QUICK REFERENCE · TK109

CAN Bus

lonely binary
open-source is love

An SN65HVD230 transceiver for a microcontroller with its own CAN controller, such as an ESP32.

3.3 V only: never 5 V

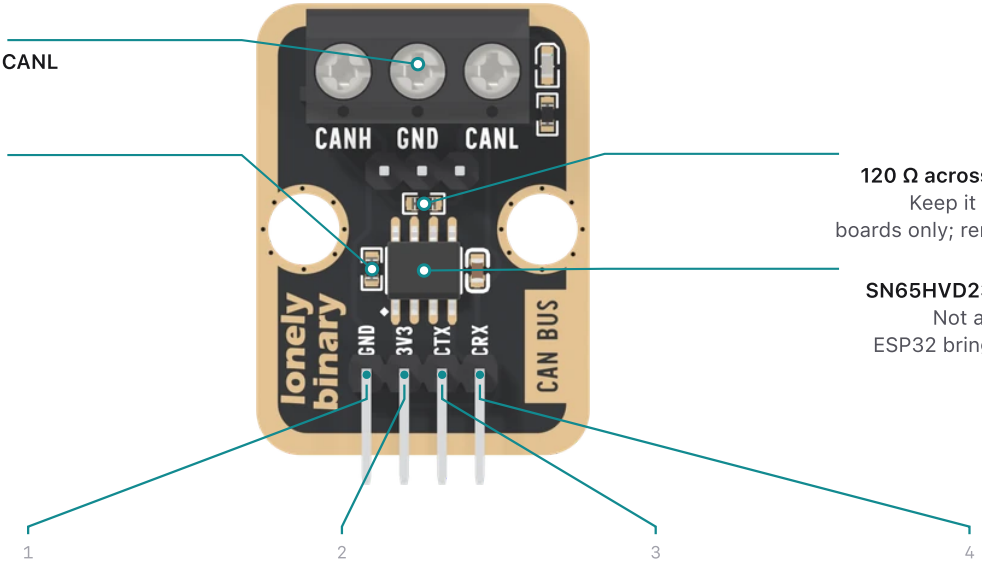
The chip runs from 3.0 to 3.6 V, and CTX and CRX take at most 0.5 V above its supply. **Power it from 3V3, and drive it only from 3.3 V logic.** An Uno has no CAN controller and 5 V pins.

Pins, and which to wire

Drawn from above, terminal at the top. CTX and CRX are named from the microcontroller's side, so they go **straight across**: CTX to its CAN TX pin, CRX to its CAN RX pin. CANH, GND and CANL are on the terminal and again on the three pins below it.

Screw terminal: CANH GND CANL
The same nets as the three pins below. GND runs too.

10 kΩ on Rs: slope control
Slower edges, a quieter bus. Tested at 500 kbit/s.



120 Ω across CANH, CANL
Keep it on the two end boards only; remove between.

SN65HVD230 transceiver
Not a controller: the ESP32 brings that (TWAI).

Front	GND	3V3	CTX	CRX
Back	GND	3V3	CTX	CRX
ESP32-S3	GND	3V3	TX	RX

One board alone never finishes a message

Every CAN frame must be acknowledged by another board. Alone, a sender resends each frame and adds 8 to its error count every time, up to 128. Nothing is broken: test with two boards at one bit rate.

The 120 Ω, on how many boards

The two end boards

60 Ω across CANH, CANL

25 mA for 1.5 V dominant: the load the driver is built for.

3 fitted

40 Ω across CANH, CANL

38 mA for 1.5 V dominant: past the 60 Ω the driver is built for.

All six fitted

20 Ω across CANH, CANL

75 mA for 1.5 V dominant: past the 60 Ω the driver is built for.

Every board ships with its 120 Ω fitted. Keep it on **the two boards at the ends of the cable**. With none, the bus cannot fall back to recessive, which is not driven; with every one fitted, the driver pushes into a third of its load.

Arithmetic, not measurements: n resistors of 120 Ω in parallel, and the current for the 1.5 V minimum dominant difference in TI's datasheet (SLOS346O), which specifies the driver into 60 Ω. The fitted chip is Tokmas's SN65HVD230DR, a second source. Tested on two ESP32-S3 boards at 500 kbit/s.

Two ESP32-S3, one sketch

Arduino IDE · Board: ESP32S3 Dev Module · USB CDC On Boot: Enabled · No libraries

```
// SENDER true on one board, false on the other.
#include "driver/twai.h"

const bool SENDER = true;
const int CAN_TX = 8, CAN_RX = 9;

void setup() {
  Serial.begin(115200);
  delay(500);
  twai_general_config_t g = TWAI_GENERAL_CONFIG_DEFAULT(
      (gpio_num_t)CAN_TX, (gpio_num_t)CAN_RX, TWAI_MODE_NORMAL);
  twai_timing_config_t t = TWAI_TIMING_CONFIG_500KBITS();
  twai_filter_config_t f = TWAI_FILTER_CONFIG_ACCEPT_ALL();
  twai_driver_install(&g, &t, &f);
  twai_start();
}

void loop() {
  if (SENDER) {
    static uint8_t count = 0;
    twai_message_t msg = {};
    msg.identifier = 0x100;
    msg.data_length_code = 1;
    msg.data[0] = count;
    twai_transmit(&msg, pdMS_TO_TICKS(100));
    delay(1000);
    twai_status_info_t s;
    twai_get_status_info(&s);
    Serial.printf("sent %u, tx errors %lu\n", count++,
                  (unsigned long)s.tx_error_counter);
  } else {
    twai_message_t msg;
    if (twai_receive(&msg, pdMS_TO_TICKS(1000)) == ESP_OK)
      Serial.printf("got 0x%03lX: %u\n",
                    (unsigned long)msg.identifier, msg.data[0]);
  }
}
```

```
got 0x100: 0
got 0x100: 1
```

The receiver's Serial Monitor at 115200. The sender prints sent 0, tx errors 0: zero errors means every frame was acknowledged.

Wiring

	FRONT	EACH ESP32-S3
1	**GND**	**GND**
2	**3V3**	**3V3**
3	**CTX**	**GPIO 8**
4	**CRX**	**GPIO 9**

Each ESP32-S3 to its own TK109, then board to board: CANH to CANH, GND to GND, CANL to CANL. Keep both 120 Ω fitted: two boards are the two ends. On a classic ESP32, GPIO 8 and 9 belong to the flash chip: use two free pins such as 5 and 4.

Ground and a twisted pair

The receiver reads CANH – CANL, but it needs both wires within -2 V to +7 V of its own ground. Boards on separate supplies share no ground until a wire gives them one, so run GND with the bus. Put CANH and CANL on one twisted pair, so noise lands on both alike and cancels. Keep each branch from the cable to a board short, and the bus one line from end to end.

When it does not work

The sender's error count climbs to 128

Nobody acknowledges. Start the second board's sketch, and check its CRX reaches its RX pin and the bus wires join.

Errors from the very first frame

The two sketches set different bit rates, or CANH and CANL are crossed between the boards.

It broke when I added boards

Too many 120 Ω. CANH to CANL with the power off: about 60 Ω is right.

It prints nothing, not even errors

The pins in the sketch are not the pins wired, or the driver never started. Match them.

Fine on the desk, not on the cable

Run GND with the pair, twist CANH and CANL together, and keep the 120 Ω on the two end boards.

Nothing works with no resistors fitted

The bus cannot get back to recessive. Keep a 120 Ω at each end of the cable.

Specifications

Chip	SN65HVD230 (Tokmas), SOIC-8	**Supply and logic**	3.0 to 3.6 V
Host	A CAN controller: every ESP32	**Bus pins**	-2 to +7 V, ±16 kV ESD
Termination	120 Ω, fitted on every board	**Speed**	Tested at 500 kbit/s
Edges	10 kΩ slope control	**Board**	22.4 × 30.4 mm, headers fitted