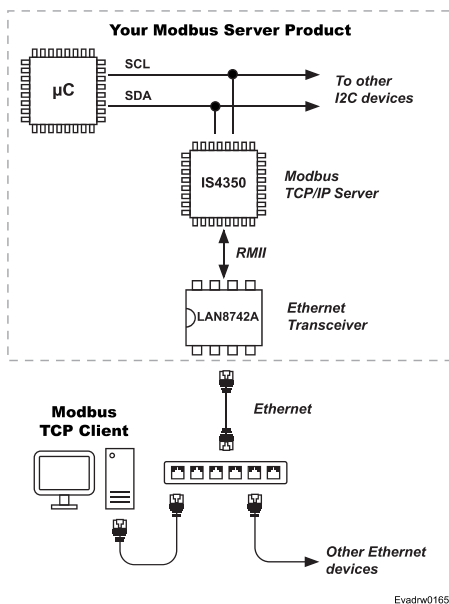
Modbus TCP/IP is one of the most widely used industrial communication protocols due to its simplicity and robustness. It operates over standard Ethernet networks, enabling straightforward and transparent integration, as Ethernet is widely available in industrial and commercial environments.

Benefits

The IS4350 eliminates the need for engineers to invest time in implementing the low-level details of Ethernet communication and the Modbus protocol. This significantly reduces development and validation effort, shortens time to market, and enables faster product release.

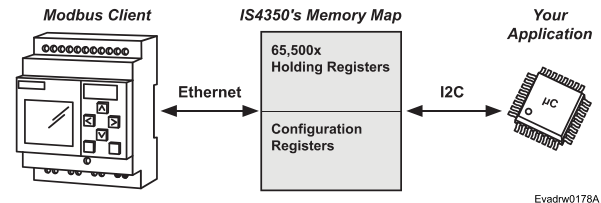
How It Works

The IS4350 features two communication interfaces: an RMII for the LAN8742A transceiver and an I2C-serial interface for your microcontroller.



Once your product is connected to Ethernet and you have started the server via I2C by setting the `GO_ONLINE` register to 1, the Modbus TCP/IP server becomes accessible on the local area network.

The IS4350 provides 65,500 Holding Registers available to be used for your application. These Holding Registers are used for data exchange between the PLC (Modbus client) and your microcontroller (customer application).



In a motor controller application, for example, the PLC can write the desired motor speed to specific Holding Registers and read the motor's power consumption from others. These registers act as a shared memory bank where information is exchanged between the two components of the Modbus communication system: the client and the server.

Essentially, the IS4350 operates as an I2C memory device integrated with the Ethernet and Modbus TCP/IP stack, where the internal memory map is directly exposed as Holding Registers.

Modbus and Ethernet

The IS4350 supports the most commonly used Modbus function codes:

- 03 – Read Holding Registers
- 06 – Write Single Register
- 16 – Write Multiple Registers

These function codes provide read and write access to all 65,500 Holding Registers available for application use.

The Unit Identifier is intended for Modbus Serial Line devices accessed through gateways, routers, or bridges that use a single IP address to represent multiple independent Modbus endpoints. This is not applicable to the IS4350, as it uses a direct TCP/IP connection where the IP address uniquely identifies the device. In this case, the recommended Unit Identifier value is `0xFF`, as defined by the standard.

Once the IS4350 is put online by writing 1 to the `GO_ONLINE` register, it will listen on the official Modbus port, 502.

The IS4350 supports DHCP and static IP configuration; see the `DHCP` register description for more details.

The IS4350 supports multiple MAC address configuration modes, including manual configuration, automatic generation from the internal IC serial number, random generation, and automatic loading from an external EUI-48 EEPROM (AT24MAC402-STUM-T). See the `MAC_CFG` register for more details.

5 Safety

5.1 I2C-CRC Protection

Data integrity on the I2C serial interface, where the IS4350 is connected, is critical because it carries the holding register data exchanged with your microcontroller. This data may include sensor readings that your system collects and makes available to a PLC. Any data corruption may cause a sensor value to be altered, resulting in a completely different measurement being reported.

Similarly, if your system controls an actuator, corrupted data may modify a control value, such as a motor speed command, causing unintended operation. In both cases, data corruption may lead to unsafe system behavior or create a hazardous situation.

To prevent these situations, I2C-CRC protection can be enabled by setting the I2C_CRC pin high. This pin is read at startup only and should not change its value during operation. When enabled, all I2C read and write operations between the microcontroller and the IS4350 are protected by a 16-bit CRC, allowing communication errors to be detected before corrupted data is accepted.

The CRC algorithm used to improve I2C communication reliability is CRC-16/Modbus¹, the same algorithm used in Modbus RTU communications.

There are several ways to calculate this CRC. Listing 1 is a C implementation that does not require lookup tables and can be used as a reference in your application.

From the entire I2C frame, only the data bytes are used for the CRC calculation. The I2C Device Address, Pointer Register High, and Pointer Register Low bytes are excluded.

For faster firmware development, it is recommended to initially disable the I2C-CRC protection feature by setting the I2C_CRC pin low or leaving it floating. This simplifies the validation of I2C communication between your microcontroller and the IS4350. Once basic communication has been verified, the I2C-CRC protection feature can be enabled if required.

Your microcontroller must support I2C clock stretching, as the IS4350 may hold SCL low while it calculates the CRC or performs other internal operations.

```
// This routine calculates and returns the Modbus RTU CRC-16 without lookup tables for I2C communications.
uint16_t CRC16(uint8_t *message, uint16_t messageLength)
{
    // Modbus CRC starts with all bits set.
    uint16_t crc = 0xFFFF;

    // Process each byte in the message.
    while (messageLength--) {
        // XOR the next byte into the low CRC byte.
        crc ^= *message++;

        // Process all eight bits in the byte.
        for (uint8_t bit = 0; bit < 8; bit++) {

            // Shift right; apply the reversed Modbus polynomial when the LSB is set.
            if (crc & 0x0001) {
                crc = (crc >> 1) ^ 0xA001;
            }
            else {
                crc >>= 1;
            }
        }
    }
    return crc;
}
```

Listing 1: Routine for calculating the CRC-16 without lookup tables.

¹ For further details about the algorithm, refer to the Modbus over Serial Line Specification and Implementation Guide V1.02, Appendix B, LRC/CRC Generation.

5.1.1 I2C-CRC for Read Operations

When I2C-CRC protection is enabled your microcontroller can check that the data it reads from the IS4350 was not corrupted on the bus.

The IS4350 doesn't know in advance how many registers your microcontroller will read, so it can't add a CRC to the end of the read frame. Instead, it calculates the CRC of the data bytes it sent during the last I2C read operation and stores the result in the `I2C_READ_CRC` register. Your microcontroller then reads this register in a second read operation and compares the value with the CRC it calculated itself from the data it received.

Validating an I2C read with the I2C-CRC feature activated therefore requires two I2C read operations and four steps. For the following example, assume that your microcontroller needs to read 3 Holding Registers, `HOLD_10`, `HOLD_11`, and `HOLD_12`, which contain:

- `HOLD_10` = 0xAABB
- `HOLD_11` = 0x1234
- `HOLD_12` = 0x0001

1. Read the desired Holding Registers from the IS4350.

Read `HOLD_10`, `HOLD_11`, and `HOLD_12` as you normally would. Only the data bytes (`HOLD_10`, `HOLD_11`, and `HOLD_12`, 6 bytes) are used for the CRC calculation. The I2C Device Address, Pointer Register High, and Pointer Register Low bytes are excluded.

Byte	Value	Used for CRC	Description
1	0x30	No	Device Address 24 with bit R/W = 0
2	0x00	No	Pointer Register Hi
3	0x0A	No	Pointer Register Lo
4	0x31	No	Device Address 24 with bit R/W = 1
5	0xAA	Yes	<code>HOLD_10</code> Hi
6	0xBB	Yes	<code>HOLD_10</code> Lo
7	0x12	Yes	<code>HOLD_11</code> Hi
8	0x34	Yes	<code>HOLD_11</code> Lo
9	0x00	Yes	<code>HOLD_12</code> Hi
10	0x01	Yes	<code>HOLD_12</code> Lo

2. Read the `I2C_READ_CRC` register.

Right after the data read, read the `I2C_READ_CRC` register (address 65524). It contains the CRC that the IS4350 calculated for the data sent in the previous read operation. The CRC is sent most significant byte first (big-endian), the same order used for the register data.

Byte	Value	Description
1	0x30	Device Address 24 with bit R/W = 0
2	0xFF	Pointer Register Hi
3	0xF4	Pointer Register Lo
4	0x31	Device Address 24 with bit R/W = 1
5	0x7C	<code>I2C_READ_CRC</code> Hi
6	0x79	<code>I2C_READ_CRC</code> Lo

3. Calculate the CRC of the received data.

Calculate the CRC-16/Modbus of the data bytes received in step 1. In this example, the CRC is calculated over the bytes 0xAA, 0xBB, 0x12, 0x34, 0x00, 0x01, which gives a CRC of 0x7C79.

4. Compare the two CRC values.

If the CRC calculated by your microcontroller matches the value read from `I2C_READ_CRC`, the data is valid. Otherwise, a data integrity error has been detected in either the data or the CRC. Discard the data and repeat the whole read sequence.

5.1.2 I2C-CRC for Write Operations

When I2C-CRC protection is enabled, every I2C write to the IS4350 must end with a 16-bit CRC calculated over the data being written. The IS4350 calculates the CRC of the data it receives and compares it with the CRC sent by your microcontroller. The written data is accepted only if the two values match.

Validating an I2C write with the I2C-CRC feature activated requires three steps. For the following example, assume that your microcontroller needs to write 3 Holding Registers, HOLD_10, HOLD_11, and HOLD_12, with the following values:

- HOLD_10 = 0xAABB
- HOLD_11 = 0x1234
- HOLD_12 = 0x0001

1. Calculate the CRC of the data to be written.

Only the data bytes (HOLD_10, HOLD_11, and HOLD_12, 6 bytes) are used for the CRC calculation. The I2C Device Address, Pointer Register High, and Pointer Register Low bytes are excluded. In this example, the CRC-16/Modbus is calculated over the bytes 0xAA, 0xBB, 0x12, 0x34, 0x00, 0x01, which gives a CRC of 0x7C79.

2. Add the CRC to the end of the write frame.

Send the CRC as two extra bytes directly after the last data byte. Send the most significant byte first (big-endian), the same order used for the register data. In this example, send 0x7C, then 0x79.

Note: this differs from Modbus RTU, where the CRC is sent least significant byte first.

Byte	Value	Used for CRC	Description
1	0x30	No	Device Address 24 with bit R/W = 0
2	0x00	No	Pointer Register Hi
3	0x0A	No	Pointer Register Lo
4	0xAA	Yes	HOLD_10 Hi
5	0xBB	Yes	HOLD_10 Lo
6	0x12	Yes	HOLD_11 Hi
7	0x34	Yes	HOLD_11 Lo
8	0x00	Yes	HOLD_12 Hi
9	0x01	Yes	HOLD_12 Lo
10	0x7C	No	CRC Hi
11	0x79	No	CRC Lo

3. Execute the write operation.

The IS4350 calculates the CRC of the received data bytes, excluding the last two bytes, and compares it with the received CRC. If the two values match, the data is written to the memory map. Otherwise, a data integrity error has been detected, and the IS4350 discards the entire frame without modifying any register.

The IS4350 checks the CRC only after the Stop Condition, so it acknowledges (ACK) every byte of the frame, including the CRC bytes, even when the CRC is wrong. An I2C ACK therefore does not mean that the data was accepted.

The two CRC bytes are never written to the memory map. In the example above, HOLD_13 is not modified by the CRC bytes that follow HOLD_12 in the frame.

To confirm that the IS4350 accepted the written data, read back the written registers using the I2C-CRC read procedure described in section 5.1.1 I2C-CRC for Read Operations.

5.2 I2C Watchdog Timer (I2C-WDT)

In a typical application, your microcontroller keeps reading and writing the IS4350 Holding Registers. In a sensor, it writes new measurements that Modbus clients read. In a controller, it acts on the values that Modbus clients write.

Now suppose your microcontroller stops working, for example because of a firmware fault. The IS4350 keeps running and the Modbus server stays online, but the Holding Registers are no longer updated. Modbus clients keep reading stale data with no sign that anything is wrong, which may be dangerous.

The I2C watchdog timer prevents this. To enable it, start the Modbus server by writing 2 to the `GO_ONLINE` register

instead of 1. This starts an internal 500 ms watchdog timer. The timer restarts whenever your microcontroller writes 2 to the `GO_ONLINE` register again.

If 500 ms pass without the write, the IS4350 assumes your microcontroller has failed and stops the Modbus server automatically.

You can restart the Modbus server at any time by writing 1 (without the watchdog) or 2 (with the watchdog) to the `GO_ONLINE` register.

Refer to section `GO_ONLINE` Register for more information about I2C-WDT functionality.

5.3 Fail-Safe Power-Up

At power-up, the IS4350 does not start the Modbus server automatically. The `GO_ONLINE` register defaults to 0 (offline), so your microcontroller must start the server on purpose. This way, the server never comes online while your microcontroller is still starting up or has failed.

For example, if an end user damages your product and your microcontroller stops working while the IS4350 still

runs, the Modbus server would otherwise come online with nothing updating it. It could then report stale or invalid data. Because the IS4350 never starts the server on its own, this cannot happen.

Refer to section `GO_ONLINE` Register for more information about how to turn on the Modbus server.

6 Usage

Firmware Design Structure

When using the IS4350, it is important to distinguish between the two types of users of the device:

- Embedded Engineer – The engineer integrating the IS4350 into the end product. This is typically the user of this datasheet.
- Automation Engineer – The engineer integrating the final Modbus device into an industrial system or facility.

This section covers the embedded engineer's use of the IC, specifically the development of the firmware that connects your microcontroller with the IS4350.

Developing firmware for the IS4350 is straightforward. No specific libraries are required; only standard I2C read and write operations are needed, making integration fast and easy.

For example, when developing a Modbus TCP/IP temperature sensor, the integration process consists of the following steps:

1. Write the temperature value to Holding Register 0 (`HOLD_0`) and update it periodically.
2. Enable the Modbus server by writing a 1 via I2C to the register `GO_ONLINE`.

The Modbus server is now online, and the temperature value is accessible to any Modbus client. Continue updating the temperature value as needed.

By default, the IS4350 is configured in DHCP mode; therefore, no Ethernet configuration is required, making it very easy and straightforward to use.

Accessing the IS4350 via Modbus

Use any Modbus TCP/IP client to connect to the IS4350. For a list of recommended Modbus client applications, refer to the IS4350 product page: www.inacks.com/is4350

- Before establishing a connection, verify that the Modbus server is online by checking `STATUS_BITS` = 5, indicating that the Modbus server is ready and listening.
- To determine the IP address of your Modbus server, read the `IP` address registers.
- The Modbus TCP port is 502 (standard Modbus TCP port).
- Once a connection is established, `CLIENT_BIT` will be set to 1. This bit can optionally be used to indicate connection status, for example by driving an LED to show that the device is in use.

Firmware Best Practices

- Leave a few milliseconds between consecutive I2C operations.

This provides the IS4350 with sufficient time to complete the internal processing associated with each I2C transaction.

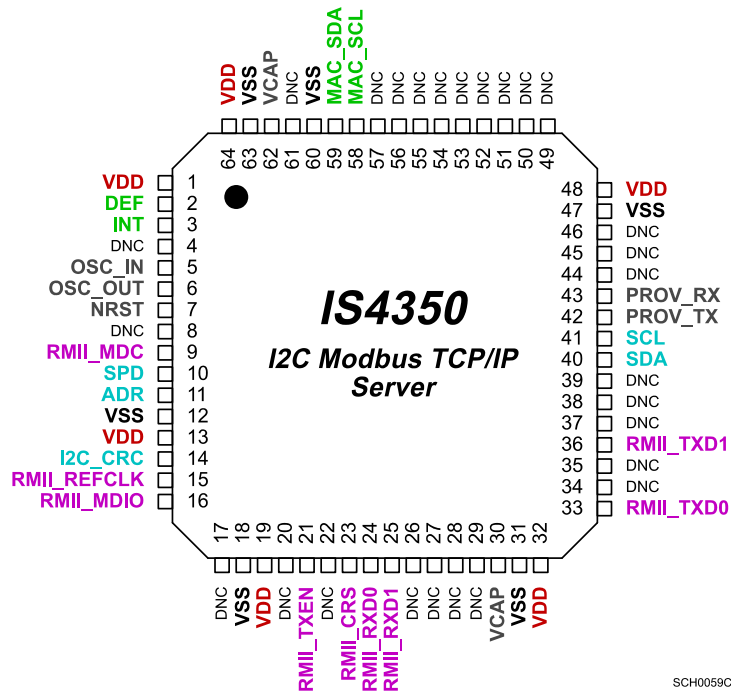
- Before modifying the Ethernet configuration registers (`DHCP`, `IP`, `MASK` and `GATEWAY`), stop the Modbus server by setting `GO_ONLINE` = 0.

This ensures an orderly reconfiguration of the Ethernet settings.

- Update the `HOLD_X` registers data before bringing the Modbus server online.

This ensures that the Holding Registers contain valid data as soon as the server becomes accessible. Otherwise, there may be a brief interval between the server going online and the microcontroller updating the registers, during which a Modbus client could read invalid or uninitialized data.

7 Pins



Pin	Name	Type	Description
Functionalities			
2	DEF	Digital Input	Restore default network configuration. Hold high for 3 s. Optional; internal weak pull-down, leave floating if unused.
3	INT	Digital Output Push-Pull	Goes high for about 100 ms each time a Modbus write (FC06 or FC16) is received. Optional; leave floating if unused.
58	MAC_SCL	Open Drain	This I2C-serial interface is intended to obtain a unique MAC Address (EUI-48) from the AT24MAC402-STUM-T EEPROM. This functionality is optional; leave floating if unused.
59	MAC_SDA		
I2C			
41	SCL	Open Drain	I2C serial interface. These pins connect to your microcontroller. Two external pull-up resistors to 3.3 V or 5 V are required.
40	SDA		
14	I2C_CRC	Digital Input	I2C CRC protection enable. High: enabled. Low or floating: disabled (internal weak pull-down). Read at start-up only.
10	SPD	Analog Input	I2C serial interface speed selection pin. - Connect to VSS for 100 kHz. - Connect to a VDD/2 voltage divider (1.65 V) for 400 kHz. - Connect to VDD (3.3 V) for 1 MHz.
11	ADR		I2C serial interface address selection pin. - Connect to VSS for address 24. - Connect to VDD/2 (1.65 V) for address 25. - Connect to VDD for address 26.

Pin	Name	Type	Description
Ethernet			
9	RMII_MDC	Digital Input Output	Connect this pin to the LAN8742A MDC pin.
15	RMII_REFCLK		Connect this pin to the LAN8742A REFCLK pin.
16	RMII_MDIO		Connect this pin to the LAN8742A MDIO pin.
21	RMII_TXEN		Connect this pin to the LAN8742A TXEN pin.
23	RMII_CRS		Connect this pin to the LAN8742A CRS_DV pin.
24	RMII_RXD0		Connect this pin to the LAN8742A RXD0 pin.
25	RMII_RXD1		Connect this pin to the LAN8742A RXD1 pin.
33	RMII_TXD0		Connect this pin to the LAN8742A TXD0 pin.
36	RMII_TXD1		Connect this pin to the LAN8742A TXD1 pin.
Power			
1, 19, 32, 48	VDD	Supply	3.3 V power supply pins. Each of these pins must be decoupled using a 100 nF ceramic capacitor.
64			3.3 V power supply pin. Decouple this pin using a 100 nF and a 4.7 μF ceramic capacitor.
13			3.3 V power supply pin. Decouple with a 100 nF and a 1 μF ceramic capacitor.
12, 18, 31, 47, 60, 63	VSS	Reference	Voltage reference pin (0 V).
System			
5	OSC_IN	Input	25 MHz external crystal oscillator pins.
6	OSC_OUT	Output	
7	NRST	Digital Input	Active low reset input pin with internal weak pull-up. Connect this pin to the LAN8742A $\overline{\text{NRST}}$ pin to synchronize the reset of both ICs.
30, 62	VCAP	Stabilization	Internal LDO regulator stabilization capacitors Connect pins 30 and 62 together, and connect them to two 2.2 μF ceramic capacitors in parallel, both bypassed to VSS. IMPORTANT: Do not connect 3.3 V to these pins.
43	PROV_RX	Digital Input	Provisioning UART.
42	PROV_TX	Digital Output Push-Pull	
Remaining	DNC	Do Not Connect	Leave these pins floating.

7.1 DEF Pin

Restore Default Values Pin.

This pin allows the Ethernet configuration to be restored to its default values, and restarts the IS4350. If a user configures an incorrect network configuration and is consequently unable to reconnect to the device, the configuration can be recovered by momentarily holding this pin high for 3 seconds (for example, using a push button). When activated, all configuration registers are reset to the factory default values and the device restarts, making it accessible again.

It is recommended to integrate a push button for this function in the final product. To prevent accidental activation, the button should be recessed and not easily accessible.

The implementation of this pin is optional. Leave it floating if not used; it has an internal weak pull-down (~40 k Ω).

When this pin is activated, IS4350 registers are loaded to their default values:

- Server is stopped (`GO_ONLINE = 0`)
- Holding Registers are all set to `0xFFFF`
- DHCP is enabled (`DHCP = 1`) and the `IP`, `MASK` and `GATEWAY` registers will show the corresponding DHCP data.
- MAC address is set to locally administered, with the address generated from the IS4350 serial number (`MAC_CFG = 1`). And `MAC_ADR` will show the corresponding MAC address values.

7.2 INT Pin

Holding Register Data Available Interrupt Pin.

This pin goes momentarily high for approximately 100 ms whenever a Modbus write operation is received. The pulse is generated even if the written value is identical to the value already stored and no actual change occurs.

The purpose of this pin is to generate an interrupt in your microcontroller to indicate that a Modbus write operation has been received, avoiding the need for continuously checking the memory map (polling) and allowing it to access the registers only when a write operation occurs.

The pin asserts when receiving any of the following write function codes:

- Function Code 06 — Write Single Register
- Function Code 16 — Write Multiple Registers

The implementation of this pin is optional. Leave it floating if not used; it has an internal weak pull-down (~40 kΩ).

Note: this pin asserts for every write function code received, even if the written data is identical to the previous value and therefore no actual change occurs in the internal memory map.

7.3 PROV_RX & PROV_TX Pins

Provisioning Pins.

The provisioning UART pins are used for firmware updates. They should be accessible to an external USB-to-UART adapter so that the IS4350's firmware can be kept up to date if any system or security patch is released. Typically, a 3-pin header with 2.45 mm pitch (PROV_RX, PROV_TX, and GND) is used for this connection.

7.4 I2C_CRC Pin

I2C CRC Protection Pin.

This pin enables CRC validation for I2C communications. CRC ensures the integrity of data transmitted and received over the I2C-serial interface, allowing corrupted data caused by bus noise, malfunctioning I2C devices on the bus, or other communication errors to be detected and discarded.

To enable I2C CRC protection, connect this pin to VDD. To operate the I2C interface without CRC protection, leave this pin floating or connect it to VSS. This pin is internally connected to a weak pull-down resistor.

The configuration of this pin is read when the IS4350 starts up and can't be changed during runtime.

Refer to section I2C-CRC Protection for more details.

7.5 SCL & SDA Pins

I2C-Compatible Bus Interface Pins.

SCL (Serial Clock Line): This pin is used to synchronize data transfer between the IS4350 device and your microcontroller or other CPU.

SDA (Serial Data Line): This bidirectional pin is used for both sending and receiving data between the IS4350 and your microcontroller or other CPU.

Both pins are open-drain and must be pulled up to 3.3 V or 5 V. The pull-up resistor value should be chosen based on the bus speed and capacitance. Typical values are 4.7 kΩ for Standard Mode (100 kHz) and 2 kΩ for Fast Mode and Fast Mode Plus (400 kHz and 1 MHz).

7.6 MAC_SCL & MAC_SDA Pins

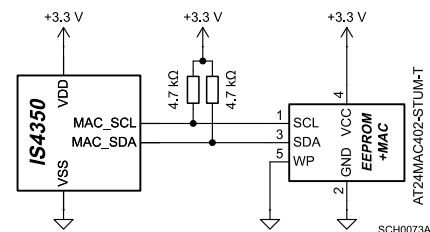
Dedicated I2C-serial interface for an external AT24MAC402-STUM-T EEPROM providing a unique MAC Address.

If you require a unique EUI-48 MAC address for your Modbus TCP/IP device but your production volume is not large enough to acquire a minimum block of MAC addresses, or if you are developing a prototype, you can connect an AT24MAC402-STUM-T EEPROM. This device includes a factory-programmed unique EUI-48 MAC address, allowing your product to obtain a unique MAC address at minimal cost.

The IS4350 automatically detects and reads the MAC address on every power-up cycle. To use the MAC address stored in the external EEPROM, configure the MAC_CFG register to 3 (MAC address obtained from external MAC EEPROM).

Connect the AT24MAC402-STUM-T I2C-serial interface pins to MAC_SCL and MAC_SDA pins using pull-up resistors to 3.3 V. Do not apply 5 V to these pins. Typical values are 4.7 kΩ for Standard Mode (100 kHz) and 2 kΩ for Fast Mode and Fast Mode Plus (400 kHz and 1 MHz).

The implementation of this I2C-serial interface is optional. Leave its pins floating if unused.



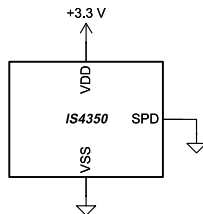
7.7 SPD Pin

I2C Speed Selection Pin.

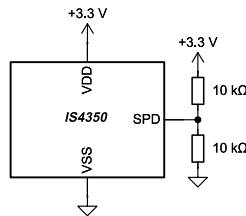
This pin configures the IS4350 internal I2C-serial interface timings and filters to properly work with the selected bus speed. This pin is only checked at power-up.

Do not leave this pin floating.

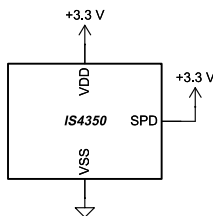
For a **100 kHz** setting, set the SPD pin to VSS.



For a **400 kHz** setting, set the SPD pin to 1.65 V ($V_{DD}/2$) using a balanced voltage divider. This can be achieved by placing two 10 k Ω resistors from the SPD pin: one to VDD and the other to VSS.



For a **1 MHz** setting, set the SPD pin to VDD.



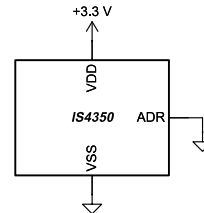
7.8 ADR Pin

I2C Device Address Selection Pin.

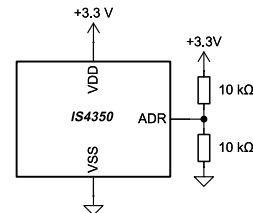
This pin configures the IS4350 internal I2C device address. This pin configuration is only checked at power-up.

Do not leave this pin floating.

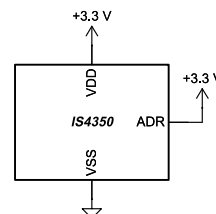
For **I2C device address 24**, set the ADR pin to VSS.



For **I2C device address 25**, set the ADR pin to 1.65 V ($V_{DD}/2$) using a balanced voltage divider. This can be achieved by placing two 10 k Ω resistors from the ADR pin: one to VDD and the other to VSS.



For **I2C device address 26**, set the ADR pin to VDD.



7.9 NRST Pin

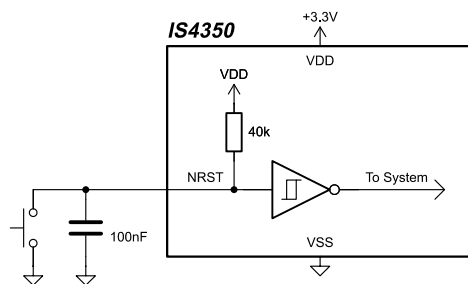
CPU Reset Pin.

When this pin is asserted low, the execution of the IS4350 internal firmware is restarted. All non-volatile data is preserved (IP, MAC, etc.). This is not a factory-default reset; it only restarts the firmware execution.

A 100 nF bypass capacitor between this pin and VSS is sufficient to protect the IS4350 against parasitic resets.

Connect this pin to the LAN8742A nRST pin to synchronize the reset of both ICs.

The pin has an internal weak pull-up resistor. An external pull-up resistor may be connected to provide a stronger pull-up and improve reset signal integrity.



8 Memory Map

The IS4350 memory map registers are 16 bits wide, and I2C memory addressing uses 2-byte addresses.

I2C Register Address		Register Name	Description	
Holding Registers	0	HOLD_0	Holding Register 0	Accessible via I2C and Modbus
	...	...	Holding Register 1 to 65,498	
	65499	HOLD_65499	Holding Register 65,499	
Ethernet Configuration Registers	65500	DHCP	DHCP Configuration: 0: IP address set manually 1: IP address assigned by DHCP	Accessible only via I2C
	65501	IP	These registers show/configure the IP address bytes	
	...			
	65504			
	65505	MASK (Subnet)	These registers show/configure the subnet mask bytes	
	...			
	65508			
	65509	GATEWAY	These registers show/configure the gateway address bytes	
	...			
	65512			
IS4350 Configuration Registers	65513	GO_ONLINE	0: Stop the Modbus server 1: Start the Modbus server 2: Start the Modbus server with Watchdog (I2C-WDT)	Accessible only via I2C
	65514	STATUS	This register indicates the server status. The main status codes are: 1: Server stopped 2: Server starting 3: Waiting for IP confirmation 4: IP assigned, not yet ready 5: Modbus server ready and listening 101: ERROR – MAC address chip not detected 102: ERROR – Configured MAC address is invalid 103: ERROR – Network cable disconnected 104: ERROR – LAN8742A IC not detected Bit 8: A client is connected Bit 9: Same as INT pin; new data is available	
	65515	MAC_CFG	This register selects how the MAC address is generated	
	65516	MAC_ADR	These registers show/configure the MAC address bytes	
	...			
	65521			
	65522	CHIP_ID	IS4350 Chip Identification number (never changes)	
	65523	FW_REV	IS4350 Firmware Revision number (may change)	
	65524	I2C_READ_CRC	CRC of the last I2C read operation	
	65525	Reserved	These registers are reserved for future use and should not be used	
...				
65535				

Table 1: IS4350's Memory Map

8.1 HOLD_X Registers

Modbus Holding Registers.

The `HOLD_X` registers are the Modbus Holding Registers available for use in your application. They can be accessed (read and written) by both the Modbus client and your microcontroller.

If your product is a sensor, these registers are typically written by your microcontroller with sensed data, such as pressure, temperature, or humidity, and read by the Modbus client.

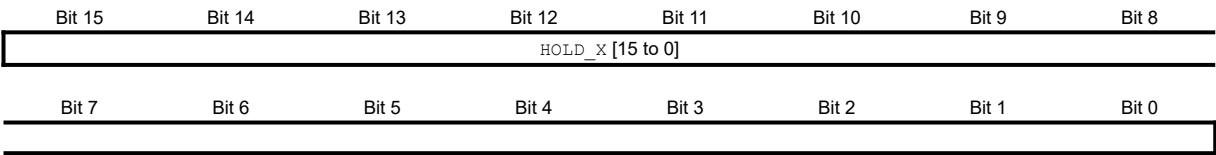
If your product is an actuator, these registers are typically written by the Modbus client with control data, such as relay states (on/off), motor speed, or solenoid valve positions, and read by your microcontroller.

These registers can also serve as a bidirectional data exchange area, allowing both the Modbus client and your microcontroller to read and write data.

It is good practice to update the `HOLD_X` data before bringing the Modbus server online. This ensures that, once the server is set online, the Holding Registers already contain valid data. Otherwise, there may be a short interval after the server goes online and before your microcontroller updates the registers, during which the Modbus client could read invalid or uninitialized data.

A total of 65,500 Modbus Holding Registers are available. They are implemented in volatile RAM and initialized to `0xFFFF` by default.

Name:	HOLD_0 to HOLD_65499
Description:	Modbus Holding Registers
Address Range:	0 to 65499
Default Value:	0xFFFF
Memory Type:	Volatile RAM
Allowed Values:	0x0000 to 0xFFFF
Accessible by:	Modbus and I2C (Read and Write)



8.2 DHCP Register

Enable or disable DHCP (dynamic IP address assignment).

Writing 1 to this register enables DHCP. When DHCP is enabled, the IS4350 automatically obtains and assigns the IP address, subnet mask, and gateway. In this mode, the `IP`, `MASK`, and `GATEWAY` registers are automatically updated with the assigned values, and any write operation to these registers has no effect.

Writing 0 to this register disables DHCP. When DHCP is disabled, the `IP`, `MASK`, and `GATEWAY` registers must be configured manually via Modbus or I2C.

Any modification to this register is saved in internal non-volatile memory. Writing the same DHCP

configuration value that is already stored does not trigger a non-volatile memory write cycle.

To modify this register via I2C, first set the Modbus server offline (`GO_ONLINE = 0`) and wait a few milliseconds to allow the server to stop. The register can then be modified, and the Modbus server can be brought online again.

To modify this register via Modbus, execute Function Code 6 (Write Single Register) or Function Code 16 (Write Multiple Registers). Wait a few milliseconds and restart the device to load the new configuration. Note that the Modbus server must be online when modifying this register via Modbus; otherwise, a Modbus connection cannot be established.

Name:	DHCP
Description:	Use automatically assigned IP
Address:	65500
Default Value:	1 (DHCP)
Memory Type:	Flash
Allowed Values:	0 or 1
Accessible by:	Modbus and I2C (Read always allowed; recommended I2C write operations when <code>GO_ONLINE = 0</code>)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Not Used							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used							DHCP

8.3 IP Registers

The IP address is composed of four registers, each one corresponding to one field of the IPv4 address. To correctly configure the IP address, all four IP registers must be programmed with the desired values.

When the IS4350 operates in DHCP mode (DHCP register set to 1), the IP registers reflect the IP address assigned by the DHCP server.

When DHCP is disabled (DHCP register set to 0), this register must be manually configured with the desired IP address.

Any modification to this register is saved in internal non-volatile memory. Writing the same IP configuration

value that is already stored does not trigger a non-volatile memory write cycle.

To modify this register via I2C, first set the Modbus server offline (`GO_ONLINE = 0`) and wait a few milliseconds to allow the server to stop. The register can then be modified, and the Modbus server can be brought online again.

To modify this register via Modbus, execute Function Code 6 (Write Single Register), or Function Code 16 (Write Multiple Registers). Wait a few milliseconds and restart the device to load the new configuration. Note that the Modbus server must be online when modifying this register via Modbus; otherwise, a Modbus connection cannot be established.

Name:	IP
Description:	Read or set the IPv4 address
Address:	65501, 65502, 65503, 65504
Default Value:	Unknown: assigned by DHCP
Memory Type:	Flash
Allowed Values:	0 to 255
Accessible by:	Modbus and I2C (Read always allowed; recommended I2C write operations when <code>GO_ONLINE = 0</code>)

Register 65501

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	IP Field 1 [xxx.---.---.---]

Register 65502

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	IP Field 2 [---.xxx.---.---]

Register 65503

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	IP Field 3 [---.---.xxx.---]

Register 65504

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	IP Field 4 [---.---.---.xxx]

8.4 MASK Register

The subnet mask is composed of four registers, each one corresponding to one field of the subnet mask. To correctly configure the subnet mask, all four subnet mask registers must be programmed with the desired values.

When the IS4350 operates in DHCP mode (DHCP register set to 1), the MASK registers reflect the subnet mask assigned by the DHCP server.

When DHCP is disabled (DHCP register set to 0), this register must be manually configured with the desired subnet mask.

Any modification to this register is saved in internal non-volatile memory. Writing the same mask

configuration value that is already stored does not trigger a non-volatile memory write cycle.

To modify this register via I2C, first set the Modbus server offline (GO_ONLINE = 0) and wait a few milliseconds to allow the server to stop. The register can then be modified, and the Modbus server can be brought online again.

To modify this register via Modbus, execute Function Code 6 (Write Single Register), or Function Code 16 (Write Multiple Registers). Wait a few milliseconds and restart the device to load the new configuration. Note that the Modbus server must be online when modifying this register via Modbus; otherwise, a Modbus connection cannot be established.

Name:	MASK
Description:	Read or set the subnet mask
Address:	65505, 65506, 65507, 65508
Default Value:	Unknown: assigned by DHCP
Memory Type:	Flash
Allowed Values:	0 to 255
Accessible by:	Modbus and I2C (Read always allowed; recommended I2C write operations when GO_ONLINE = 0)

Register 65505

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Subnet Mask Field 1 [xxx . --- . --- . ---]

Register 65506

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Subnet Mask Field 2 [--- . xxx . --- . ---]

Register 65507

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Subnet Mask Field 3 [--- . --- . xxx . ---]

Register 65508

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Subnet Mask Field 4 [--- . --- . --- . xxx]

8.5 GATEWAY Register

The gateway address is composed of four registers, each corresponding to one field of the IPv4 address. To correctly configure the gateway, all four registers must be programmed with the desired values.

When the IS4350 operates in DHCP mode (DHCP register set to 1), the GATEWAY registers reflect the gateway address assigned by the DHCP server.

When DHCP is disabled (DHCP register set to 0), this register must be manually configured with the desired gateway address.

Any modification to this register is saved in internal non-volatile memory. Writing the same gateway

configuration value that is already stored does not trigger a non-volatile memory write cycle.

To modify this register via I2C, first set the Modbus server offline (GO_ONLINE = 0) and wait a few milliseconds to allow the server to stop. The register can then be modified, and the Modbus server can be brought online again.

To modify this register via Modbus, execute Function Code 6 (Write Single Register), or Function Code 16 (Write Multiple Registers). Wait a few milliseconds and restart the device to load the new configuration. Note that the Modbus server must be online when modifying this register via Modbus; otherwise, a Modbus connection cannot be established.

Name:	GATEWAY
Description:	Read or set the Gateway address
Address:	65509, 65510, 65511, 65512
Default Value:	Unknown: assigned by DHCP
Memory Type:	Flash
Allowed Values:	0 to 255
Accessible by:	Modbus and I2C (Read always allowed; recommended I2C write operations when GO_ONLINE = 0)

Register 65509

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Gateway Field 1 [xxx. ---. ---. ---]

Register 65510

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Gateway Field 2 [---. xxx. ---. ---]

Register 65511

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Gateway Field 3 [---. ---. xxx. ---]

Register 65512

Bit 15 to Bit 8	Bit 7 to Bit 0
Not used	Gateway Field 4 [---. ---. ---. xxx]

8.6 GO_ONLINE Register

Activate or deactivate the Modbus server.

- Writing 0 to this register puts the Modbus server offline, making it inaccessible to Modbus clients.
- Writing 1 to this register puts the Modbus server permanently online (listen state), allowing Modbus clients, such as a PLC or computer, to connect.
- Writing 2 to this register puts the Modbus server online in watchdog mode (I2C-WDT). This increases the reliability of your product, but you need to keep refreshing the watchdog.

Writing 1 or 2 to this register initiates an automatic process that performs all tasks necessary to bring the server online, without requiring any further action on your side. From the moment this value is written until the server is fully online and listening, the process may take anywhere from a few milliseconds to several seconds, or it may fail entirely, for example, due to an incorrect network configuration or network conditions.

The online operation may fail due to incorrect configuration of the DHCP, IP, MASK, GATEWAY, MAC_CFG, or MAC_ADR registers, a disconnected network cable, or other network causes.

Writing 1 or 2 to this register therefore does not guarantee that the server will successfully enter the online state. The STATUS register should be used to verify the actual state of the server.

Watchdog

Writing a 1 to this register starts the Modbus server without the watchdog feature. The server will then remain permanently online, even if your microcontroller fails during operation and can no longer update the server with valid values — a condition that may be dangerous.

To prevent this, write a 2 to this register to start the Modbus server with the watchdog feature enabled. The internal 500 ms watchdog timer then begins counting down. It is reset by writing 2 to this register again. If the timer is not reset within 500 ms, the IS4350 automatically stops the server. The server can be restarted by writing 1 or 2 to this register.

Note: Once this register is set to 2, the watchdog timer is activated and starts counting regardless of the connection status. For example, it may take 5 seconds from the moment the value 2 is written until the Modbus server comes online. During these 5 seconds, while the server is not yet online, the watchdog timer must still be updated.

Fail-Safe Register Default Value

For safety, the default value of this register is 0 (offline) at power-up. This ensures the Modbus server does not come online automatically if your microcontroller is not yet ready or has failed.

Name:	GO_ONLINE
Description:	Activate or deactivate the Modbus server.
Address:	65513
Default Value:	0 (Offline)
Memory Type:	Volatile RAM
Allowed Values:	0, 1, or 2
Accessible by:	I2C (Read and Write)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Not Used							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used						GO_ONLINE	

8.7 STATUS Register

This register provides information about the Modbus server status.

Bits 0 to 7 – STATUS_BITS

Indicates the current server status. Possible values are:

- **1: Server stopped** – The server is stopped because the `GO_ONLINE` register is set to 0.
- **2: Server starting** – The server is initializing and preparing to go online after the `GO_ONLINE` register has been set to 1 or 2. No action is required; you should wait for the server to advance to the next state.
- **3: Waiting for IP confirmation** – The server is waiting for IP address validation. No action is required; you should wait for the server to advance to the next state.
- **4: IP assigned, not yet ready** – An IP address has been assigned, but the server is not yet ready to accept connections. No action is required; you should wait for the server to advance to the next state.
- **5: Modbus server ready and listening** – The server is online and ready to accept Modbus client connections.
- **101: ERROR** – MAC address chip not detected.

- **102: ERROR** – Configured MAC address is invalid
- **103: ERROR** – Network cable disconnected.
- **104: ERROR** – LAN8742A IC not detected.

Bit 8 – CLIENT_BIT

This bit is automatically set to 1 when a Modbus client connects to the IS4350 and is automatically cleared when the client disconnects.

Bit 9 – INT_BIT

This bit is automatically set to 1 whenever a Modbus write function is received (FC06 – Write Single Register or FC16 – Write Multiple Registers). The bit must be cleared manually by your microcontroller.

The `INT_BIT` is set whenever a write function is received, regardless of whether the new value differs from the previous value. Therefore, the bit may be set even if the Holding Registers remain unchanged.

By monitoring this bit, your microcontroller can determine whether any Modbus write operation has occurred without continuously reading the entire memory map and checking for changes. This reduces I2C traffic and processing time.

The `INT` pin provides similar functionality, but generates a short pulse instead of latching a status bit. This pulse is intended to trigger an interrupt on your microcontroller.

Name: STATUS
Description: Provide information about the Modbus server.
Address: 65514
Default Value: 1
Memory Type: Volatile RAM
Accessible by: I2C (Read and Write)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Not Used						INT_BIT	CLIENT_BIT
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STATUS_BITS[7 to 0]							

8.8 MAC_CFG Register

MAC Address Configuration Register.

This register configures the IS4350's MAC address (EUI-48), which can be either a Universally Administered Address or a Locally Administered Address.

In a Universally Administered Address, the MAC is assigned by the IEEE and is guaranteed to be globally unique. In a Locally Administered Address, the MAC is set either by you or by the IS4350.

The following list explains in detail how the MAC address is generated according to the value set in the `MAC_CFG` register.

- **1: Automatic Fixed Locally Administered Address** – The MAC address is generated using an algorithm based on the IS4350's serial number. This does not guarantee uniqueness. The address remains the same across power-up cycles or resets. This is the default value.
- **2: Manual Locally Administered Address** – The MAC address is set manually by the user by writing to the `MAC_ADR` registers. When using this

configuration, the MAC address must be a valid locally administered unicast address.

- **3: Automatic Universally Administered Address** – The MAC address is obtained from an external EEPROM containing a globally unique IEEE-assigned address. The supported EEPROM is the AT24MAC402-STUM-T. When this configuration is selected, the IS4350 automatically connects to the EEPROM via I2C using the `MAC_SCL` and `MAC_SDA` pins and retrieves the address. Refer to the `MAC_SCL` & `MAC_SDA` Pins section for more details.
- **4: Automatic Random Locally Administered Address** – The MAC address is randomly generated on every IS4350 power-up cycle or reset.

Any modification to this register is stored in internal non-volatile memory and does not change the active MAC address until the next power-up cycle or reset. Writing the same value as previously stored does not trigger a non-volatile memory write cycle.

Name: `MAC_CFG`
Description: MAC Address Configuration
Address: 65515
Default Value: 1 (MAC address derived from the IS4350's serial number)
Memory Type: Flash
Allowed Values: 1 to 4
Accessible by: I2C (Read and Write)
Note: Changes to this register are stored immediately but do not take effect until the next power-up cycle or reset.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Not Used							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used					MAC_CFG		

8.9 MAC_ADR Registers

MAC Address Value Registers.

The IS4350 MAC address (EUI-48) is composed of six registers, each corresponding to one field of the address.

The behavior of the MAC_ADR registers depends on the value set in the MAC_CFG register.

- If **MAC_CFG = 1, 3 or 4**: The MAC address is automatically assigned by the IS4350 itself or by an external MAC address EEPROM. In this case, the MAC_ADR registers reflect the assigned MAC address and are read-only. Writing to these registers has no effect.
- If **MAC_CFG = 2** (manual configuration): The MAC address is manually configured by writing the desired MAC address to the MAC_ADR registers. Two conditions must be met:
 - **Unicast**: Bit 0 of the first octet (MAC Address Field 1) must be cleared.
 - **Locally administered**: Bit 1 of the first octet (MAC Address Field 1) must be set.

Setting a MAC address that is not unicast or not locally administered will set the STATUS register to 102: ERROR – Configured MAC address is invalid.

Any modification to the MAC_ADR registers (when MAC_CFG = 2) is saved in internal non-volatile memory and does not take effect until the next power-up or reset. Writing the same value as previously stored does not trigger a non-volatile memory write.

Name: MAC_ADR
Description: Read or set the MAC address
Address Range: 65516 to 65521
Default Value: MAC address generated from the IS4350's serial number
Memory Type: Flash
Allowed Values: 0 to 255
Accessible by: I2C (Read and Write)
Note: Changes to this register are stored immediately but do not take effect until the next power-up cycle or reset.

Register 65516

Bit 15 to Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used	MAC Address Field 1 [XX:--:--:--:--]							

Register 65517

Bit 15 to Bit 8	Bit 7 to Bit 0
Not Used	MAC Address Field 2 [--:XX:--:--:--:--]

Register 65518

Bit 15 to Bit 8	Bit 7 to Bit 0
Not Used	MAC Address Field 3 [--:--:XX:--:--:--]

Register 65519

Bit 15 to Bit 8	Bit 7 to Bit 0
Not Used	MAC Address Field 4 [--:--:--:XX:--:--]

Register 65520

Bit 15 to Bit 8	Bit 7 to Bit 0
Not Used	MAC Address Field 5 [--:--:--:--:XX:--]

Register 65521

Bit 15 to Bit 8	Bit 7 to Bit 0
Not Used	MAC Address Field 6 [--:--:--:--:--:XX]

8.10 CHIP_ID Register

The `CHIP_ID` register contains the chip identifier, which is a fixed value of 150. This value is used for production tracking. It is stored in ROM and will not change throughout the product's life-cycle.

Since this register value is constant, reading it during firmware development can help verify that I2C communications are working and that the chip's memory can be properly read.

This register is read-only.

Name: `CHIP_ID`
Description: Chip Identification Number
Address: 65522
Value: 150
Memory Type: ROM
Accessible by: I2C (Read)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
0	0	0	0	0	0	0	0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	1	0	1	1	0

8.11 FW_REV Register

The `FW_REV` register indicates the chip internal firmware revision code.

This value is intended for production and product tracking. It is stored in ROM and may change throughout the product's life-cycle.

This register is read-only.

Name: `FW_REV`
Description: Firmware Revision Number
Address: 65523
Value: Depends on the revision, this document refers to: 0
Memory Type: ROM
Accessible by: I2C (Read)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
FW_REV[15 to 0]							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

8.12 I2C_READ_CRC Register

This register stores the CRC value of the last I2C read operation.

When I2C-CRC protection mode is enabled, (pin `I2C_CRC` = High), this register automatically stores the CRC of the last data frame read by your microcontroller from the IS4350.

For example, if your microcontroller performs an I2C read operation from `HOLD_0` to `HOLD_9`, once all the data has been received by your microcontroller, the IS4350 calculates the CRC of the data bytes (only the data bytes) and stores the result in this register. Your microcontroller can then perform a separate read operation to retrieve this value and compare it with the CRC it calculated independently.

If the two CRC values match, the read operation was completed successfully and data integrity is verified. If

they do not match, data corruption may have occurred during I2C communication, and the received data should be discarded.

Only the register data bytes (in the example: `HOLD_0` to `HOLD_9`) are included in the CRC calculation. The IS4350 I2C device address and the Pointer Register High and Pointer Register Low bytes are not included in the CRC generation.

Ensure that your microcontroller supports the I2C clock stretching feature, as the IS4350 may temporarily hold the I2C clock line low while computing the CRC.

The algorithm used to calculate the CRC for the I2C read operations is the standard CRC-16/Modbus. Refer to section I2C-CRC Protection for more details about the CRC implementation.

Name:	<code>I2C_READ_CRC</code>
Description:	CRC result of the last I2C read operation.
Address:	65524
Default Value:	0
Memory Type:	Volatile RAM
Accessible by:	I2C (Read)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
I2C_READ_CRC [15 to 0]							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

9 I2C

The IS4350 operates as a slave in the I2C-serial interface. It supports Standard Mode (100 kHz), Fast Mode (400 kHz), and Fast Mode Plus (1 MHz). The I2C master device, typically a microcontroller or a microprocessor, initiates and manages all read and write operations to the slave.

The IS4350 is represented on the I2C bus by one of the following addresses: 24, 25, 26, depending on the configuration of the ADR pin. See section ADR Pin for more details.

Pull-up resistors are required on the SCL and SDA lines for proper operation. The resistor values depend on the bus capacitance and operating speed. Typical values are 4.7 k Ω for Standard Mode (100 kHz) and 2 k Ω for Fast Mode and Fast Mode Plus (400 kHz and 1 MHz).

The high level of the IS4350 I2C pins can be either 3.3 V or 5 V. A logical '0' is transmitted by pulling the line low, while a logical '1' is transmitted by releasing the line, allowing it to be pulled high by the pull-up resistor. The master controls the Serial Clock (SCL) line, which generates the synchronous clock used by the Serial Data (SDA) line to transmit data.

The memory map consists of 65,536 registers; therefore, addressing a register requires 2 bytes. Each register is 16 bits wide.

The IS4350 requires clock stretching.

The operability of the Read and Write commands of the IS4350 is very similar to an EEPROM memory. Thinking of the IS4350 as an EEPROM memory is a good analogy to quickly understand how to communicate with it.

9.1 Highlights

I2C Device Address: 24, 25, 26, depending on the ADR pin configuration.

- 24 if pin ADR set to VSS.
- 25 if pin ADR set to VDD/2 (1.65 V).
- 26 if pin ADR set to VDD.

I2C Memory Map Addressing (Pointer Register) Size: 16-bit word (2 bytes), big-endian.

This is how you address from your microcontroller to the IS4350 memory map register you wish to read or write. As the IS4350 has more than 255 registers, you need two bytes for the addressing. It is big-endian, send the most significant byte first.

I2C Memory Map Register Size: 16-bit word (2x 8-bit), big-endian.

I2C Clock Stretching: Required.

Compatible I2C Speeds:

- Standard Mode (100 kHz), recommended SCL and SDA pull-up value: 4.7 k Ω
- Fast Mode (400 kHz), recommended SCL and SDA pull-up value: 2 k Ω
- Fast Mode Plus (1 MHz), recommended SCL and SDA pull-up value: 2 k Ω

Supported Operations:

- Single-Word Write
- Multiple-Word Write
- Single-Word Read
- Multiple-Word Read

9.2 Single Word Read

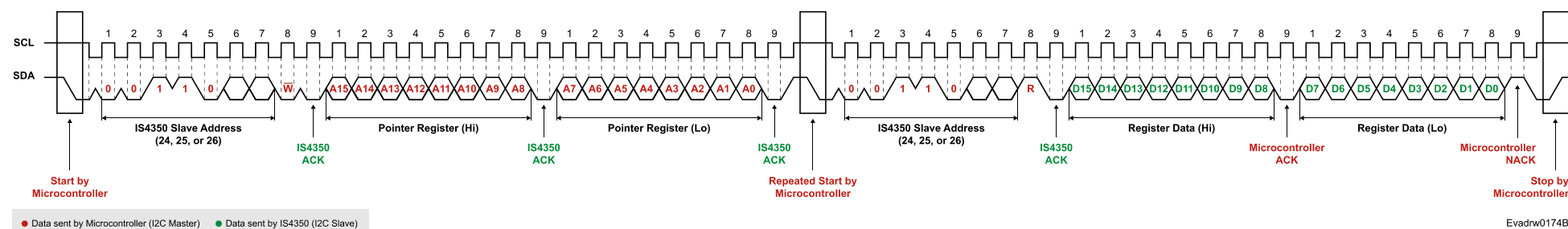
Reading a single word is an action performed by your microcontroller (I2C master) to access any register within the IS4350 memory (I2C slave), regardless of the last read or written position. To perform this action, your microcontroller must first load the address of the IS4350 register to be read into the IS4350's internal Pointer Register. Once the address is set, your microcontroller can retrieve the data from the specified register.

To initiate the Single Word Read operation, the following steps must be performed from the beginning: Your microcontroller starts by pulling SDA low while SCL is

high to generate a Start Condition. It then sends the IS4350 I2C device slave address 24, 25, or 26 with the R/W bit set to 0 (write). Upon receiving the device address, the IS4350 acknowledges it. Subsequently, your microcontroller sends the two bytes of the Pointer Register address: the most significant byte first, followed by the least significant byte, each acknowledged by the IS4350. This sets the address of the next word to be read in the Pointer Register.

Next, the register the Pointer Register points to needs to be read, which is a word (two bytes).

Your microcontroller generates a Repeated Start Condition, followed by the IS4350 I2C device address 24, 25, or 26, with the R/W bit set to 1 (read), instructing the IS4350 to retrieve data. The IS4350 acknowledges and responds with the most significant byte, which your microcontroller acknowledges (ACK). Then, the IS4350 sends the least significant byte, which your microcontroller does not acknowledge (NACK). Finally, your microcontroller issues a Stop Condition by raising the SDA line while the SCL is high.



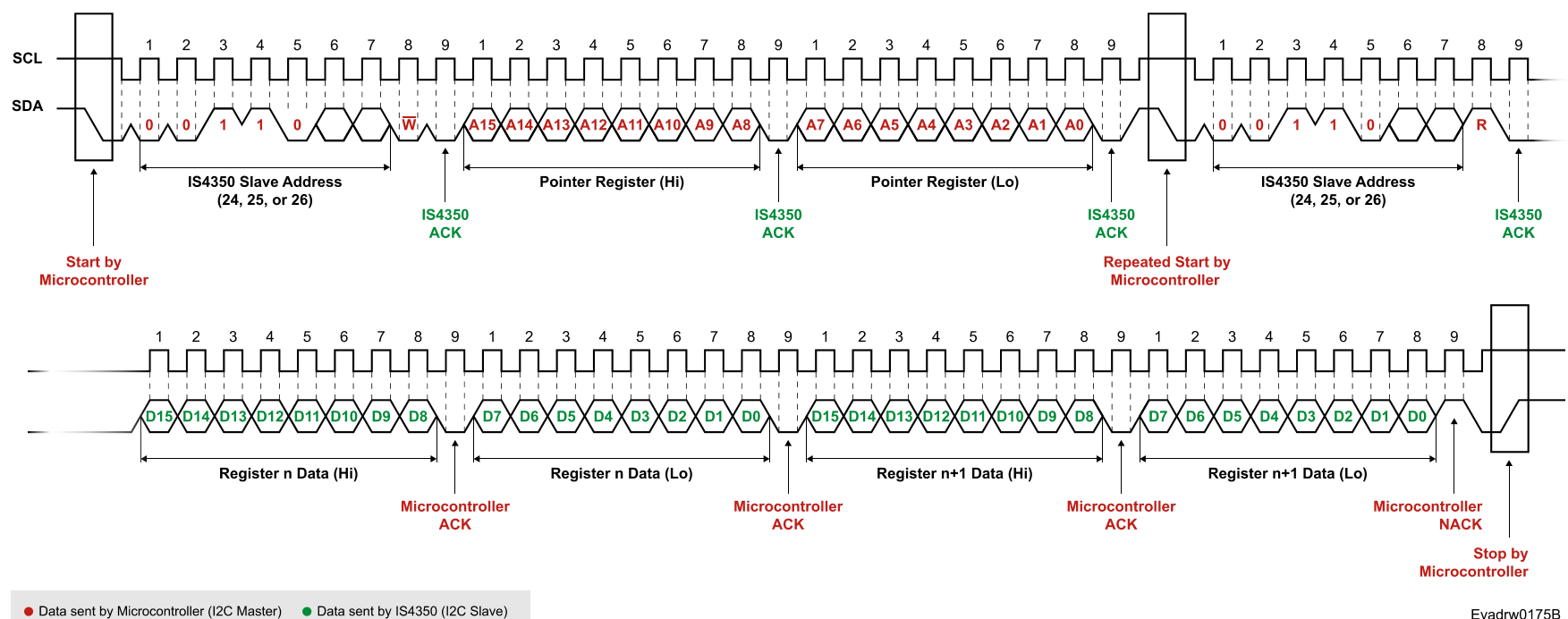
9.3 Multiple Word Read

Multiple Word Read operation functions similarly to Single Word Read but instead of reading a single memory register, it can read multiple of them in a single operation. Remember, the registers are 16-bit words consisting of 2 bytes, so the number of bytes retrieved should always be even.

To perform a Multiple Word Read operation, follow the same procedure as for a Single Word Read until the first data word is received. After receiving the first word, instead of generating a Not Acknowledge (NACK), your microcontroller should continue acknowledging (ACK) each received data byte from the IS4350 for as many

bytes as it intends to read. To conclude the read operation, after reading the last data word, your microcontroller should generate a Not Acknowledge (NACK) and a Stop Condition.

With each word read, the Pointer Register increments by one.



Evadrw0175B

9.4 Single Word Write

Writing a single word is an action performed by your microcontroller (I2C master) to write data to any register within the IS4350 memory (I2C slave), regardless of the last read or written position. To perform this action, your microcontroller must load the register address intended to be written into the IS4350's Pointer Register. Once the address is set, your microcontroller can send the data to be written into IS4350's memory map.

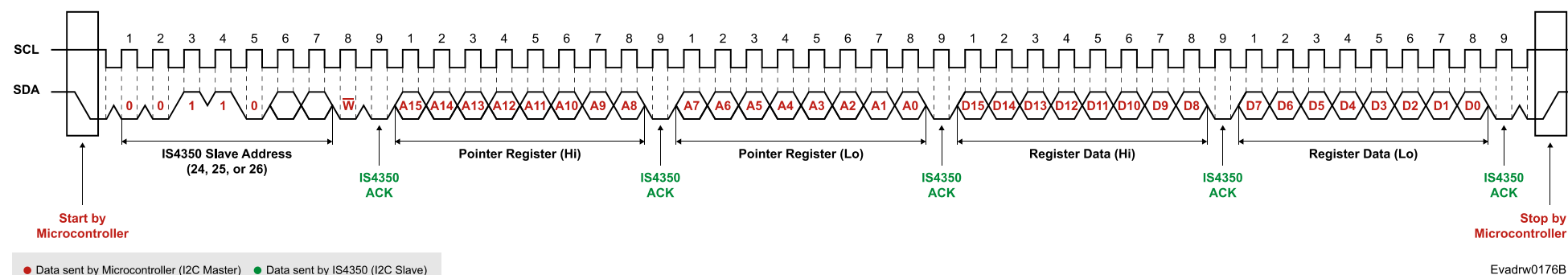
To initiate the Single Word Write operation, the following steps must be performed from the beginning: Your

microcontroller begins by pulling down the SDA line while the SCL line is high, creating a Start Condition. It then sends the IS4350 I2C device address 24, 25, or 26 with the R/W bit set to 0 (indicating a write operation).

Upon receiving the device address, the IS4350 acknowledges it. Subsequently, your microcontroller sends the two bytes of the register address it intends to write: the most significant byte first, followed by the

least significant byte, each acknowledged by the IS4350.

Your microcontroller then sends the most significant byte of the word to be written first, which the IS4350 acknowledges (ACK). Your microcontroller follows by sending the least significant byte of the word, which the IS4350 also acknowledges. Finally, your microcontroller issues a Stop Condition by raising the SDA line while the SCL line is high.

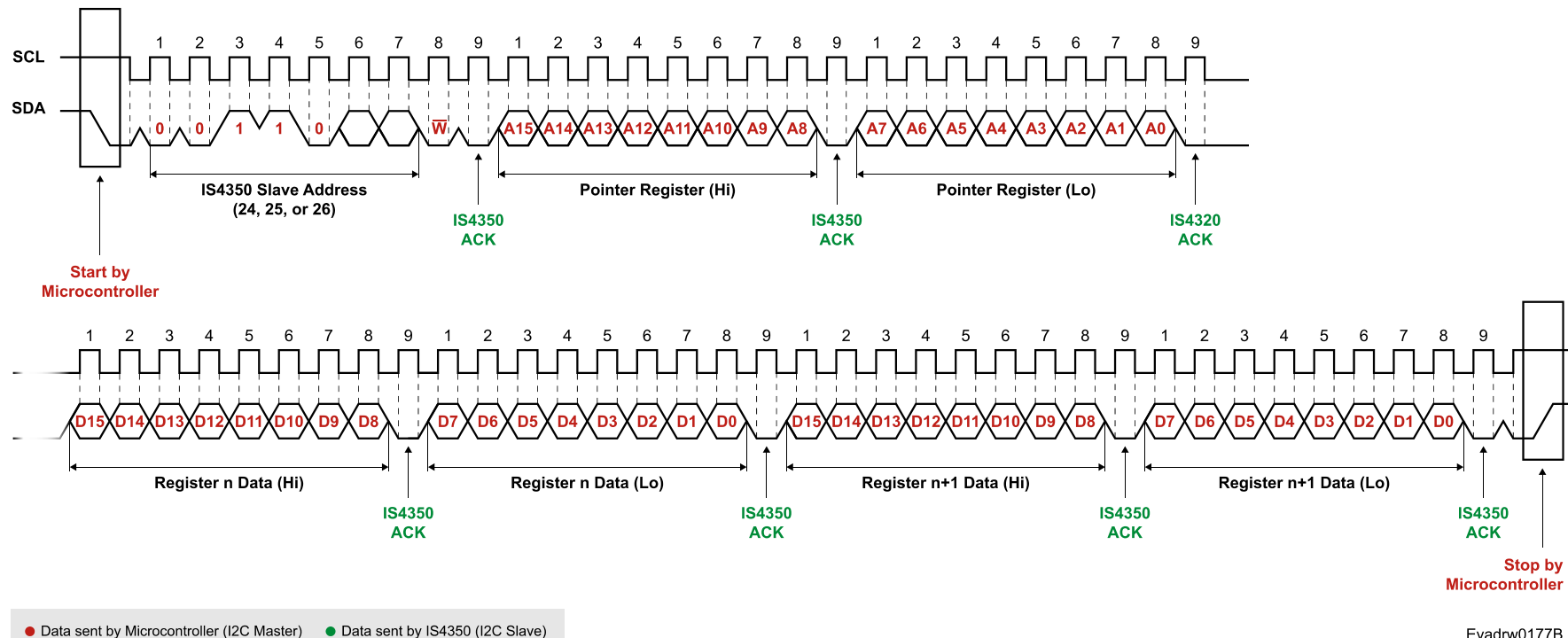


9.5 Multiple Word Write

The Multiple Word Write operation functions similarly to the Single Word Write, but instead of writing only one register, it can write multiple of them in a single operation.

To perform a Multiple Word Write operation, follow the same procedure as for a Single Word Write until the first data word is written. After writing the first word, instead of generating a Stop Condition, your microcontroller

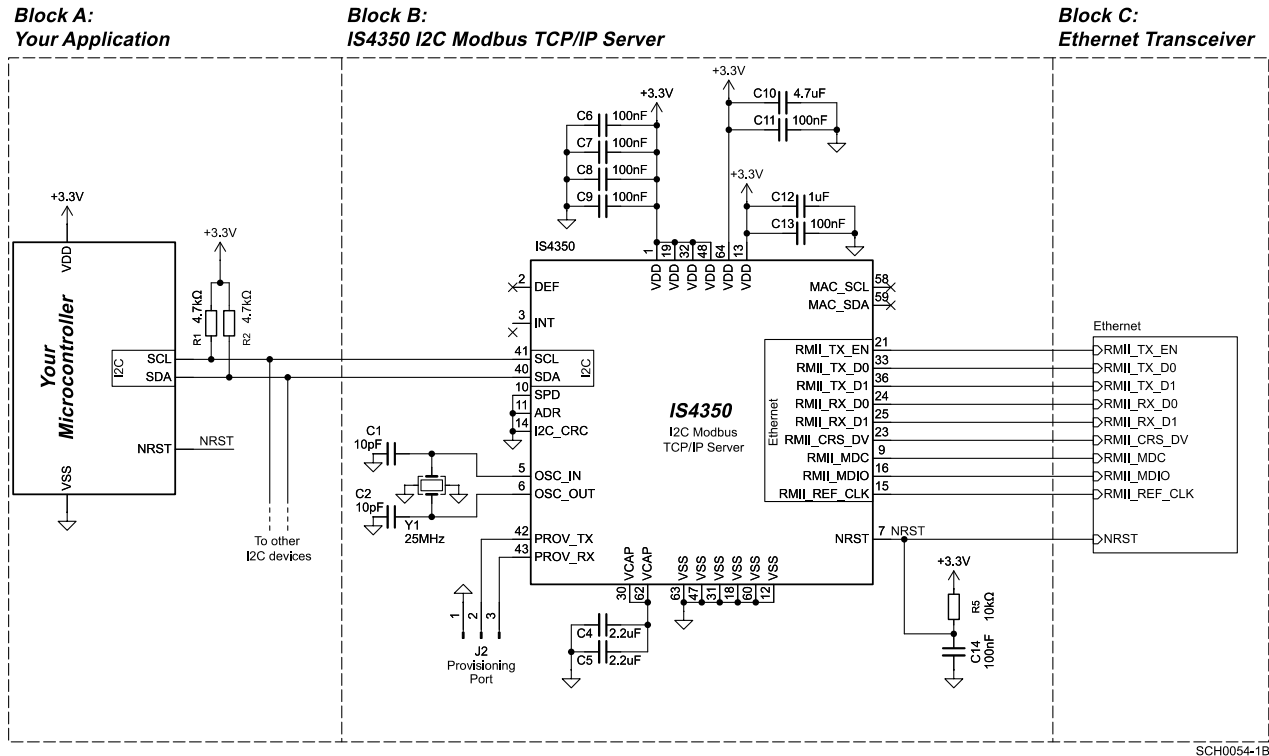
should continue writing data words. To conclude the write operation, after sending the last data word, your microcontroller should generate a Stop Condition.



10 Schematic Example

The following chapter represents an application design example for explanatory purposes and is not part of the product standard. The customer must design his own solution, choose its most appropriate components and validate the final product according to the legislation and the protocols specifications.

This example shows the design of a simple Modbus TCP/IP server, without using the INT pin, DEF pin and without using the external MAC address IC.



Block A: Your Application

Here is where typically we would find your microcontroller or microprocessor, acting as an I2C master and interfacing with the IS4350 and other I2C slave devices via the I2C serial interface.

Block B: IS4350 I2C Modbus TCP/IP Server

The IS4350 is simple to integrate into a design, only the Ethernet transceiver requires some special attention on the PCB design.

Decoupling Capacitor

Decoupling capacitors must be placed on the VDD pins as specified in the schematic, place them as close as possible to the IS4350.

Stabilization Capacitors

Stabilization capacitors must be connected to the VCAP pins for the proper operability of the IS4350 internal LDO regulator. The two VCAP pins must be tied together and connected to two 2.2 μ F capacitors in parallel. **These capacitors must not be connected to VDD.** Place them as close as possible to the IS4350.

DEF Pin

The DEF pin allows the use of a push button to recover the device in case it becomes unreachable due to an incorrect network configuration.

The implementation of this pin is optional. Leave it floating if not used; it has an internal weak pull-down (~40 k Ω).

If used, connect a push button to 3.3 V. An anti-bounce circuit may also be implemented.

INT Pin

The INT pin may be connected to the microcontroller to generate an interrupt whenever a Modbus client writes data to the IS4350. This functionality is useful when working with a large number of Holding Registers and continuous polling of the IS4350 memory is undesirable. Using the INT pin reduces your microcontroller CPU load looking for changes and minimizes I2C-serial interface utilization.

The implementation of this pin is optional. Leave it floating if not used.

Oscillator

The OSC_IN and OSC_OUT pins represent the typical oscillator configuration for a microcontroller. A 25 MHz crystal is required; the example shown uses an NX2016SA. Place the crystal as close as possible to the IS4350.

Reset

The NRST pin of the IS4350 connects to the nRST pin of the LAN8742A. This ensures the LAN chip starts in a known state whenever the IS4350 resets, preventing link errors and simplifying initialization.

Your microcontroller reset pin may be connected (if it is active-low and 3.3 V compatible) to the IS4350 NRST pin and the LAN8742A nRST pin. Check your microcontroller specifications to ensure proper interoperability between the three reset pins.

If interoperability between your microcontroller and the IS4350 is not guaranteed, you may drive the IS4350 reset pin using a GPIO or another output from your microcontroller.

External Unique MAC Address EEPROM

The MAC_SCL and MAC_SDA pins may be connected to a special EEPROM (AT24MAC402-STUM-T) that provides a global MAC address. When connected, and with the MAC_CFG register set to 3, the IS4350 will load the unique EUI-48 MAC address from the EEPROM and use it.

If a locally administered MAC address is used (MAC_CFG set to 1, 2 or 4), the AT24MAC402-STUM-T is not necessary, and the pins can be left floating.

I2C-Serial Interface

The IS4350 operates as an I2C-slave device and connects to your microcontroller via this I2C-serial interface.

The I2C bus requires pull-up resistors on the SDA and SCL lines. The resistor values depend on the bus capacitance and operating speed. Typical values are 4.7 kΩ for Standard Mode (100 kHz) and 2 kΩ for Fast Mode and Fast Mode Plus (400 kHz and 1 MHz).

The device includes internal input filters for the I2C serial interface. The filter settings are selected using the SPD pin: connect SPD to VSS for 100 kHz operation, to VDD/2 (1.65 V) for 400 kHz operation, or to VDD for 1 MHz operation.

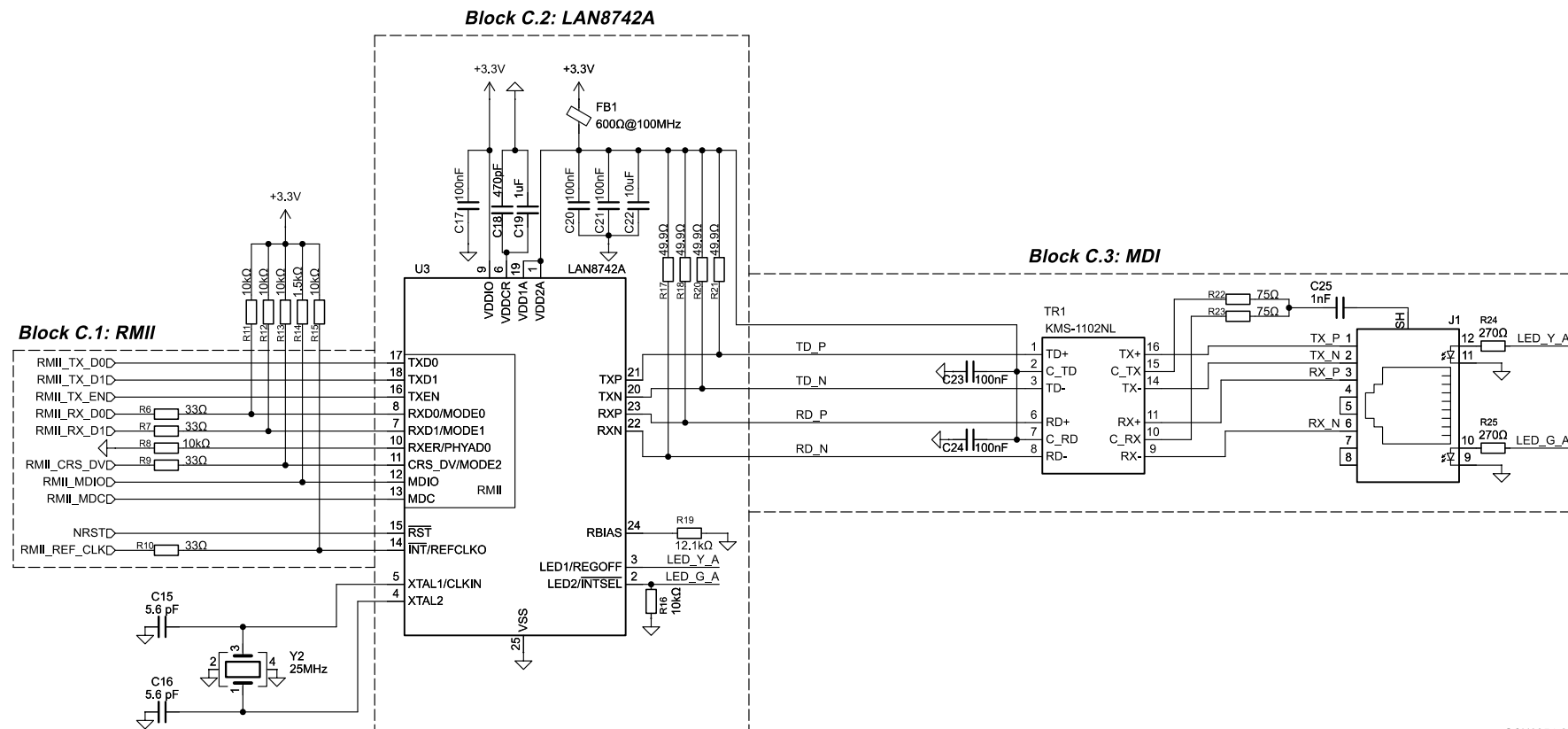
The I2C-slave address is selected using the ADR pin: connect ADR to VSS to select address 24, to VDD/2 (1.65 V) for address 25, and to VDD for address 26.

To enable the I2C-CRC protection feature, which protects I2C communication between your microcontroller and the IS4350 against data corruption, set the I2C_CRC pin high. The use of this feature is optional. If not used, leave the pin floating. It is internally connected to a weak pull-down resistor (~40 kΩ).

Block C: Ethernet Transceiver

The main component of this block is the LAN8742A, a 10/100 Mbps Ethernet physical layer transceiver (PHY). Its primary function is to bridge the digital signals from the IS4350 to the analog signals on the Ethernet twisted-pair cable.

The PCB layout of this block must follow proper Ethernet routing guidelines to ensure reliable communication with the local area network. **Failure to route this section according to LAN8742A specifications will result in a non-functional PCB.**



SCH0054-2A

Block C.1: RMII

The RMII (Reduced Media Independent Interface) section provides the digital communication interface between the IS4350 and the LAN8742A Ethernet PHY.

Basic routing guidelines for the RMII signals are provided below. The designer is responsible for verifying the exact routing requirements specified in the LAN8742A datasheet.

- The total length of each RMII trace should be less than 15 mm.
- RMII traces should be length-matched within 1.27 mm.
- RXD0 should be length-matched to RXD1, and TXD0 should be length-matched to TXD1.
- The single-ended impedance should be 50 Ω .

Block C.2: LAN8742A

Oscillator

The XTAL1 and XTAL2 pins represent the typical oscillator configuration. A 25 MHz crystal, such as the NX2016SA, may be used. Place the crystal as close as possible to the LAN8742A and keep the associated traces short.

Bias Resistor

The RBIAS pin connects to digital ground through a 12.1 k Ω resistor with a tolerance of 1.0%. This resistor is used to establish critical internal bias currents for the LAN8742A.

Block C.3: MDI

MDI (Medium Dependent Interface) traces are the Ethernet differential signal pairs that carry Ethernet data between the LAN8742A and the Ethernet magnetics (transformer).

Basic routing guidelines for the MDI traces are provided below. The designer is responsible for verifying the exact routing requirements specified in the LAN8742A datasheet.

- The total length of each MDI trace should be less than 50 mm.
- The differential pairs should be length-matched within 1.27 mm.
- The differential impedance should be 100 Ω .

Magnetics

The Ethernet magnetics may be implemented as a discrete component or integrated into the RJ45 connector.

RJ45 connectors with integrated magnetics reduce PCB area and simplify routing, but are typically more expensive than solutions using discrete magnetics.

11 Firmware Examples

11.1 Arduino Example

Example Name

ISXMPL4350ex3

Example Description

This example increments Holding Register 0 (HOLD_0) by 1 every second.

Code Structure

Writing Arduino code for the IS4350 is straightforward: no dedicated library is needed, only the standard Wire library and two short routines, one to write a register and one to read it.

Register addresses and register values are both 16-bit and are sent MSB first. A write therefore sends four bytes: two for the register address and two for the data. A read sends the two address bytes, and the IS4350 then returns the two data bytes.

Before going online, the example writes an initial value to HOLD_0, so Modbus clients never read invalid data. It then writes 1 to the GO_ONLINE register to start the Modbus server, and polls the STATUS register until it reads 5 (Modbus server ready and listening).

By default, the IS4350 uses DHCP and a locally administered MAC address generated from its serial number, so no network configuration is needed.

Once the server is online, the application only needs to read and write the Holding Registers it uses..

```

/*
 * ISXMPL4350ex3: IS4350 - Simple Arduino Uno Example
 * -----
 * - Uses DHCP (default, no configuration needed)
 * - Brings the Modbus server online (GO_ONLINE = 1)
 * - Increments Holding Register 0 (HOLD_0) by 1 every second
 *
 * Wiring:
 *   Arduino Uno A4 (SDA) -> IS4350 SDA (4.7 kOhm pull-up to 5 V)
 *   Arduino Uno A5 (SCL) -> IS4350 SCL (4.7 kOhm pull-up to 5 V)
 *   Arduino Uno GND -> IS4350 VSS
 *   ADR pin on IS4350 tied to GND -> I2C address 24 (0x18)
 *   SPD pin on IS4350 tied to GND -> 100 kHz (Arduino Wire default)
 *
 * Notes:
 *   - Register addresses and values are 16-bit, sent MSB first.
 *   - GO_ONLINE is 0 after every power-up, so set it at startup.
 *   - Leave a few milliseconds between consecutive I2C operations.
 */

#include <Wire.h>

const uint8_t IS4350_ADDR = 24; // I2C address (ADR pin tied to GND)
const uint16_t REG_HOLD_0 = 0; // Holding Register 0
const uint16_t REG_GO_ONLINE = 65513; // Modbus server on/off (I2C only)
const uint16_t REG_STATUS = 65514; // Server status (low byte: 5 = ready)

uint16_t counter = 0;

void is4350WriteRegister(uint16_t regAddr, uint16_t value) {
  Wire.beginTransaction(IS4350_ADDR);
  Wire.write((uint8_t)(regAddr >> 8)); // Register address, MSB first
  Wire.write((uint8_t)(regAddr & 0xFF));
  Wire.write((uint8_t)(value >> 8)); // Value, MSB first
  Wire.write((uint8_t)(value & 0xFF));
  Wire.endTransmission();
  delay(10); // A few ms between I2C operations
}

uint16_t is4350ReadRegister(uint16_t regAddr) {
  Wire.beginTransaction(IS4350_ADDR);
  Wire.write((uint8_t)(regAddr >> 8)); // Register address, MSB first
  Wire.write((uint8_t)(regAddr & 0xFF));
  Wire.endTransmission(false); // Repeated start (no stop)
  Wire.requestFrom(IS4350_ADDR, (uint8_t)2);
  uint8_t msb = Wire.read(); // Value, MSB first

```

```
uint8_t lsb = Wire.read();
delay(10); // A few ms between I2C operations
return ((uint16_t)msb << 8) | lsb;
}

void setup() {
  Serial.begin(9600);
  Wire.begin();
  delay(1000); // Let the IS4350 start up

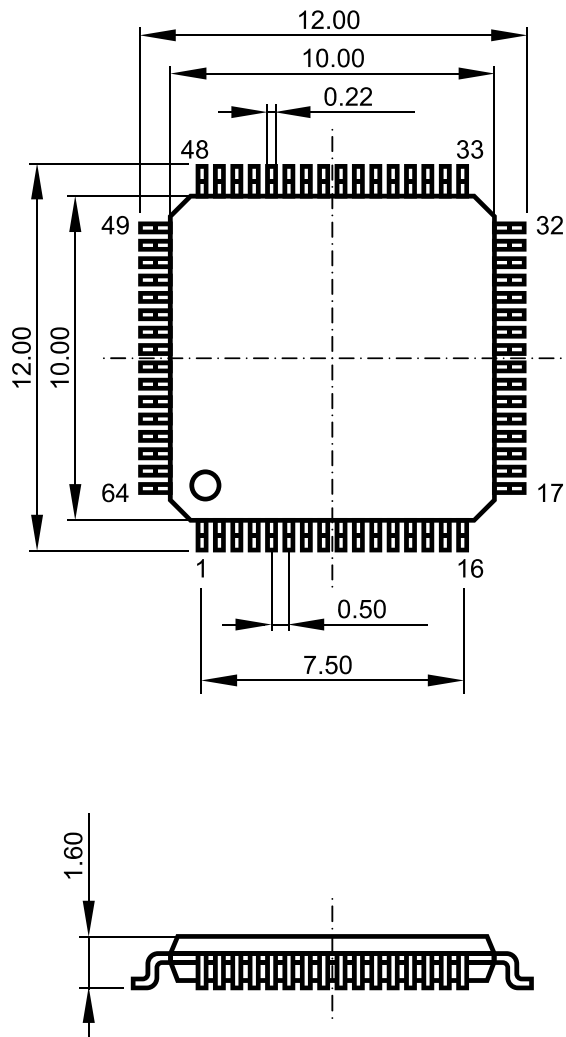
  is4350WriteRegister(REG_HOLD_0, counter); // Valid data before going online
  is4350WriteRegister(REG_GO_ONLINE, 1); // Bring the server online

  // Wait until the server is ready and listening (STATUS low byte = 5)
  uint8_t status = 0;
  while (status != 5) {
    delay(500);
    status = is4350ReadRegister(REG_STATUS) & 0xFF;
    Serial.print("STATUS: ");
    Serial.println(status);
  }
  Serial.println("Server online.");
}

void loop() {
  counter++;
  is4350WriteRegister(REG_HOLD_0, counter);
  Serial.print("HOLD_0 = ");
  Serial.println(counter);
  delay(1000);
}
```

12 Mechanical

LQFP64 Package

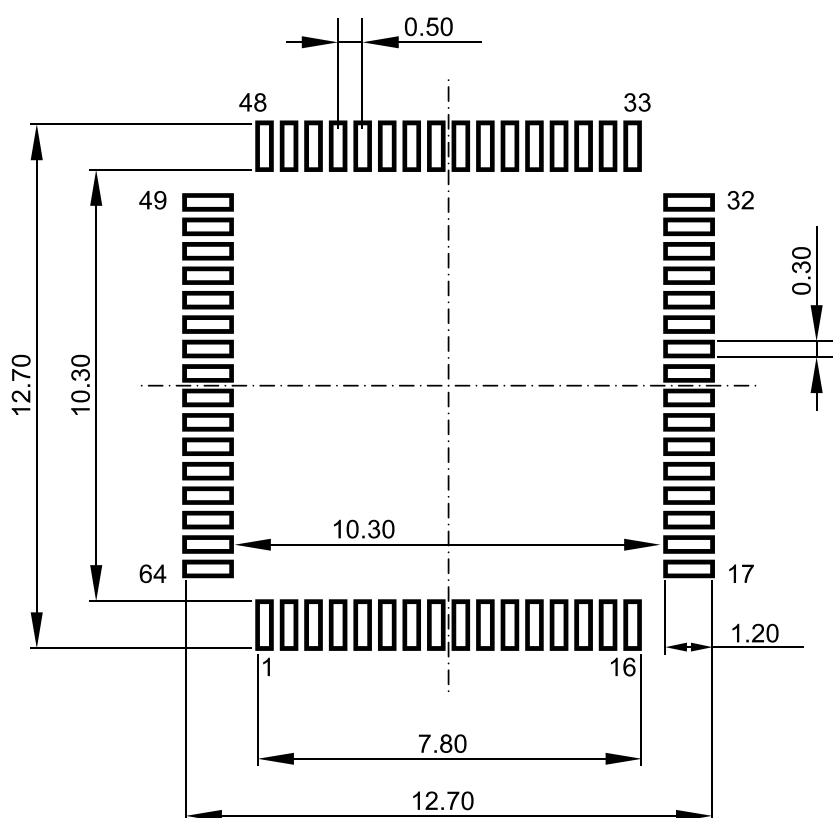


Units: millimeters

Notes:
This drawing is for general information only.
Drawing not to scale.

Evadw0107r1

LQFP64 Recommended Footprint



Units: millimeters

Notes:
This drawing is for general information only.
Drawing not to scale.

Evadw0108r1

Content

IS4350: I2C Modbus TCP/IP Server.....	1	7.6 MAC_SCL & MAC_SDA Pins.....	14
1 Ordering Information.....	2	7.7 SPD Pin.....	15
2 Revision History.....	3	7.8 ADR Pin.....	15
2.1 Document Revision.....	3	7.9 NRST Pin.....	16
2.2 Firmware Revision.....	3	8 Memory Map.....	17
3 Specifications.....	4	8.1 HOLD_X Registers.....	18
Absolute Maximum Ratings.....	4	8.2 DHCP Register.....	19
Recommended Operating Conditions.....	4	8.3 IP Registers.....	20
Electrical Characteristics.....	4	8.4 MASK Register.....	21
Flash Memory Cell Characteristics.....	5	8.5 GATEWAY Register.....	22
Modbus Characteristics.....	5	8.6 GO_ONLINE Register.....	23
Modbus Performance.....	5	8.7 STATUS Register.....	24
Ethernet Characteristics.....	5	8.8 MAC_CFG Register.....	25
4 Description.....	6	8.9 MAC_ADR Registers.....	26
What It Is and What It Does.....	6	8.10 CHIP_ID Register.....	27
Utility.....	6	8.11 FW_REV Register.....	27
Benefits.....	6	8.12 I2C_READ_CRC Register.....	28
How It Works.....	6	9 I2C.....	29
Modbus and Ethernet.....	6	9.1 Highlights.....	29
5 Safety.....	7	9.2 Single Word Read.....	30
5.1 I2C-CRC Protection.....	7	9.3 Multiple Word Read.....	31
5.1.1 I2C-CRC for Read Operations.....	8	9.4 Single Word Write.....	32
5.1.2 I2C-CRC for Write Operations.....	9	9.5 Multiple Word Write.....	33
5.2 I2C Watchdog Timer (I2C-WDT).....	10	10 Schematic Example.....	34
5.3 Fail-Safe Power-Up.....	10	11 Firmware Examples.....	38
6 Usage.....	11	11.1 Arduino Example.....	38
Firmware Design Structure.....	11	12 Mechanical.....	40
Accessing the IS4350 via Modbus.....	11	Content.....	42
Firmware Best Practices.....	11	Appendix.....	43
7 Pins.....	12	Documentation Feedback.....	43
7.1 DEF Pin.....	13	Sales Contact.....	43
7.2 INT Pin.....	14	Customization.....	43
7.3 PROV_RX & PROV_TX Pins.....	14	Trademarks.....	43
7.4 I2C_CRC Pin.....	14	Disclaimer.....	44
7.5 SCL & SDA Pins.....	14		

Appendix

Documentation Feedback

Feedback and error reporting on this document are very much appreciated. Please indicate the code or title of the document.

feedback@inacks.com

Sales Contact

For special order requirements, large volume orders, or scheduled orders, please contact our sales department at:

sales@inacks.com

Customization

INACKS can develop new products or customize existing ones to meet specific client needs. Please contact our engineering department at:

engineering@inacks.com

Trademarks

This company and its products are developed independently and are not affiliated with, endorsed by, or associated with any official protocol or standardization entity. All trademarks, names, and references to specific protocols remain the property of their respective owners.

Disclaimer

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, INACKS does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. INACKS takes no responsibility for the content in this document if provided by an information source outside of INACKS.

In no event shall INACKS be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, INACKS's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of INACKS.

Right to make changes — INACKS reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — INACKS products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an INACKS product can reasonably be expected to result in personal injury, death or severe property or environmental damage. INACKS and its suppliers accept no liability for inclusion and/or use of INACKS products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. INACKS makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using INACKS products, and INACKS accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the INACKS product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

INACKS does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using INACKS products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). INACKS does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — INACKS products are sold subject to the general terms and conditions of commercial sale, as published at <http://www.inacks.com/commercialsaleterms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. INACKS hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of INACKS products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or

the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Non-automotive qualified products — This INACKS product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. INACKS accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

Protocol Guidance Disclaimer: The information provided herein regarding the protocol is intended for guidance purposes only. While INACKS strive to provide accurate and up-to-date information, this content should not be considered a substitute for official protocol documentation. It is the responsibility of the client to consult and adhere to the official protocol documentation when designing or implementing systems based on this protocol.

INACKS make no representations or warranties, either expressed or implied, as to the accuracy, completeness, or reliability of the information contained in this document. INACKS shall not be held liable for any errors, omissions, or inaccuracies in the information or for any user's reliance on the information.

The client is solely responsible for verifying the suitability and compliance of the provided information with the official protocol standards and for ensuring that their implementation or usage of the protocol meets all required specifications and regulations. Any reliance on the information provided is strictly at the user's own risk.

Certification and Compliance Disclaimer: Please be advised that the product described herein has not been certified by any competent authority or organization responsible for protocol standards. INACKS do not guarantee that the chip meets any specific protocol compliance or certification standards.

It is the responsibility of the client to ensure that the final product incorporating this product is tested and certified according to the relevant protocol standards before use or commercialization. The certification process may result in the product passing or failing to meet these standards, and the outcome of such certification tests is beyond our control.

INACKS disclaim any liability for non-compliance with protocol standards and certification failures. The client acknowledges and agrees that they bear sole responsibility for any legal, compliance, or technical issues that arise due to the use of this product in their products, including but not limited to the acquisition of necessary protocol certifications.