EEPROM Memory

16,384 bytes that keep their values with the power off, on two wires.

VCC goes to your board's logic supply

The pull-ups on SDA and SCL end at VCC, so the bus rests at whatever VCC is. **5V on an Uno, 3V3 on an ESP32, ESP32-S3 or Pico.** 5V on VCC beside a 3.3 V board puts 5 V on two of its pins.

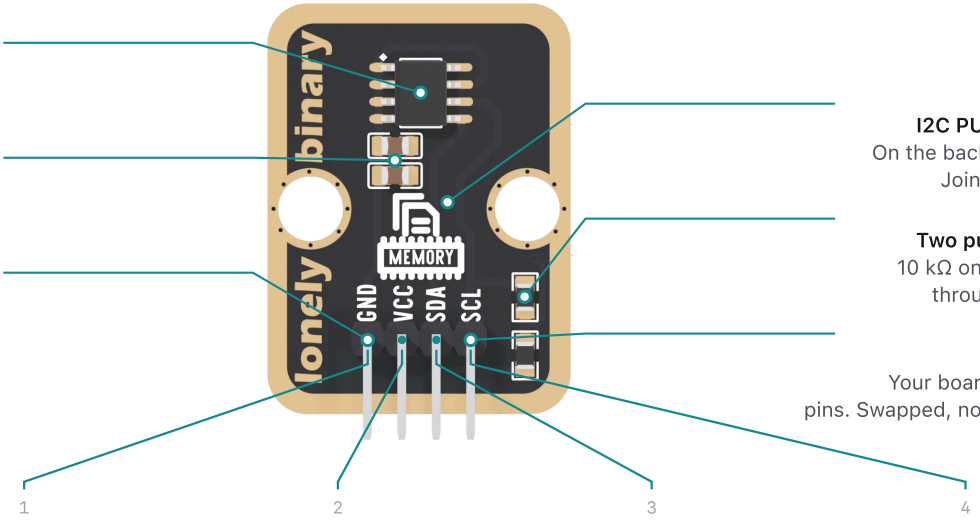
Pins, and which to wire

Parts up, header at the bottom. All four pins are wired. There is no light: the chip acknowledging 0x50 is the sign it works.

AT24C128 at 0x50
16,384 bytes. Address fixed: one TK31 per bus.

Supply capacitors
100 nF and 10 µF, for the write bursts.

GND is the square pad
Count from it. The back prints no pin names.



I2C PULL-UP jumper
On the back, behind here. Joined as shipped.

Two pull-ups to VCC
10 kΩ on SDA and SCL, through the jumper.

SCL and SDA
Your board's default I2C pins. Swapped, nothing answers.

Front	GND	VCC	SDA	SCL
Arduino	GND	VCC	SDA	SCL
MicroPython	GND	VCC	SDA	SCL

It answers nobody while it writes

After every write the chip spends up to 5 ms storing it and acknowledges nothing, not even 0x50. A read in that time fails or returns the old value. Wait after every write.

Three rules for every write

Wait after a write

5 ms at most

delay(5), or ask 0x50 until it acknowledges. The datasheet gives only the maximum.

Stay inside a page

64 bytes a page

A write past a multiple of 64 wraps to the page's start and overwrites it. On an Uno, 30 bytes a write at most.

Write on change

1M writes a byte

Ten writes a second wears a byte out in about 28 hours. Read first; write only if it differs.

Send two address bytes, high then low, wait after every write, cut long writes at page edges, and write a value only when it changes. Reads cost nothing.

From the HXY AT24C128 datasheet: 2.2 to 5.5 V, 5 ms maximum write, 1,000,000 writes a byte at 25 °C and 5 V, 40-year retention. Wear times are worked out from that rating, not measured. Filling the chip at 100 kHz takes about 88 s a byte at a time and 7 s in 16-byte pieces.

Save a value, only when it changes

Arduino IDE · Board: your board · ESP32-S3: USB CDC On Boot: Enabled

```
#include <Wire.h>
const int EEPROM_ADDR = 0x50;       // fixed on the TK31

int readByte(uint16_t at) {
  Wire.beginTransmission(EEPROM_ADDR);
  Wire.write((uint8_t)(at >> 8));   // address, high byte
  Wire.write((uint8_t)(at & 0xFF)); // then low byte
  Wire.endTransmission(false);
  Wire.requestFrom(EEPROM_ADDR, 1);
  return Wire.available() ? Wire.read() : -1;
}

void update(uint16_t at, uint8_t value) {
  if (readByte(at) == value) return; // same: no wear
  Wire.beginTransmission(EEPROM_ADDR);
  Wire.write((uint8_t)(at >> 8));
  Wire.write((uint8_t)(at & 0xFF));
  Wire.write(value);
  Wire.endTransmission();
  delay(5);                         // the chip is writing
}

void setup() {
  Serial.begin(115200);
  Wire.begin();                     // default I2C pins
  int starts = readByte(0);
  if (starts == 255) starts = 0;    // never written
  update(0, starts + 1);
  Serial.println(starts + 1);
}

void loop() {
}
```

```
1
2
3
```

Serial Monitor at 115200, one line per start: unplug the board and plug it back in, and the count carries on.

Wiring

	FRONT	UNO	ESP32	S3	PICO
1	**GND**	GND	GND	GND	GND
2	**VCC**	5V	3V3	3V3	3V3(OUT)
3	**SDA**	A4	GPIO 21	GPIO 8	GP4
4	**SCL**	A5	GPIO 22	GPIO 9	GP5

Hole 1 is the square pad. VCC is the board's logic supply, never 5V beside a 3.3 V board. The SDA and SCL pins are each board's defaults, so Wire.begin() needs no arguments.

In MicroPython

On an ESP32, ESP32-S3 or Pico: `i2c = I2C(0, sda=Pin(21), scl=Pin(22))` (S3: 8 and 9; Pico: 4 and 5), then `i2c.writeto_mem(0x50, 0, b"\x2a", addrsize=16)`, `time.sleep_ms(5)`, and `i2c.readfrom_mem(0x50, 0, 1, addrsize=16)`. Leave out addrsize=16 and MicroPython sends one address byte, which this chip reads as the wrong place.

When it does not work

Nothing answers at 0x50

Count from the square pad: GND, VCC, SDA, SCL. SDA and SCL swapped is the usual fault.

Every byte reads 255

The sketch sends one address byte. This chip needs two, high then low.

One byte will not hold

Wear: a write in loop() on every pass. Write only on change, and move it.

Reads back the old value

The read went out during the 5 ms write. Wait after every write.

Long text comes back mixed

A write crossed a 64-byte page edge and wrapped. Cut writes at page edges.

Two TK31s on one bus

Both answer at 0x50, and the address cannot move. One TK31 per bus.

Specifications

Chip	HXY AT24C128, SOIC-8	**Memory**	16,384 bytes, 64-byte pages
Address	0x50, fixed; two address bytes	**Supply**	2.2 to 5.5 V: your logic supply
Write	5 ms max; 1,000,000 a byte	**Retention**	40 years
Pull-ups	10 kΩ to VCC, cut jumper	**Clock**	400 kHz from 2.2 V, 1 MHz from 2.5 V