

Was ist der I2C-Bus?

Quelle: <https://de.i2c-bus.org/>

Dieser Bus wurde in den frühen 80er Jahren von Philips für die einfache Kommunikation von Bausteinen entwickelt, welche sich auf einer gemeinsamen Platine befinden.

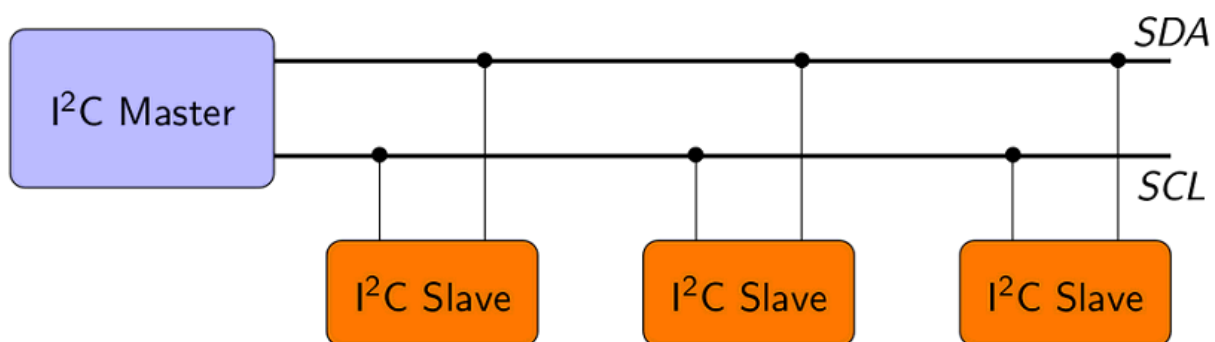
Der I2C-Bus wird gelegentlich auch als IIC-Bus oder I²C-Bus genannt, die Bezeichnung spielt auf Inter-IC an. Ursprünglich wurde der Bus für eine Übertragungsrate von 100 kbit/s definiert, was für viele Anwendungen ausreichend ist. Ferner existiert eine 400 kbit- und seit 1998 eine Highspeed 3.4 Mbit-Variante.

Geeignet ist der I2C-Bus auch dann, wenn sich Komponenten nicht auf derselben Platine befinden, sondern über Kabel miteinander verbunden sind. Die Flexibilität und vergleichsweise einfache Implementierung machen den I2C-Bus für viele Applikationen attraktiv.

I2C ist als Master-Slave-Bus konzipiert. Ein Datentransfer wird immer durch einen Master initiiert; der über eine Adresse angesprochene Slave reagiert darauf. Mehrere Master sind möglich (Multimaster-Mode). Im Multimaster-Mode können zwei Master-Geräte direkt miteinander kommunizieren, dabei arbeitet ein Gerät als Slave. Die Arbitrierung (Zugriffsregelung auf den Bus) ist per Spezifikation geregelt.

Die Haupteigenschaften sind:

- Es werden nur zwei Busleitungen benötigt
- Es gibt keine harten Anforderungen an eine exakte Übertragungsrate.
- Zwischen allen Teilnehmern besteht eine einfache Master-Slave-Beziehung
- Jede Komponente hat eine eindeutige Adresse
- I2C ist ein echter Multimasterbus mit Kollisionsbehandlung und Zugriffsarbitrierung



Pin-Belegung:

- GND: Gemeinsamer Ground für Spannungs-Versorgung und Logik.
- VCC: Eingangs- oder Betriebsspannung 3,3V. Gleiche Spannung wie Micro:Bit.
- SCL: P19 i2c Takt (Clock)
- SDA: P20 i2c Serial Data in/out

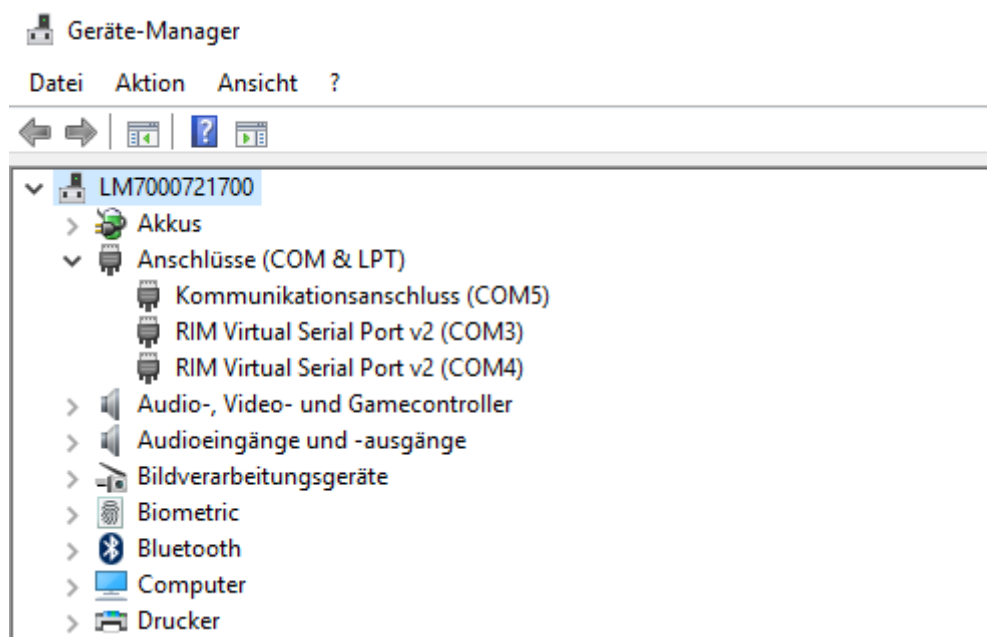
Micro:Bit und Fehlersuche

Wenn nun der angeschlossene Bauteil nicht reagiert oder die Adresse nicht bekannt ist, dann kann man folgendermaßen vorgehen um schrittweise Informationen zu erhalten.

- Öffne den Geräte-Manager und beobachte die COM Anschlüsse beim Anstecken des micro:bit.
- Stecke den micro:bit an. Nun wird seine zusätzliche COM mit Nummer angezeigt.
- Benutze die Gratis-Software ARDUINO: <https://www.arduino.cc/en/Main/Software>
- Starte die Software ARDUINO
- Im Menü Werkzeuge – Port die entsprechende COM vom micro:bit auswählen.
- Rechts oben den Seriellen Monitor starten.
- Den Python Editor aus <https://microbit.org/code/> starten.
- Das Testprogramm zum Auslesen der angeschlossenen Bauteil-Adressen starten.
- Mit `print(„Text“)` wird der Text im Seriellen Monitor angezeigt.

Micro:Bit Schritt für Schritt

- Öffne den Geräte-Manager und beobachte die COM Anschlüsse beim Anstecken des micro:bit.



- Stecke den micro:bit an. Nun wird seine zusätzliche COM mit Nummer angezeigt.
- Benutze die Gratis-Software ARDUINO: <https://www.arduino.cc/en/Main/Software>
- Starte die Software ARDUINO

Download the Arduino IDE

ARDUINO 1.8.5
 The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the Getting Started page for installation instructions.

Windows Installer, for Windows XP and up
 Windows ZIP file for non-admin install

Windows app Requires Win 8.1 or 10
 Get it

Mac OS X 10.7 Lion or newer

Linux 32 bits
 Linux 64 bits
 Linux ARM

Release Notes
 Source Code
 Checksums (sha512)

- Im Menü Werkzeuge – Port die entsprechende COM vom micro:bit auswählen.

Blink | Arduino 1.8.5

Datei Bearbeiten Sketch Werkzeuge Hilfe

Automatische Formatierung Strg+T
 Sketch archivieren
 Kodierung korrigieren & neu laden
 Serieller Monitor Strg+Umschalt+M
 Serieller Plotter Strg+Umschalt+L

WiFi101 Firmware Updater

Board: "Generic ESP8266 Module" >
 Flash Mode: "QIO" >
 Flash Size: "512K (no SPIFFS)" >
 Debug port: "Disabled" >
 Debug Level: "Keine" >
 I2C Variant: "v2 Prebuilt (MSS=536)" >
 Reset Method: "ck" >
 Crystal Frequency: "26 MHz" >
 Flash Frequency: "40MHz" >
 CPU Frequency: "80 MHz" >
 Upload Speed: "115200" >
 Port >
 Boardinformationen holen >
 Programmer: "AVRISP mkII" >
 Bootloader brennen >

- Rechts oben den Seriellen Monitor starten.

COM14

Send

Autoscroll

Both NL & CR 115200 baud Clear output

- Den Python Editor aus <https://microbit.org/code/> starten.

Python Editor

Our Python editor is perfect for those who want to push their coding skills further. A selection of snippets and a range of premade images and music give you a helping hand with your code. Powered by the global Python Community.

Let's Code
Reference



- Das Testprogramm zum Auslesen der angeschlossenen Bauteil-Adressen starten.

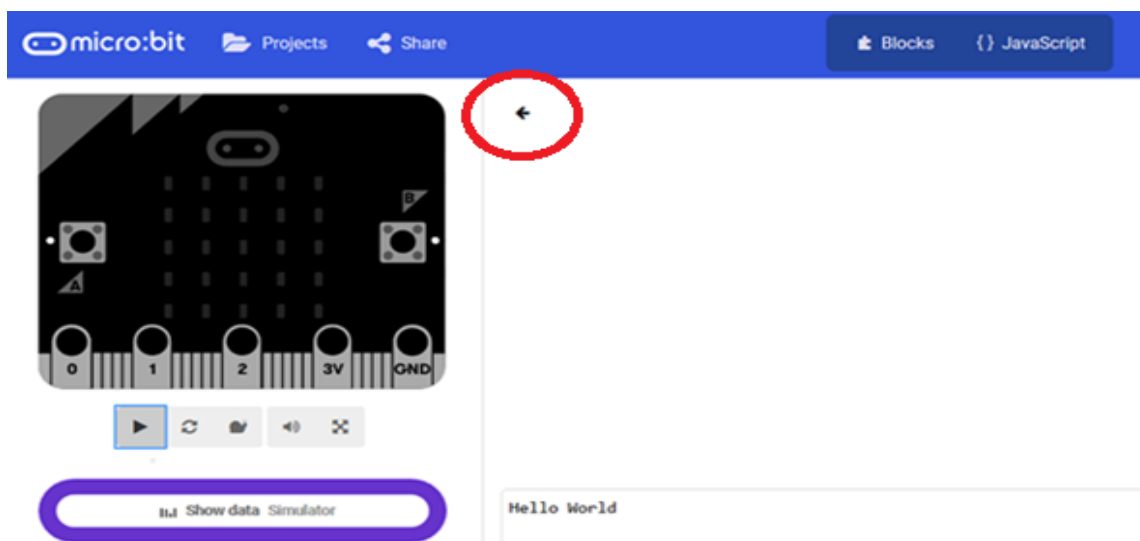
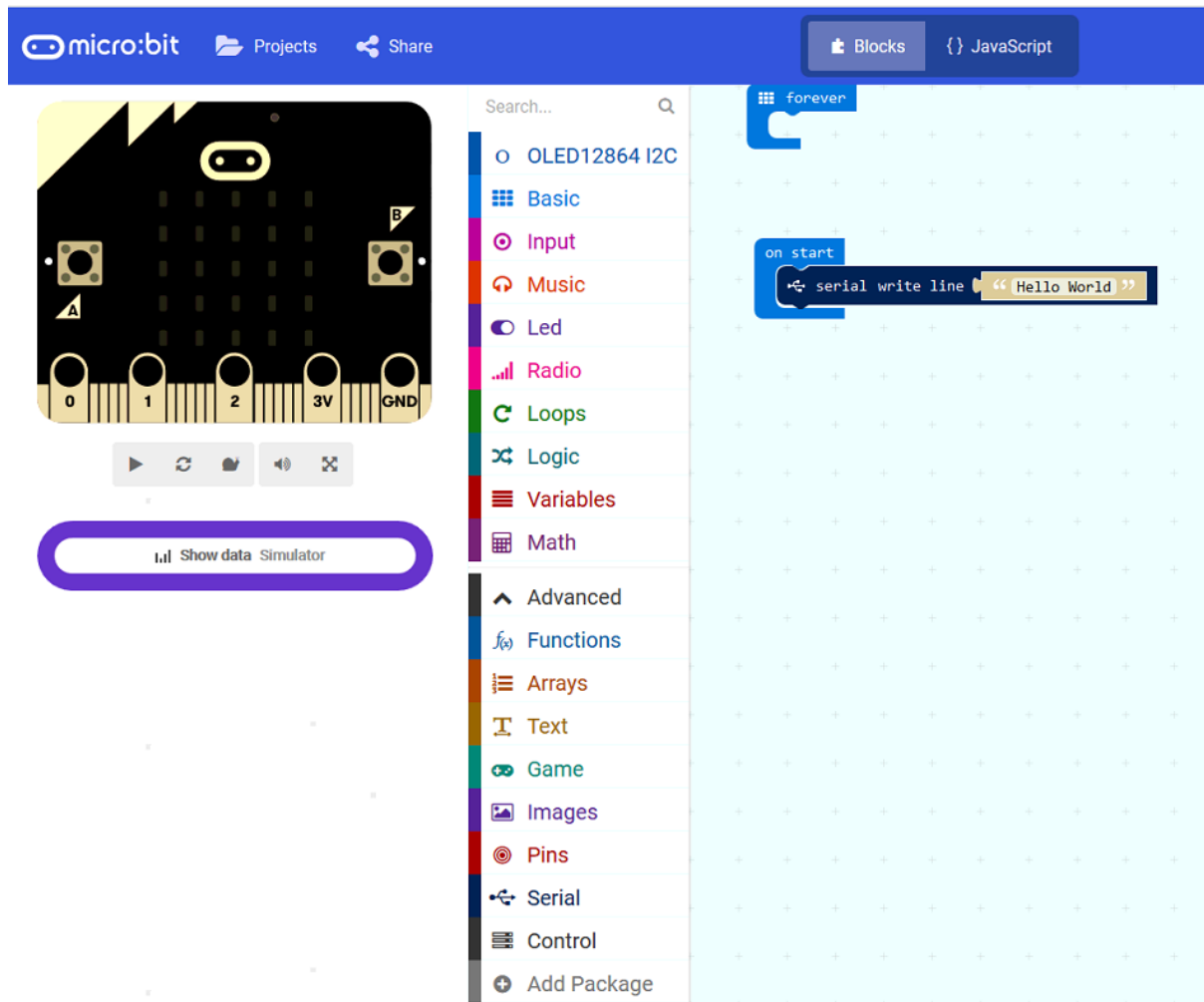


- Mit print(„Text“) wird der Text im Seriellen Monitor angezeigt.



Micro:Bit Blocks und serielle Ausgabe

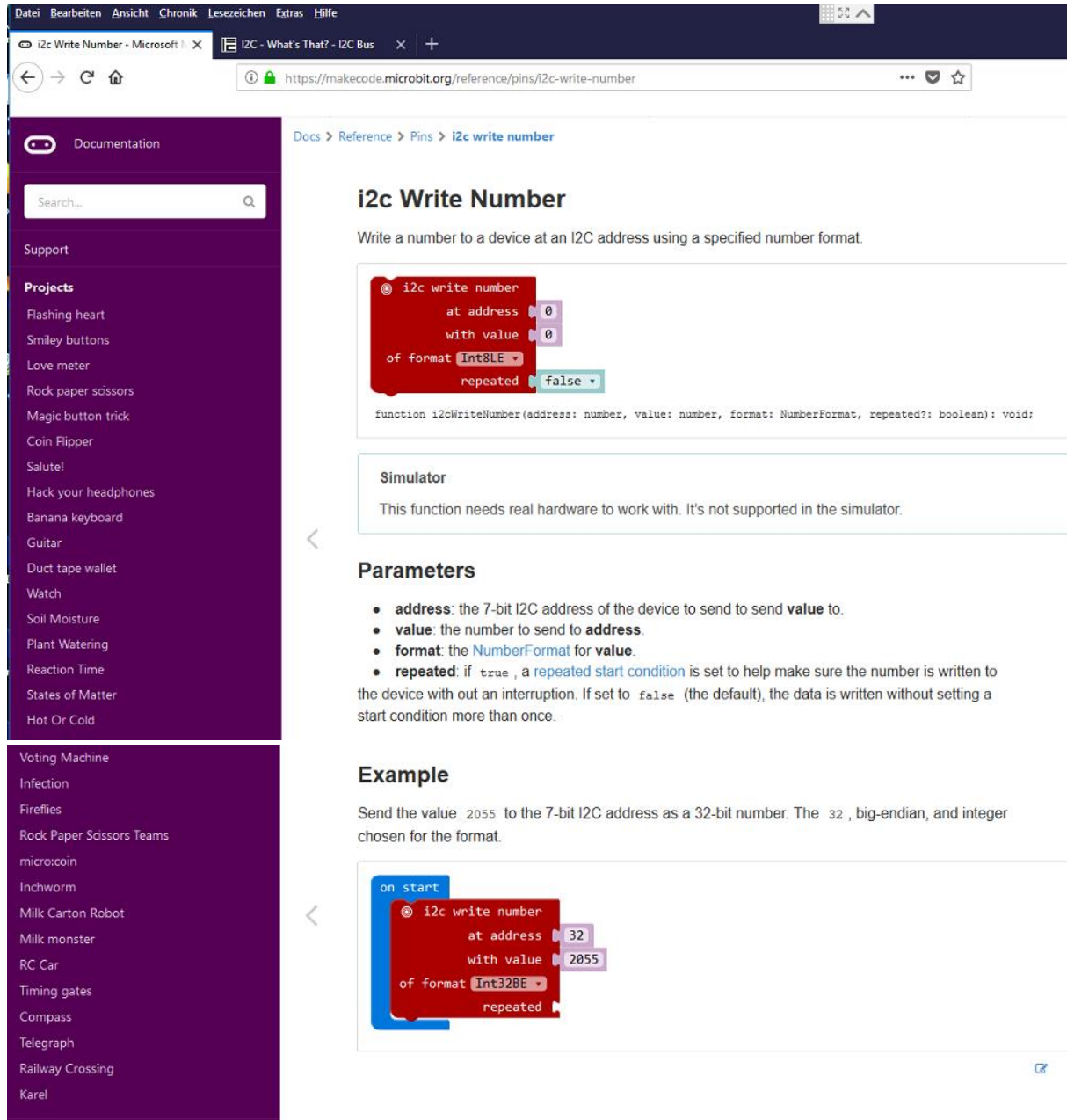
Beim Verwenden der Bibliothek „Serial“ wird der „Show data Simulator“ aktiv.



Mit dem Pfeil kommt man wieder zurück zur Block-Ansicht.

Micro:Bit Herausforderung

Benutze die i2c Blöcke aus Bibliothek „Pins – more“ und erkunde die Funktionen. Verwende dazu die Dokumentation in <https://makecode.microbit.org/reference/...>



The screenshot shows a web browser displaying the documentation for the 'i2c write number' block. The browser address bar shows <https://makecode.microbit.org/reference/pins/i2c-write-number>. The page has a dark purple sidebar on the left with a search bar and a list of projects. The main content area is white and contains the following information:

- Navigation:** Docs > Reference > Pins > i2c write number
- Section Header:** i2c Write Number
- Description:** Write a number to a device at an I2C address using a specified number format.
- Block Diagram:** A red block 'i2c write number' with four fields: 'at address' (0), 'with value' (0), 'of format' (Int8LE), and 'repeated' (false).
- Function Signature:** `function i2cWriteNumber(address: number, value: number, format: NumberFormat, repeated?: boolean): void;`
- Simulator:** A note stating 'This function needs real hardware to work with. It's not supported in the simulator.'
- Parameters:**
 - address:** the 7-bit I2C address of the device to send to send **value** to.
 - value:** the number to send to **address**.
 - format:** the `NumberFormat` for **value**.
 - repeated:** if `true`, a **repeated start condition** is set to help make sure the number is written to the device with out an interruption. If set to `false` (the default), the data is written without setting a start condition more than once.
- Example:** Send the value 2055 to the 7-bit I2C address as a 32-bit number. The 32, big-endian, and integer chosen for the format.
- Example Block Diagram:** A red block 'i2c write number' with four fields: 'at address' (32), 'with value' (2055), 'of format' (Int32BE), and 'repeated' (false).