

MOD-RS485-ISO README

Generated from WIKI-ARTICLE.txt on 2026-06-18 16:26.

Description

MOD-RS485-ISO is an isolated RS485/RS422 interface with a PIC16LF18324 controller and a TI ISO35T/ISO35TDW transceiver. The board connects through UEXT and can be used either as a transparent UART-to-RS485 adapter or as an I2C-controlled RS485 bridge.

Experimental release notice: Firmware 0x06 build 02h is an experimental release intended for validation with real RS485/Modbus installations. It fixes the default half-duplex behavior compared to the old firmware, but it should be bench-tested in the target application before production use.

Firmware 0x06 keeps compatibility with the command/register interface, but the default pass mode is now intended for normal 2-wire half-duplex RS485/Modbus use. The PIC automatically controls the ISO35 DE pin, so common UART/RS485 libraries do not need a separate DE/RE GPIO.

For 2-wire half-duplex RS485 and Modbus RTU, the **Z-B** and **Y-A** jumpers on MOD-RS485-ISO must be closed. These jumpers connect the transmit and receive differential pairs together so the board uses one shared RS485 pair. Leave them open only for RS422/full-duplex style wiring.

Leds

The device has two LEDs:

- **Green** - TX path is enabled
- **Red** - RX path is enabled

By default both paths are enabled, so both LEDs should be on.

Registers

Register map starts at address 20h.

Device registers			
Name	Address	Default	Description
DeviceID	20h	25h	Device identifier, read-only
Firmware	21h	06h	Firmware revision
Address	22h	22h	I2C slave address. PROG jumper must be closed to change it.
Mode	23h	00h	00h pass mode, 01h bridge mode
Control	24h	03h	Enable RX/TX path and optional forced TX driver
Baudrate	25h	0Ch	RS485 timing profile. In pass mode this is used for automatic DE timing.
TX	26h	x	Bridge-mode TX FIFO
RX	27h	x	Bridge-mode RX FIFO
RX count	28h	00h	Extension register: bytes waiting in bridge RX FIFO

TX free	29h	40h	Extension register: free bytes in bridge TX FIFO
Status	2Ah	x	Extension register: status bitfield
Save	2Bh	x	Extension register: write A5h to persist current control register
Firmware build	2Ch	02h	Firmware 0x06+: build/debug marker

Registers 20h through 27h remain compatible with previous firmware releases.

Modes of operation

The device has two modes:

- **Pass** - Default mode. UEXT TXD is passed to the RS485 driver input and RS485 receiver output is passed back to UEXT RXD. Firmware 0x06 controls DE automatically for half-duplex operation.
- **Bridge** - TX and RX are handled through the I2C FIFO registers. This is kept for compatibility and control use, but pass mode is recommended for Modbus RTU and normal UART libraries.

Changing mode is saved to EEPROM and applied after reset.

Automatic half-duplex direction

In older firmware, default control 03h enabled RX and held TX driver enable high. That behavior is useful for some RS422/full-duplex arrangements but is not friendly to 2-wire RS485 because the board keeps driving the bus while remote nodes need to answer.

In firmware 0x06, default control 03h means RX enabled and TX enabled with automatic DE timing. In build 02h, the PIC detects TX edges with CLC1, drives DE from firmware on RC3, and releases DE with Timer0 after the configured idle timeout. Standard Modbus RTU libraries can therefore use the board like a normal transparent RS485 adapter.

Automatic DE timing is not bus arbitration. With Z-B and Y-A jumpers closed for 2-wire half-duplex RS485, only one node may transmit at a time. If two hosts send data simultaneously, both RS485 drivers will drive the bus and data corruption is expected. Protocols such as Modbus RTU avoid this with master/slave turn-taking.

For MOD-RS485-ISO boards marked with **Y-A** and **Z-B** jumpers, both jumpers must be closed for the normal 2-wire half-duplex use case. This is the jumper state used by the transparent and Modbus examples below.

Terminal newline tests

Manual serial terminal tests can look different depending on CR/LF handling. PuTTY commonly sends Enter as carriage return (CR, 0Dh), while Arduino Serial Monitor can be configured to append no line ending, NL, CR, or both NL and CR to text sent from its input box. Terminal display options can also decide whether a received CR or LF moves to a new visual line. The firmware does not convert line endings; transparent mode passes CR, LF, and binary data unchanged. This is required for Modbus RTU and other binary protocols.

For the Arduino examples, the PC terminal baud and RS485 bus baud are not the same setting. The USB diagnostic terminal normally stays at 115200 because the sketch uses `Serial.begin(115200)`. The RS485 bus baud is selected by the sketch and must match the PIC baud timing profile.

Arduino examples

Arduino IDE installation

The Arduino support is provided as a normal Arduino library folder. To install it in Arduino IDE:

- Zip the **MOD-RS485-ISO-Arduino-demos** folder. The zip should contain that folder with **library.properties**, **src**, and **examples** inside it.
- In Arduino IDE, select **Sketch -> Include Library -> Add .ZIP Library...** and choose the zip file.
- After installation, open the examples from **File -> Examples -> MOD-RS485-ISO**.

- The **ModbusMaster** example requires the Arduino **ModbusMaster** library. Install it from Arduino IDE Library Manager before compiling that example, or let Arduino IDE install dependencies if it offers to do so.

The main examples are:

- **Transparent** - raw transparent UART-to-RS485 test.
- **ModbusMaster** - load this on one board for a Modbus RTU master test.
- **ModbusSlave** - load this on the second board for the matching slave test.
- **FullDuplex** - full-duplex/RS422-style transparent UART test with Z-B and Y-A open.

Do not load the master example on both boards. Modbus RTU needs one master and one or more slaves.

The examples default to Olimex ESP32-PoE-ISO pins: UEXT UART1 TX is GPIO4, UEXT UART1 RX is GPIO36, I2C SDA is GPIO13, and I2C SCL is GPIO16. To use another Arduino-compatible host board, edit the configuration block at the top of the sketch. Boards with a second hardware UART, such as Olimexino-2560 or Arduino Mega, can use Serial1. ATmega328-class boards, such as Olimexino-328 or Arduino Uno, can use the SoftwareSerial fallback if the selected pins are wired to the MOD-RS485-ISO UART. Keep SoftwareSerial baud rates conservative.

The FullDuplex example is not for the normal shared two-wire RS485 setup. Leave Z-B and Y-A open, use two differential pairs, and let the sketch set control 07h so RX_ENABLE, TX_ENABLE, and TX_FORCE are all enabled. TX_FORCE keeps the driver enabled for continuous transmit on the Y/Z pair while the A/B pair is used for receive.

Hardware setup

For the two-board ESP32-PoE-ISO Modbus test you need:

- 2 x ESP32-PoE-ISO boards
- 2 x MOD-RS485-ISO boards programmed with PIC firmware 0x06 build 02h
- 2 x USB cables for programming and serial monitor access
- One RS485 twisted pair between the two MOD-RS485-ISO boards

Plug each MOD-RS485-ISO into the UEXT connector of an ESP32-PoE-ISO board. Close **Z-B** and **Y-A** on both MOD-RS485-ISO boards for half-duplex operation. Connect the combined A/Y line of one board to the combined A/Y line of the other board, and the combined B/Z line to the combined B/Z line. If a third-party RS485 cable or adapter uses only A/B names and there is no communication, swap the pair.

Keep the PC serial terminals at 115200 baud. This is the USB diagnostic baud. The RS485 bus baud is configured inside the sketches with **RS485_BAUD** and **RS485_BAUD_PROFILE**; both sketches must use the same RS485 settings.

Example behavior

For a two-board Modbus RTU bench test, program one host board with ModbusMaster and the other with ModbusSlave. The master example uses the Arduino ModbusMaster library and reads holding registers 0000h and 0001h from slave ID 1. The slave example is a small self-contained function-03 responder for this test and does not require a separate Modbus slave library. With the defaults, the master should print R0=1234 and an incrementing R1 once per second.

Do not run the master example on both boards. Modbus RTU requires one master and one or more slaves. The included examples force the PIC-controlled DE signal before each complete Modbus frame, wait for the PIC status bit to confirm DE, keep the receive path enabled, drain possible local echo after the host UART flush, and then release the bus.

Control data direction

Writing 24h changes the control register.

Bit	7	6	5	4	3	2	1	0
Control	x	x	x	x	x	TX_FORC E	TX_ENABL E	RX_ENABL E

- **RX_ENABLE:**
 - 0 - Disable RX path
 - 1 - Enable RX path
- **TX_ENABLE:**
 - 0 - Disable TX path and keep DE low
 - 1 - Enable TX path
- **TX_FORCE:**
 - 0 - Automatic DE timing, recommended for 2-wire RS485
 - 1 - Force DE high while TX_ENABLE is set, legacy/full-duplex behavior

Examples:

- 03h - RX enabled, TX enabled, automatic DE, default
- 01h - RX only
- 02h - TX only with automatic DE
- 07h - RX enabled, TX enabled, DE forced high

Control changes are volatile by default. Write A5h to register 2Bh to persist the current control register.

Baudrate timing profiles

Firmware 0x06 supports common RS485/Modbus rates:

- 05h - 300 bps
- 06h - 600 bps
- 07h - 1200 bps
- 09h - 2400 bps
- 0Ah - 4800 bps
- 0Ch - 9600 bps (default)
- 0Dh - 14400 bps
- 0Eh - 19200 bps
- 0Fh - 38400 bps
- 10h - 57600 bps
- 12h - 115200 bps

In bridge mode this sets the UART baudrate. In pass mode the data path remains hardware-passed, and this value only controls the automatic DE release timing. Unsupported baudrate writes are ignored and are reported in the status register.

Status register

Register 2Ah returns:

Bit	7	6	5	4	3	2	1	0
Status	x	x	x	UNSUPPORTED_BR	PASS_MODE	DRIVER_ENABLED	TX_SPACE	RX_AVAILABLE

FIFO

Bridge mode has 64-byte TX and RX FIFOs. Reading register 28h returns the number of waiting RX bytes. Reading register 29h returns free TX space. These registers avoid the old ambiguity where reading RX returned 00h both for an empty FIFO and for a real 00h data byte.

Changing device address

Writing register 22h changes the I2C address only when the PROG jumper is closed.

Restoring default parameters

Close the PROG jumper and reset the module to restore default parameters: address 22h, pass mode, control 03h, baud timing 9600 bps.

Hardware

Latest hardware revision is revision C and uses PIC16LF18324.

For 2-wire half-duplex RS485, close the MOD-RS485-ISO **Z-B** and **Y-A** jumpers. This setting is required for the transparent and Modbus examples when using one shared RS485 pair.

Software

The firmware is built with MPLAB X and XC8. Firmware 0x06 build 02h was checked with MPLAB X 6.20, MPLAB XC8 3.10, and PIC16F1xxx_DFP 1.24.387.

Arduino examples include Transparent, ModbusMaster, ModbusSlave, and FullDuplex sketches. They default to ESP32-PoE-ISO pins but can be configured for other host boards from the comment/configuration block at the top of each sketch. The ModbusMaster sketch requires the Arduino ModbusMaster library.

Category: Interface