QUICK REFERENCE · RS4853V

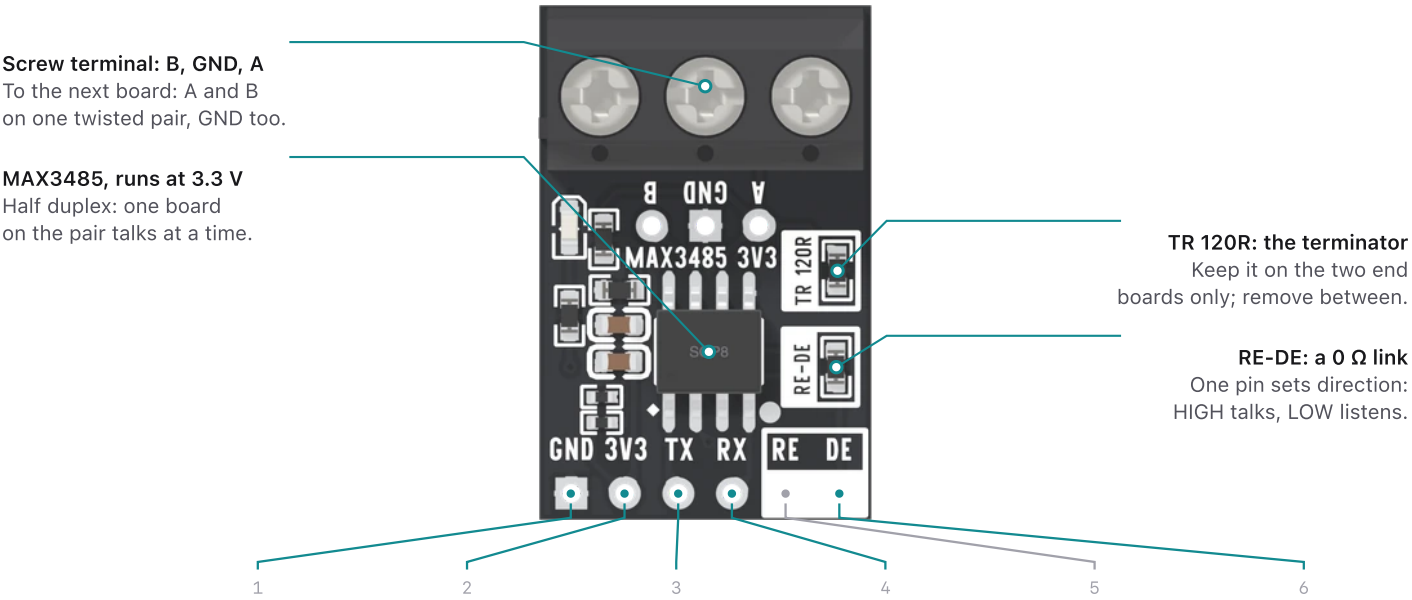
RS485 Board, MAX3485

UART to RS485 for 3.3 V microcontrollers. Its twin, the MAX485 board, is for 5 V ones.

3.3 V board: power it from 3V3 At 5 V its TX puts 5 V on your microcontroller's RX pin. **Power it from the 3V3 pin, and use the MAX485 board on a 5 V Arduino Uno.**

Pins, and which to wire

Drawn from above, terminal at the top. The six holes along the bottom go to your microcontroller; B, GND and A go to the bus, on the terminal or the three holes under it. From underneath the order reads mirror-image.



Front	**GND**	**3V3**	**TX**	**RX**	**RE**	**DE**
Back	GND	3V3	TX	RX	RE	DE
Your board	GND	3V3	RX	TX	optional	DIR

TX goes to your RX The board names its pins from its own side. Its TX is the chip's receiver output, what it heard on the bus, so it goes to your RX; its RX takes your TX. TX to TX moves nothing in either direction, with no error anywhere.

The 120 Ω, on how many boards

The two end boards

60 Ω across A and B

25 mA to make 1.5 V: inside the 54 Ω the chips are specified into.

3 fitted

40 Ω across A and B

38 mA to make 1.5 V: past the 54 Ω the chips are specified into.

All twelve fitted

10 Ω across A and B

150 mA to make 1.5 V: past the 54 Ω the chips are specified into.

Every board ships with its 120 Ω fitted. Keep it on the two boards at the ends of the cable and take it off every board between: both chips are specified into 54 Ω, and three or more fitted is past it.

Arithmetic, not measurements: n resistors of 120 Ω in parallel, the current a driver needs for the 1.5 V minimum in both datasheets, and the idle difference the 20 kΩ bias pair leaves across the terminators at 3.3 V, with the receivers' own inputs left out. A receiver is sure of a 1 only above 200 mV, so a terminated bus with nobody talking is undecided: frame your messages and ignore stray bytes.

One ESP32, two boards

Arduino IDE · Board: ESP32 Dev Module or ESP32S3 Dev Module · No libraries

```
// Board A talks on Serial1, board B listens on Serial2.
const int A_RO = 16, A_DI = 17, A_DIR = 4;
const int B_RO = 18, B_DI = 5, B_DIR = 21;
String line;
int count = 0;
uint32_t last = 0;

void setup() {
  pinMode(A_DIR, OUTPUT);
  digitalWrite(A_DIR, LOW);        // listen first
  pinMode(B_DIR, OUTPUT);
  digitalWrite(B_DIR, LOW);
  Serial.begin(115200);
  Serial1.begin(9600, SERIAL_8N1, A_RO, A_DI);
  Serial2.begin(9600, SERIAL_8N1, B_RO, B_DI);
}

void loop() {
  if (millis() - last >= 1000) {
    last = millis();
    digitalWrite(A_DIR, HIGH);     // A takes the bus
    delayMicroseconds(100);
    Serial1.printf("hello %d\n", ++count);
    Serial1.flush();               // wait for the stop bit
    digitalWrite(A_DIR, LOW);      // let go
  }
  while (Serial2.available()) {
    char c = Serial2.read();
    if (c != '\n') { if (line.length() < 40) line += c; continue; }
    line.trim();
    if (line.startsWith("hello")) Serial.println("B heard " + line);
    line = "";
  }
}
```

```
B heard hello 1
B heard hello 2
```

Serial Monitor at 115200, once a second. Each line went out of board A, across the pair, and into board B.

Wiring

	FRONT	BOARD A	BOARD B
1	**GND**	GND	GND
2	**3V3**	3V3	3V3
3	**TX**	16	18
4	**RX**	17	5
5	**RE**	—	—
6	**DE**	4	21

GPIO numbers on the ESP32. Then the two terminals A to A, B to B, GND to GND. Keep both 120 Ω fitted: two boards are the two ends. RE is the same net as DE, so it needs no wire. The pins suit a classic ESP32 and an ESP32-S3 alike.

RE and DE, one pin

A 0 Ω link in the box printed RE-DE joins RE to DE, so one GPIO sets the direction: HIGH talks, LOW listens, never both. Nothing on the board pulls it either way, so set it LOW first thing in setup(), and drop it only after flush() returns, or the last byte is cut off.

When it does not work

Nothing arrives

TX to TX is the usual cause: the board's TX goes to your RX. Then check the talker raises its direction pin to send.

I read back my own messages

The receiver is on while you send: the RE-DE link was removed and RE tied LOW. Refit the link, or give RE the DE pin.

Stray bytes when nobody talks

Normal on a terminated bus. Send whole lines and ignore anything that is not one.

Every character is wrong

A and B are crossed between the terminals, which turns every bit over. Swap them at one end, and check both ends use one baud rate.

Fine on the desk, not on the cable

Run GND with A and B, twist A and B together, and keep the 120 Ω on the two end boards only.

It broke when I added boards

Each board brought its own 120 Ω. Measure A to B with the power off: about 60 Ω is right, under 50 means too many.

Specifications

Chip	MAX3485 (UMW MAX3485ESA)	**Supply and logic**	3.0 to 3.6 V
Fastest	10 Mbit/s	**Termination**	120 Ω across A and B, on every board
Bias	20 kΩ A to VCC, B to GND	**Direction**	RE and DE joined by a 0 Ω link
Receivers	Up to 32 on one bus, one unit load each	**Board**	15.76 × 24.42 mm, headers loose