TBS Fusion Serial Interface

With the TBS Fusion Serial Interface one or more TBS Fusions can be controlled over a single wire or RS-485 connection. It can be used to change the operating frequency, query the current frequency and RSSI (Received Signal Strength Indicator), and perform “RSSI scans” over a wide range of frequencies in the 5.8 GHz band. Multiple TBS Fusions can be connected to the same bus with wire lengths of several hundred meters, enabling the use of the TBS Fusion in applications where remote management is required.

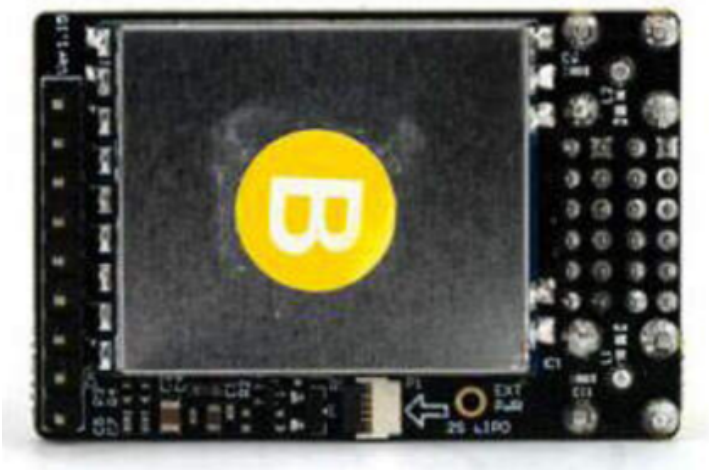
Enabling the Serial Interface

Update the firmware of your TBS Fusion to v2.34 or later to use the serial interface. It needs to be enabled and configured via CRSF (e.g. using the OLED display of the TBS Fusion or TBS Agent). To enable the serial interface, set “Channel Pin Mode” to “Serial”. In addition, one can configure the baud rate (“Serial Baud”) and the address (“Serial Addr”).

Physical Connection

The serial interface uses two of the pins on the back of the TBS Fusion. The pinout is shown below. The serial interface uses the two topmost pins, which are usually used for changing the channel using the buttons on FPV goggles. The TXRX (CH0) pin is used for both receiving and sending data. The TXEN (CH1) pin indicates when the TBS Fusion is sending data and is intending to be used with an external RS-485 transceiver, as explained below. The logic level used by the TBS Fusion is 3.3 V, but the pins are 5 V tolerant and can also be used with 5 V logic.

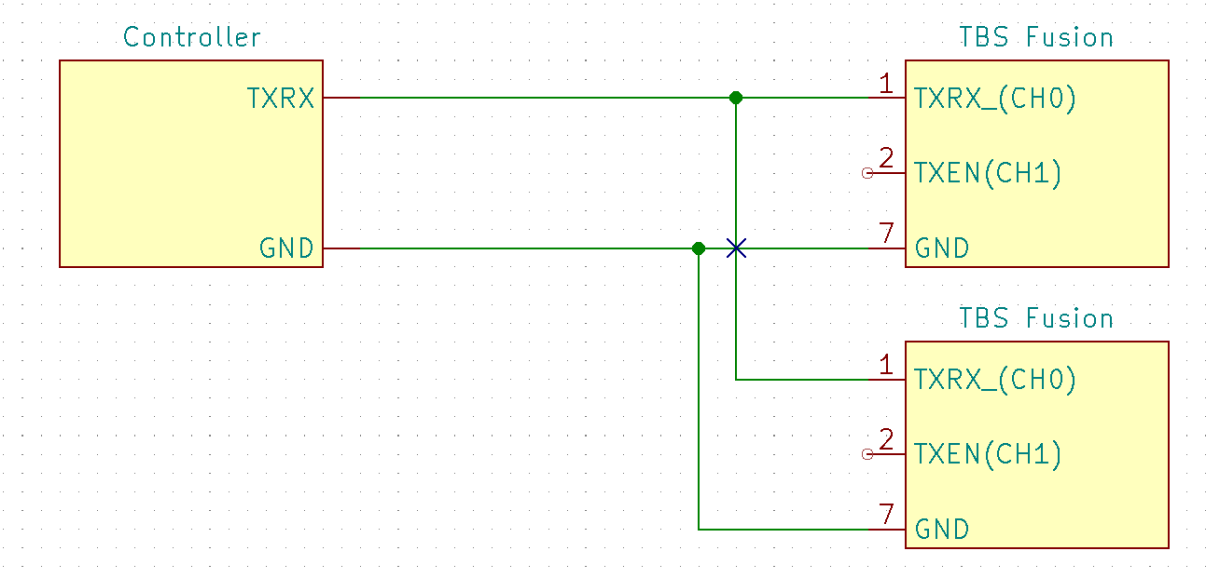
1: TXRX (CH0)
2: TXEN (CH1)
3: CH2
4: Video
5: Audio
6: Audio
7: GND
8: NC
9: 5 V



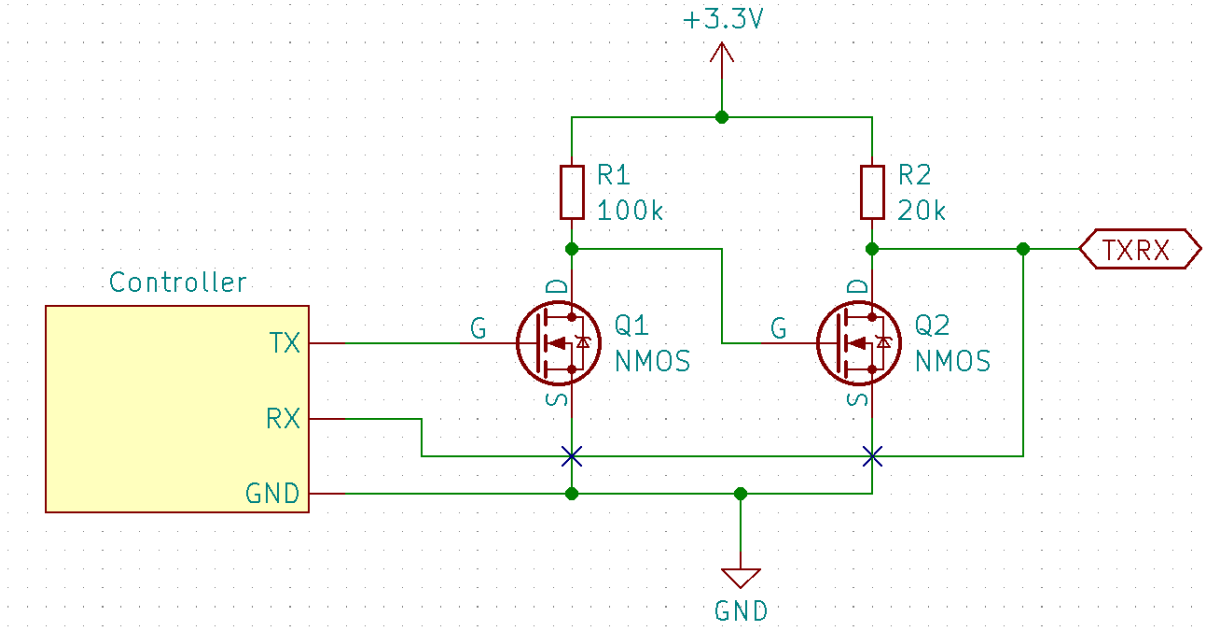
Single Wire Connection

The simplest way of connecting one or more TBS Fusions to a controller is to use only the TXRX signal. The controller could e.g. be a microcontroller or a USB-to-serial converter. Note that this connection is single-ended and the TBS Fusion has an internal 10k Ohm protection resistor, which limits the maximum cable length to a few meters. For controllers

that support sending and receiving asynchronous serial (UART) data on the same pin, the connection is very simple, as shown in the schematic below.



For controllers that have separate TX and RX pins for sending and receiving data, respectively, one option is to use a custom non-inverting open drain circuit using two N-channel MOSFETs and resistors. When selecting MOSFETs, it is important to select one with a gate threshold voltage of less than 3.3 V, such that it works with 3.3 V logic. Note that the controller will receive the data it sends to the TBS Fusion(s) and the controller software needs to discard this data before receiving the response from the TBS Fusion.

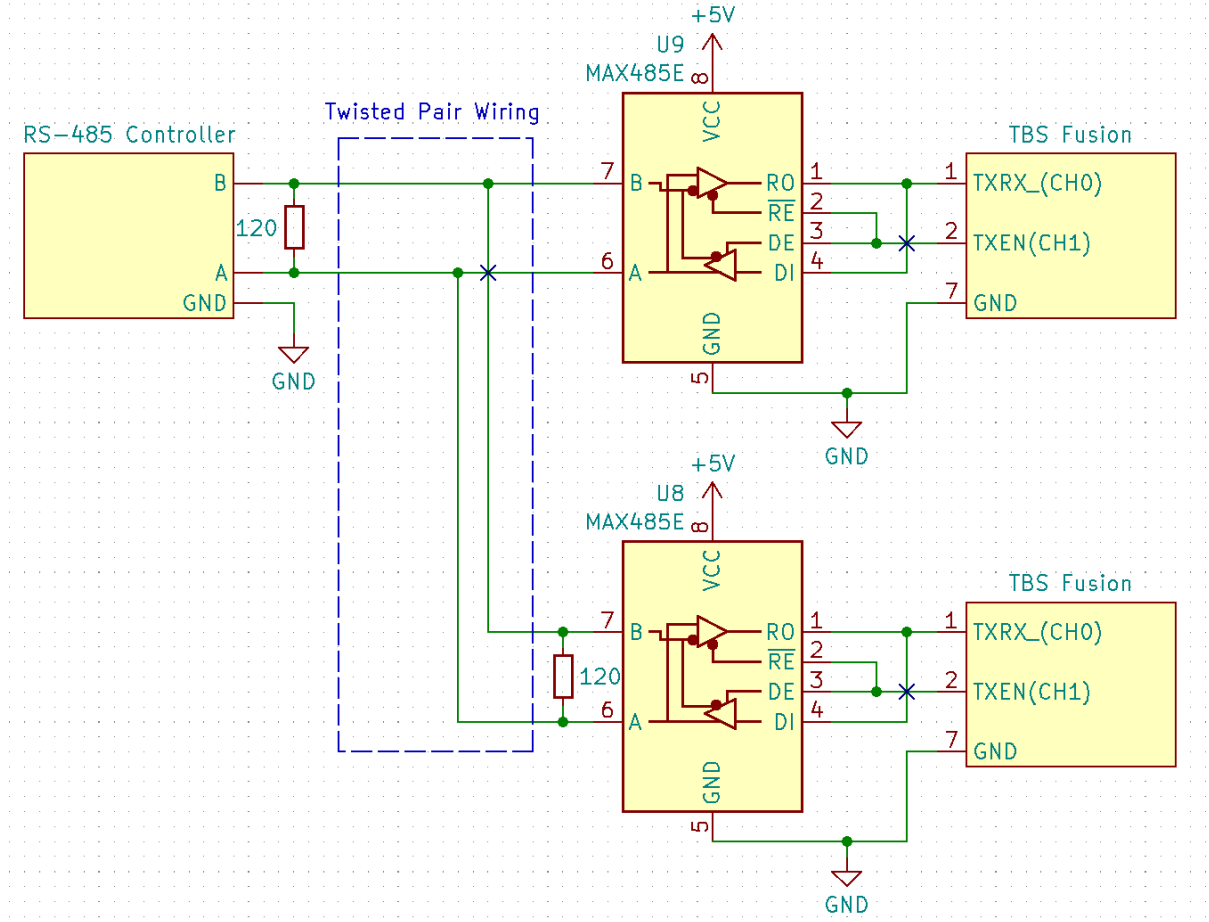


Two Wire RS-485 Connection

The recommended way of controlling one or more TBS Fusions over extended distances is to use a 2 wire RS-485 connection. RS-485 uses differential signaling, which provides noise

immunity and allows for long wires. A good introduction to RS-485 can be found here. When using RS-485, an RS-485 transceiver needs to be connected to each TBS Fusion.

An example of two TBS Fusions connected to an RS-485 controller is shown below. The controller could be a microcontroller with an RS-485 transceiver, or an USB-to-RS-485 interface (e.g. FTDI USB-RS485-WE-1800-BT). On the side of the TBS Fusion, a MAX485SE transceiver is used. The transceiver has A and B pins for the differential RS-485 signals, which are connected to the bus. Both the RO (Receiver Output), DI (Driver Input) pins need to be connected to the TXRN pin of the TBS Fusion and the DE (Driver Enable) and /RE (Not Receiver Enable) to the TXEN pin. The TXEN pin will go high when the TBS Fusion is transmitting data, which enables the RS-485 driver of the MAX485SE. Note also that a common ground (GND) is needed for all devices, so the total wires needed is 3 (A, B, GND).



Additional recommendations when using RS-485:

- The wiring should be twisted pair with 120 Ohm termination resistors at both ends.
- The A/B signal naming can sometimes be confusing. If it does not work, try swapping the A/B wires on the controller side.
- Some cheap MAX485 transceiver breakout boards available online have pull-up resistors on the RO, /RE, DE and DI pins. Those need to be removed. They may also have a 120 Ohm termination resistor, which may need to be removed as well when several TBS Fusions are connected to the same bus.

Asynchronous Serial Protocol

Technical details about the protocol are below. A reference controller implementation in Python is provided in the tbs-fusion-py package.

The serial interface uses a half-duplex asynchronous serial protocol. Multiple baud rates are supported (9600, 19200, 38400, 57600, 115200 bps). They can be configured via CRSF (see above). The serial protocol uses 8 data bits, no parity bit, and one stop bit (8N1).

Message Protocol

Messages sent to the TBS Fusion and responses from the TBS Fusion start with a header with the following contents:

Field	**Length (Bytes)**	**Description**
START CODE	2	Always 0xAA 0x55
CRC	2 (uint16)	Checksum (CRC16)
ADDRESS	1 (uint8)	TBS Fusion address
TYPE	1 (uint8)	Type of message
LENGTH	1 (uint8)	Length of payload in bytes

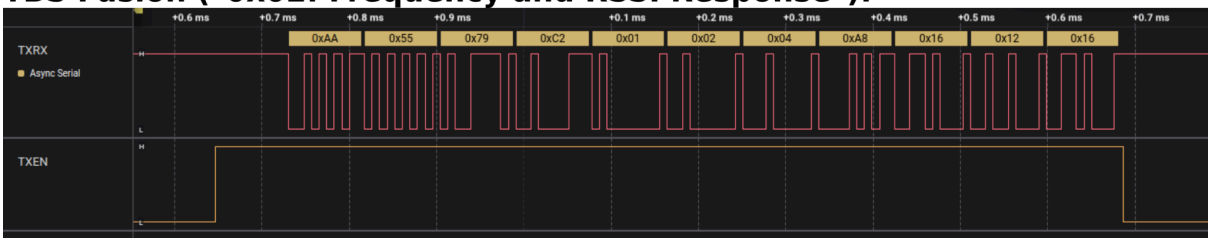
Note: All multi-byte values (CRC, LENGTH, etc.) use little-endian encoding.

The header is followed by a payload with a length of LENGTH bytes. The content of the payload depends on the type of message, as described below. The CRC is computed from the start of ADDRESS to the end of the payload. The type of CRC used is “CRC16-CCIT-FALSE” (width: 16 bits, polynomial: 0x1021, reflect in: False, reflect out: False, xor-in: 0xFFFF xor-out: 0x0000).

The TBS Fusion will only transmit messages if it has received a command or request from the controller. An example of a message exchange is shown below. The controller sends a “0x01: Frequency and RSSI Request” and the TBS Fusion responds with a “0x02: Frequency and RSSI Response” (see below for details).

Controller ("0x01: Frequency and RSSI Request"):

TBS Fusion ("0x02: Frequency and RSSI Response"):



Message Types

0x00: Message Acknowledgement

This message is sent by the Fusion in response to messages that do not request data from the Fusion, such as “0x03: Command Set Frequency”. It is also used to indicate errors in response to other messages (e.g. invalid commands or parameters). The payload of this message consists of two bytes:

Field	Length (Bytes)	Description
MESSAGE TYPE	1 (uint8)	The message type the acknowledgment is for.
RETURN CODE	1 (uint8)	Return code. Values other than 0 indicate an error.

0x01: Frequency and RSSI Request

This message gets the current frequency and the RSSI of each receiver. The message does not have a payload. The response to this message is “0x02: Frequency and RSSI Response”. Note: An example of this message is shown in the image above.

0x02: Frequency and RSSI Response

This message is sent by the Fusion in response to 0x01. The message has the following payload:

Field	Length (Bytes)	Description
FREQUENCY	2 (uint16)	Frequency in MHz
RSSI A	1 (uint8)	RSSI receiver A (scaled to 0 .. 255)
RSSI B	1 (uint8)	RSSI receiver B (scaled to 0 .. 255)

Note: An example of this message is shown in the image above. In the example, FREQUENCY is 5800 (0x16A8), RSSI A is 0x12, and RSSI B is 0x16.

0x03: Command Set Frequency
This command sets the operating frequency of the Fusion. The payload consists of the frequency, as shown below. The Fusion sends an 0x00 message to acknowledge the receipt of the message.

Field	Length (Bytes)	Description
FREQUENCY	2 (uint16)	Frequency in MHz. Minimum: 4900 Maximum: 6200.

0x04: Frequency Range Scan Request
This command initiates a frequency scan, i.e. the Fusion will measure the RSSI (signal strength) over a range of frequencies and then returns it using a “0x06: Frequency Scan Response” message. The payload of the request is as follows:

Field	Length (Bytes)	Description
START FREQUENCY	2 (uint16)	Start frequency in MHz. Minimum: 4900 Maximum: 6200.
STOP FREQUENCY	2 (uint16)	Stop frequency in MHz. Minimum: 4900 Maximum: 6200.
FREQUENCY STEP	1 (uint8)	Step size in MHz.
RECEIVER	1	Which receiver to use for the scan: 0: Both receivers (faster) 1: Receiver A 2: Receiver B
DELAY	1 (uint8)	Delay in milliseconds between setting the frequency and the RSSI measurement. If the delay is too small, the RSSI will not be stable. Recommended is 25 ms.

Note that the scan starts at START FREQUENCY, but STOP FREQUENCY is not included, i.e., it behaves like the range type in Python.

The maximum number of frequencies that can be scanned is 100. With a delay of 25 ms and both receivers, the time needed to perform the scan is about 1.6 seconds. When only one receiver is used, the time doubles to about 3.2 seconds.

0x05: Frequency List Scan Request
This command initiates a frequency scan over an arbitrary list of frequencies. The TBS Fusion responds with a "0x06: Frequency Scan Response" message. The payload of the request is as follows:

Field	Length (Bytes)	Description
RECEIVER	1	Which receiver to use for the scan: 0: Both receivers (faster) 1: Receiver A 2: Receiver B
DELAY	1 (uint8)	Delay in milliseconds between setting the frequency and the RSSI measurement. If the delay is too small, the RSSI will not be stable. Recommended is 40 ms for large frequency steps.
FREQUENCIES	N * 2 (uint16)	Frequencies to scan.

The maximum number of frequencies (N) is 100. Note that when the frequency steps are very large, a DELAY of about 40 ms is needed for the RSSI to be stable.

0x06: Frequency Scan Response
The Fusion sends this message in response to a "0x04: Frequency Scan Request". The payload of the message is an array of RSSI values encoded as 1 byte unsigned integers (uint8) with one value per requested frequency point.

Field	Length (Bytes)	Description
RSSI	N * 1 (uint8)	RSSI scaled to (scaled to 0 .. 255)

Note: When both receivers are used, the RSSI values are interleaved ("RSSI A, RSSI B, RSSI A, …").