FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

DIE HOCHSCHULE,
DIE MEHR KANN.

Serial Communication

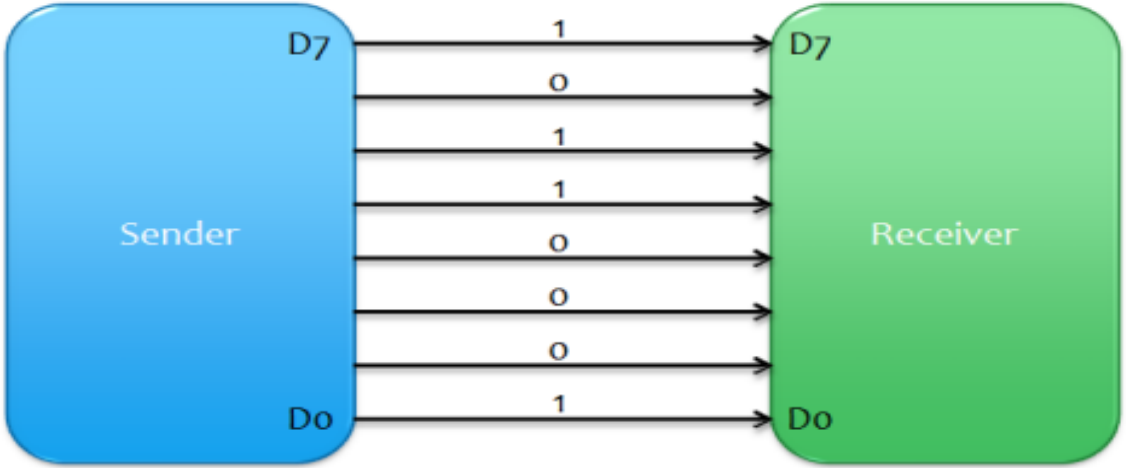
FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Serial Communication

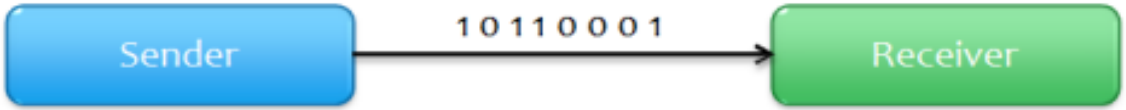
>Data transmission
>USART fundamentals
>SPI
>I2C
>PS/2 Keyboard

https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
http://maxembedded.com/2013/09/serial-communication-introduction/
https://upload.wikimedia.org/wikipedia/commons/1/1f/Serial_Programming.pdf

Parallel Transfer



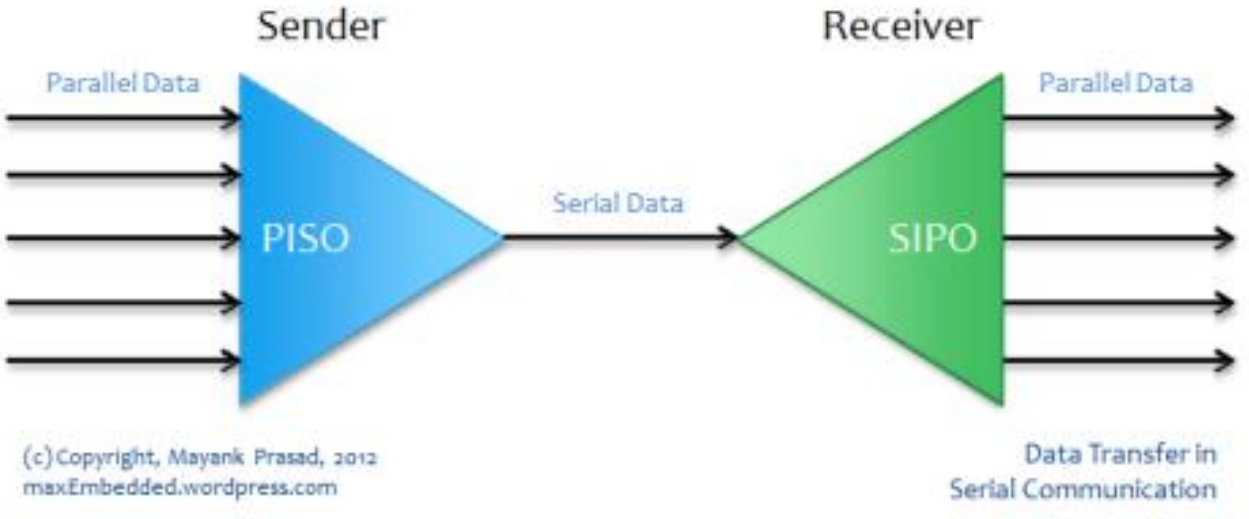
Serial Transfer



Data Transmission

Inside the CPU or Microcontroller data is transmitted in parallel using buses (8/16/32 Bit).

For long distances we use a single wire and serial transmission.



Data Receive

How long does a Bit last?
-> Transmission speed, Bitrate, Baudrate (Bit/sec)

When does the transmission begin / end?

Synchronus Transmission
-> Clock signal -> additional wire necessary

Asynchronus Transmission
-> Start-Bit & Stop-Bit
-> fixed datarate (known by sender and receiver)

USART - Universal Synchronus / Asynchronus Receiver Transmitter

RxD – Receive Data

TxD – Transmit Data

GND

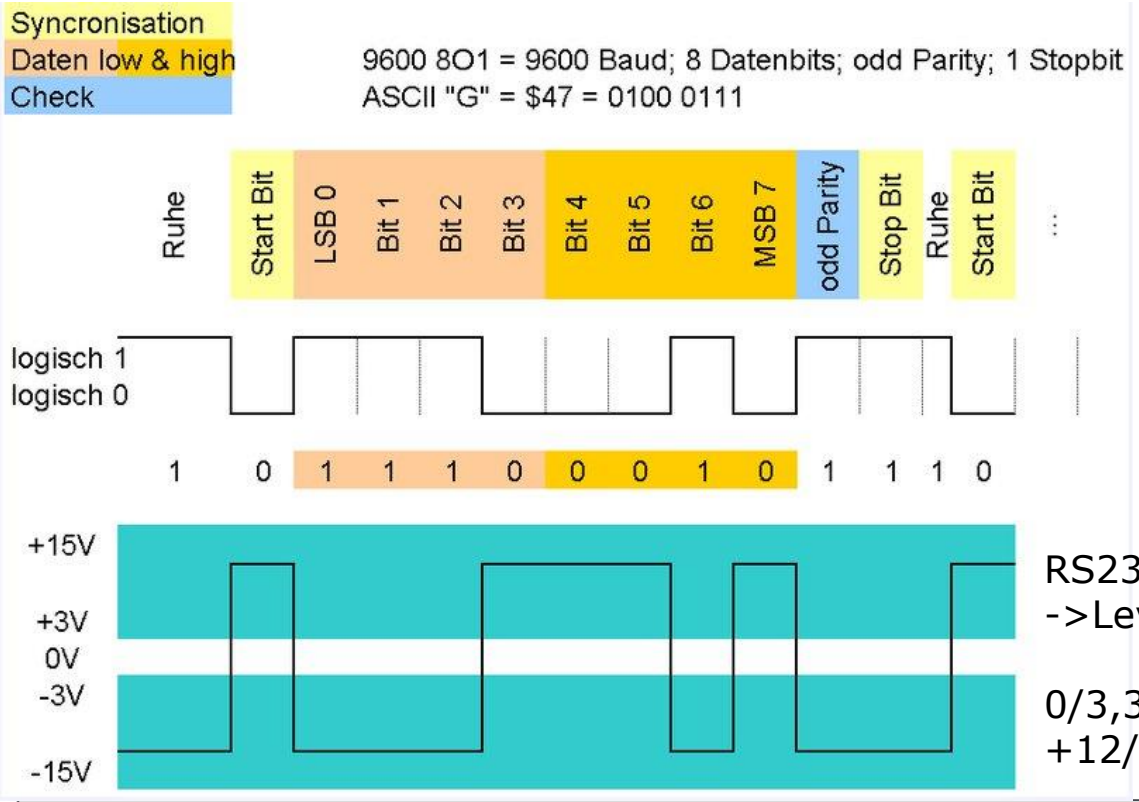
Point-to-Point Connection / Transmission

The Transmit-Pin (Tx) from the Sender hast to be connected with the Receive-Pin (Rx) from the Receiver.

Asynchronus works without a clock signal that shows when the data is valid. As a result both microcontrollers need to use the same data transmission rate well known as the baudrate !
(Typical values are 9600 // 19200 // 57600 // 115200)

Additional Control Lines -> RS232

Universal Synchronus / Asynchronus Receiver Transmitter



RS232
->Level shifter

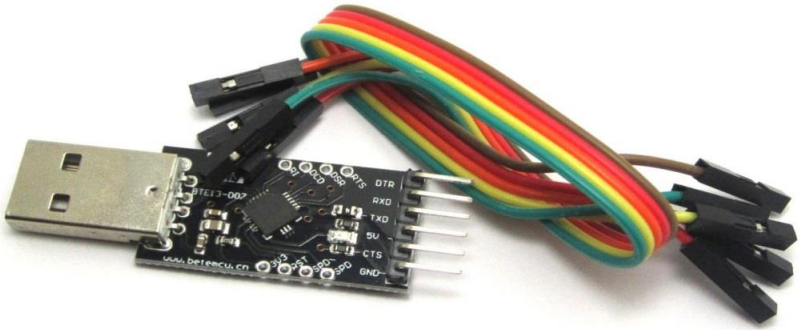
0/3,3V ->
+12/-12V

RS232 – PC - Modem

classic 2 wire
baudrate 56kBit/sec

USB – Universal Serial Bus

Due to the widespread use of the Arduino platform and the constant growth of the maker community, adapters are available that convert from USB to USART with a voltage level of 5V or 3.3V. (FTDI chip)

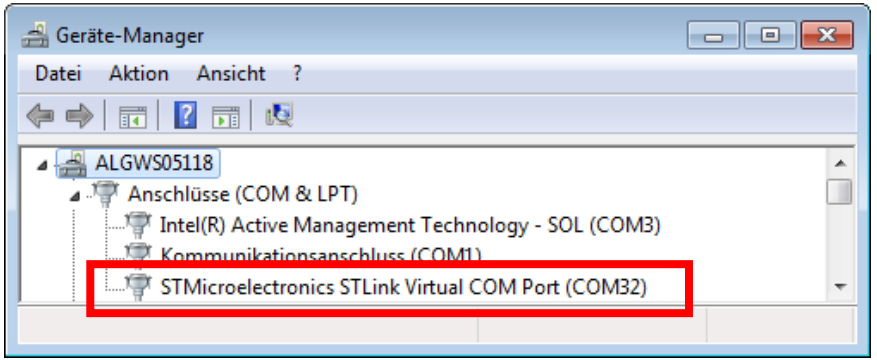


http://www.netzmafia.de/skripten/hardware/PC-Schnittstellen/usb.html
https://en.wikipedia.org/wiki/USB

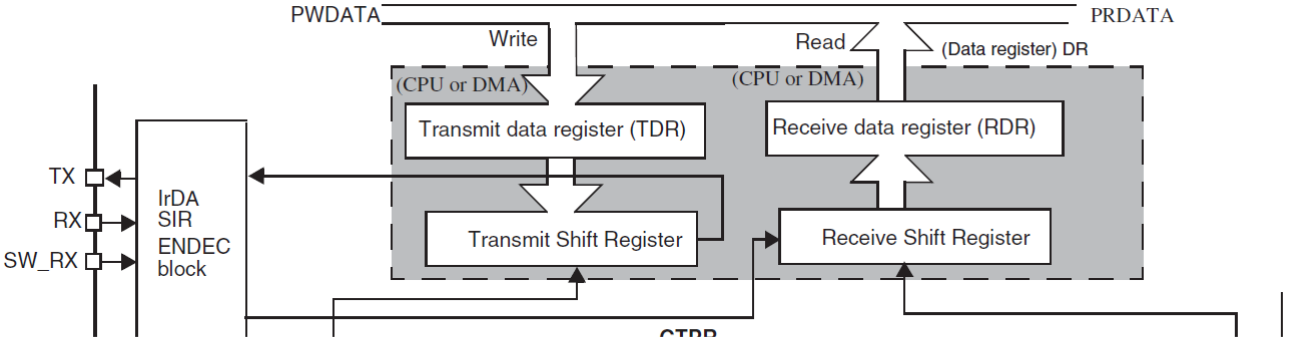
USART – STM32 - STLink

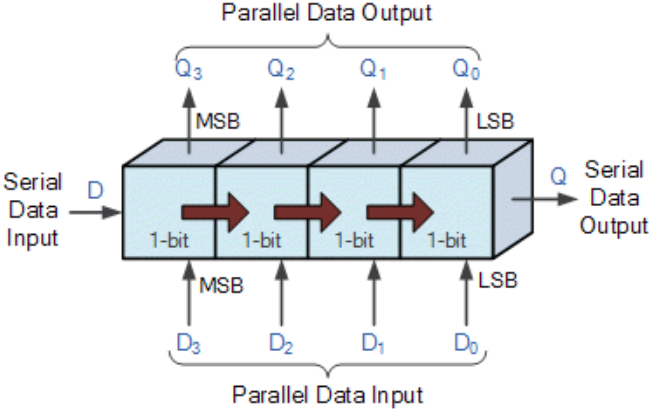
STM32 with Software (Firmware):

-> Virtual Com Port (USART to USB Tunnel)
CP210X and others

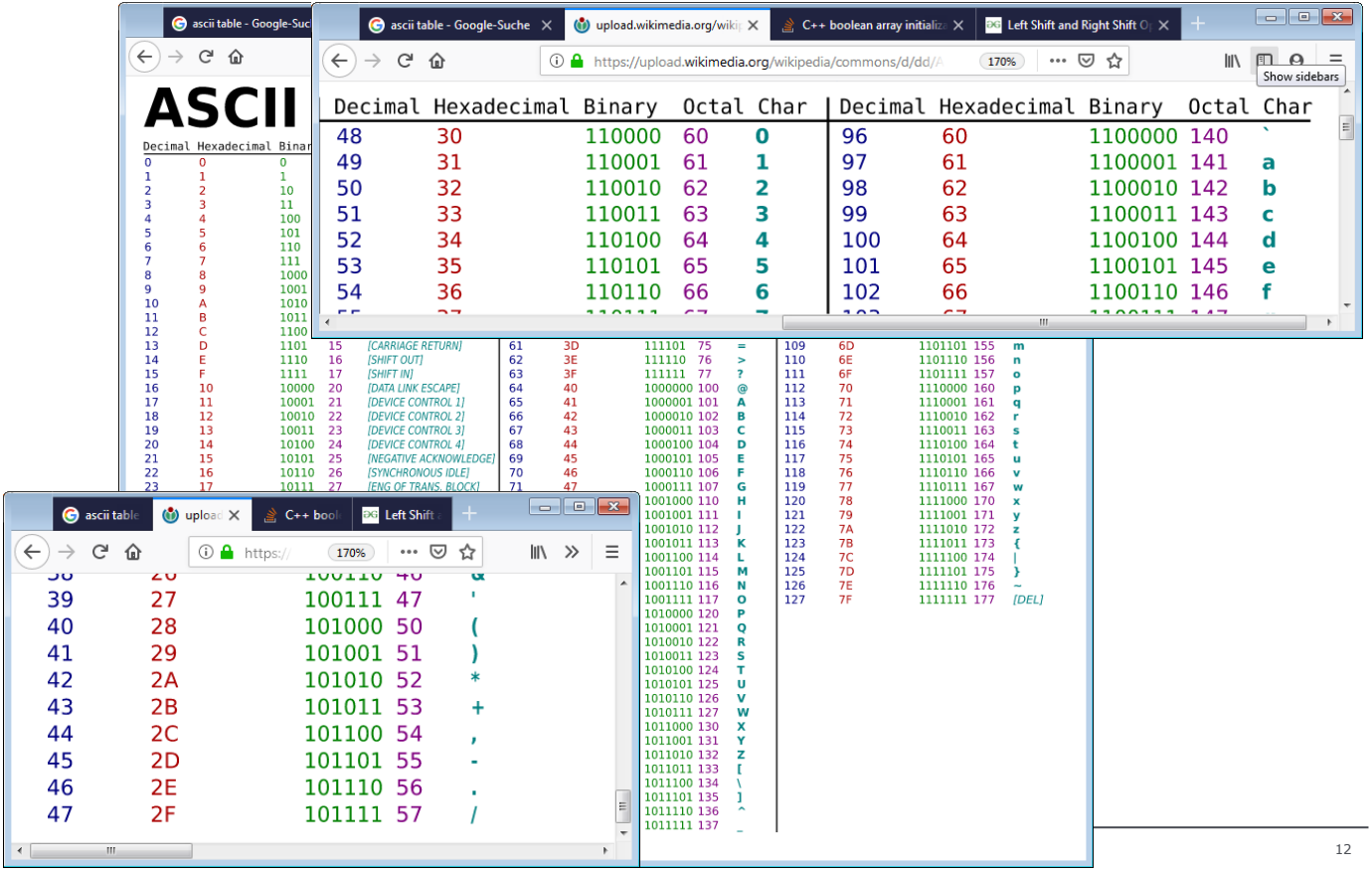


USART – STM32 – Shift Register





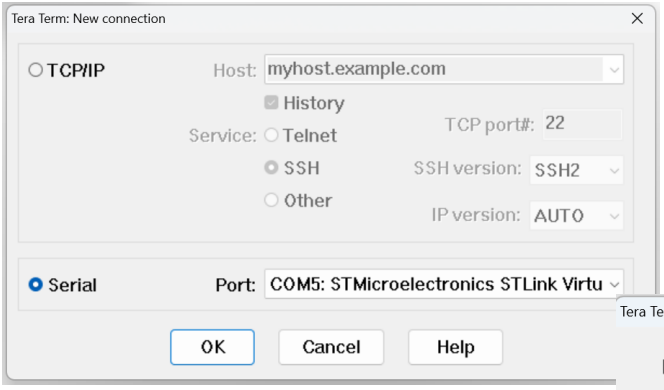
USART – ASCII

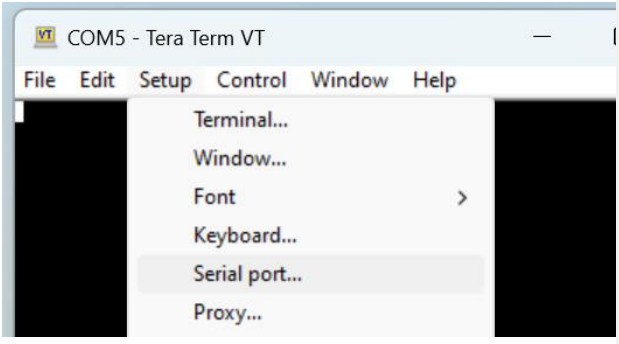


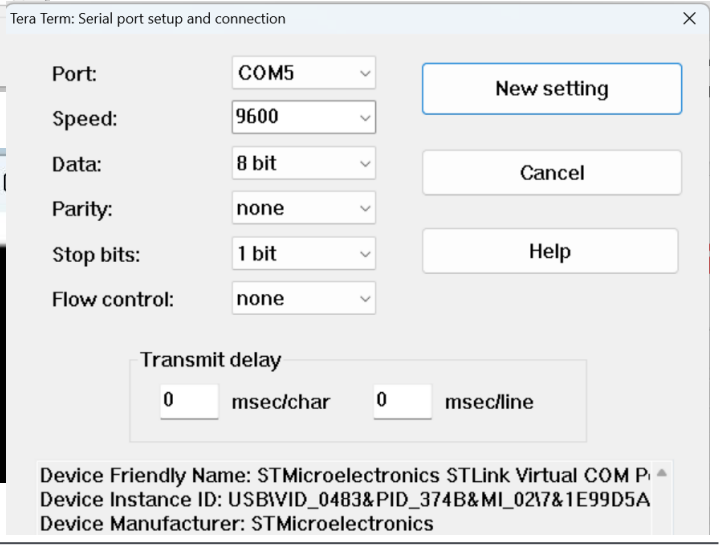
Decimal	Hexadecimal	Binary	Octal	Char
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6

Decimal	Hexadecimal	Binary	Octal	Char
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f

PC Terminal Program - TeraTerm

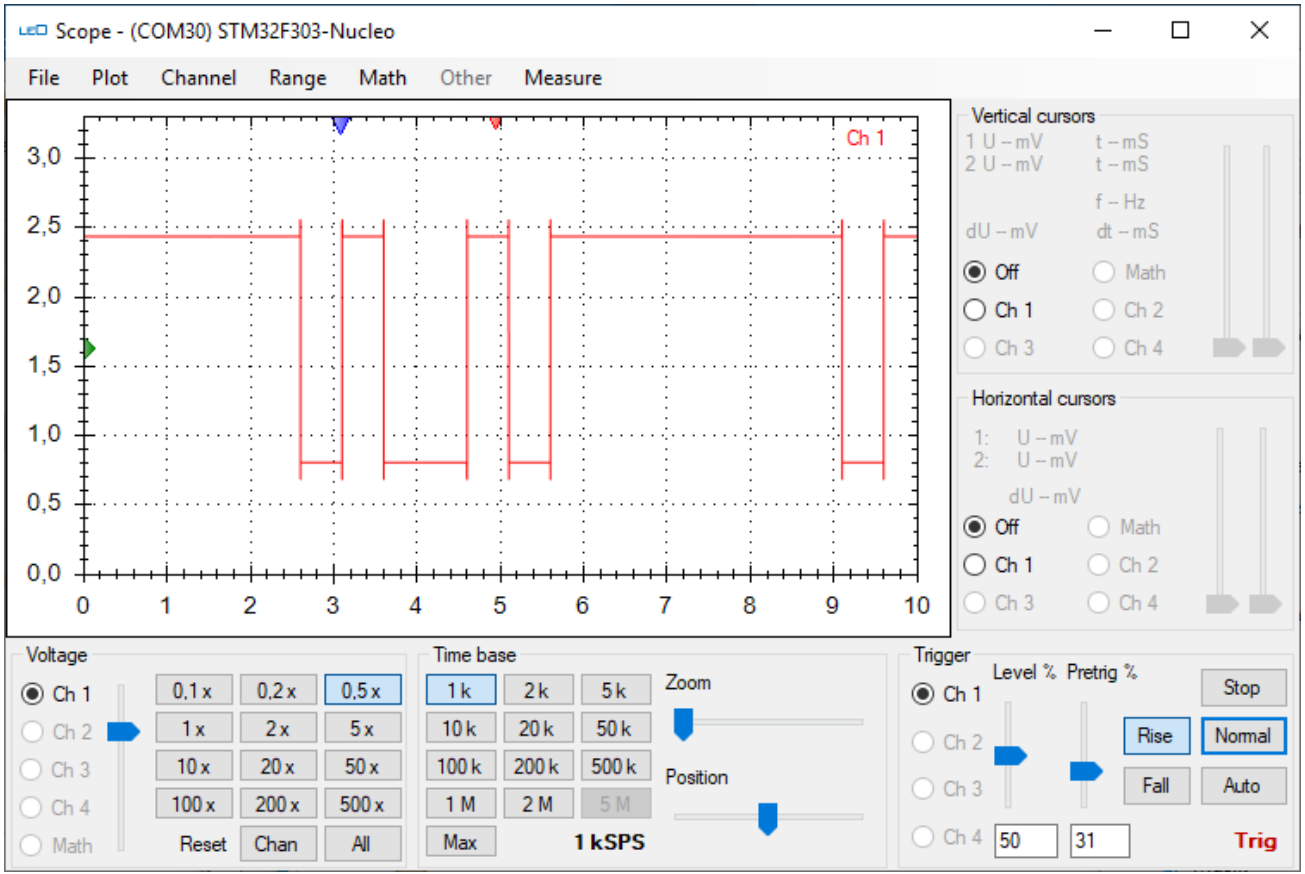






FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Data length 10k samples – 10 sek



https://randomnerdtutorials.com/esp32-digital-inputs-outputs-arduino/

RANDOM NERD TUTORIALS

Free eBooks About Contact Courses Login Get Courses

HOME ESP32 ESP8266 ESP32-CAM MICROPYTHON ARDUINO REVIEWS PROJECTS

Learn ESP32

ESP32 Introduction

ESP32 Arduino IDE

ESP32 Arduino IDE 2.0

VS Code and PlatformIO

ESP32 Pinout

ESP32 Inputs Outputs

ESP32 PWM

ESP32 Analog Inputs

ESP32 Interrupts Timers

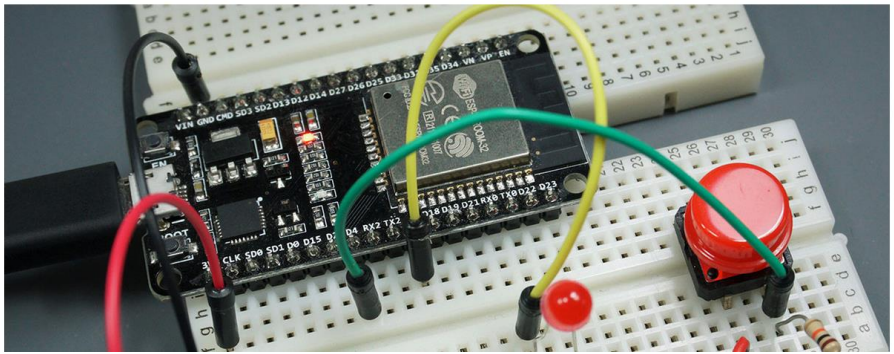
ESP32 Deep Sleep

Protocols

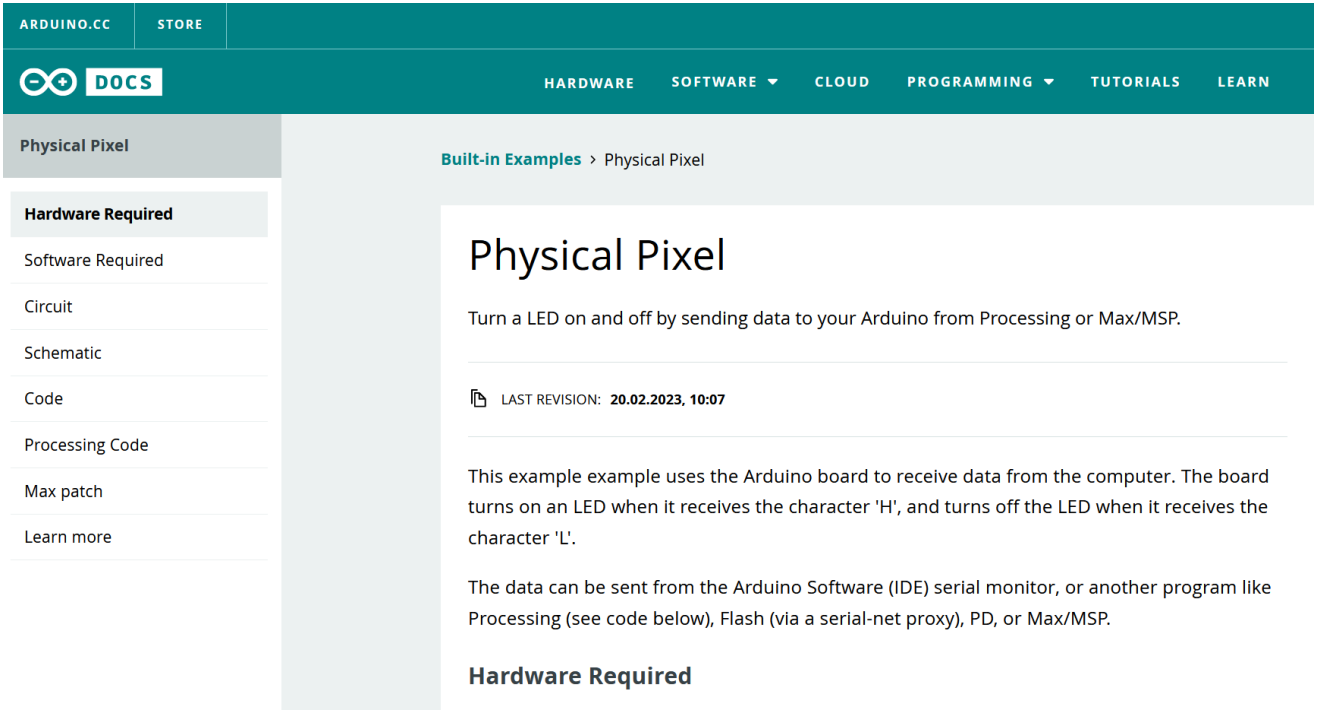
ESP32 Web Server

ESP32 Digital Inputs and Digital Outputs (Arduino IDE)

In this getting started guide you'll learn how to read digital inputs like a button switch and control digital outputs like an LED using the ESP32 with Arduino IDE.



https://docs.arduino.cc/built-in-examples/communication/PhysicalPixel



```
int incomingByte = 0; // for incoming serial data
const int ledPin = 33;
void setup() {
Serial.begin(115200);
pinMode(ledPin, OUTPUT);
}

void loop() {
//Serial.println("Hello");
//delay(1000);
 if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    incomingByte = Serial.read();
// if it's a capital H (ASCII 72), turn on the LED:
    if (incomingByte == 'H') {
     digitalWrite(ledPin, HIGH);
    }
    // if it's an L (ASCII 76) turn off the LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
 }
}
```

sketch_feb28a.ino

```
int incomingByte = 0; // for incoming serial data
const int ledPin = 33;
void setup() {
Serial.begin(115200);
pinMode(ledPin, OUTPUT);
}

void loop() {
//Serial.println("Hello");
//delay(1000);
 if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    incomingByte = Serial.read();
// if it's a capital H (ASCII 72), turn on the LED:
    if (incomingByte == 'H') {
     digitalWrite(ledPin, HIGH);
    }
    // if it's an L (ASCII 76) turn off the LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
 }
}
```

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

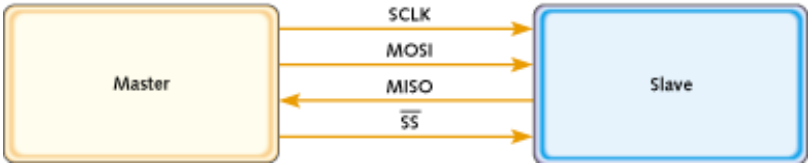
Anhang

>SPI

>I2C

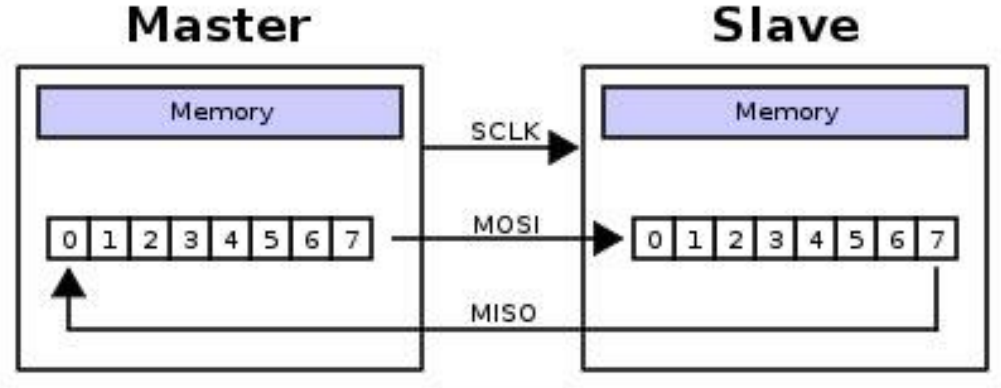
>PS/2 Keyboard

SPI Bus



- Synchronous serial data link operating at full duplex
- Master/slave relationship
- 2 data signals:
  - MOSI – master data output, slave data input
  - MISO – master data input, slave data output
- 2 control signals:
  - SCLK – clock
  - SS – slave select (no addressing)

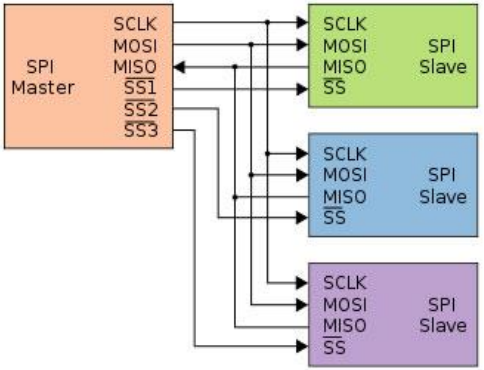
SPI uses a "shift register" model of communications



Master shifts out data to Slave, and shifts in data from Slave

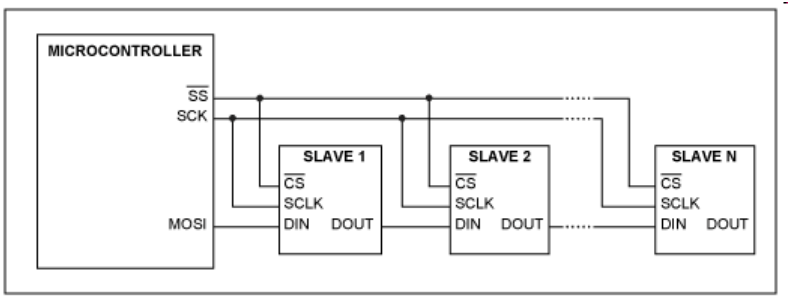
http://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/SPI_8-bit_circular_transfer.svg/400px-SPI_8-bit_circular_transfer.svg.png

Two bus configuration models



Master and multiple independent slaves

http://upload.wikimedia.org/wikipedia/commons/thumb/f/fc/SPI_three_slaves.svg/350px-SPI_three_slaves.svg.png

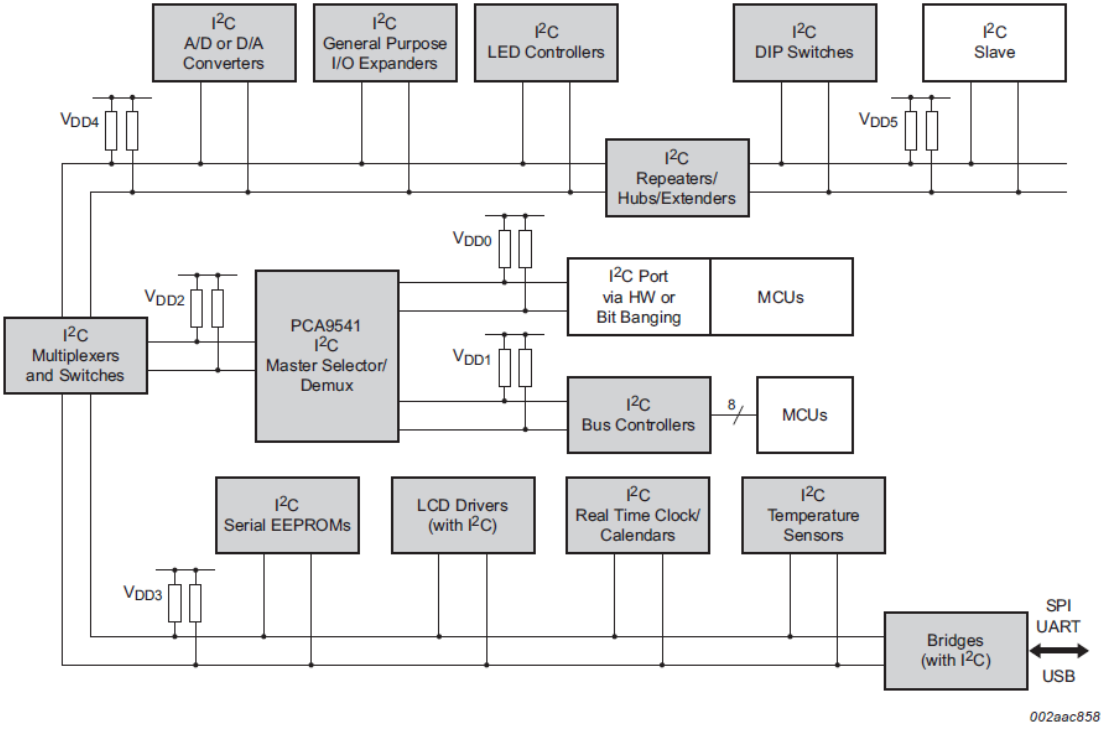


Some wires have been renamed

Master and multiple daisy-chained slaves

http://www.maxim-ic.com/appnotes.cfm/an_pk/3947

I2C / NXP UM10204



I2C NXP UM10204

Data validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see Figure 4). One clock pulse is generated for each data bit transferred.

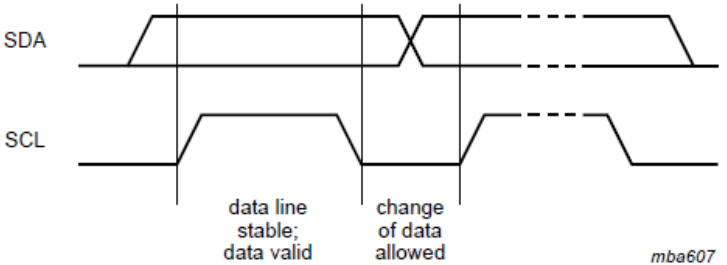
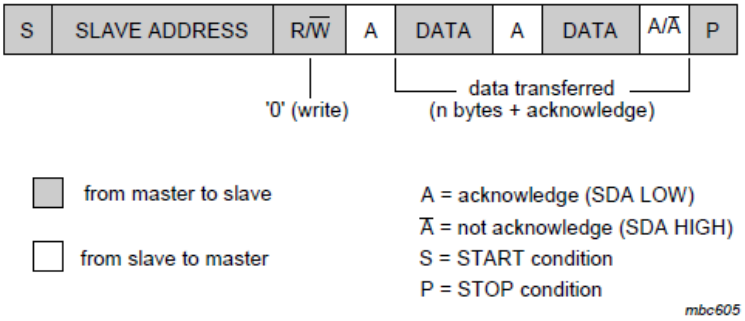
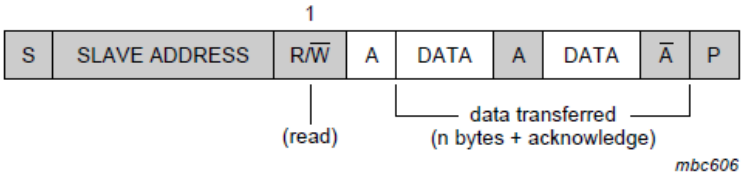


Fig 4. Bit transfer on the I²C-bus

I2C NXP UM10204



A master-transmitter addressing a slave receiver with a 7-bit address (the transfer direction is not changed)

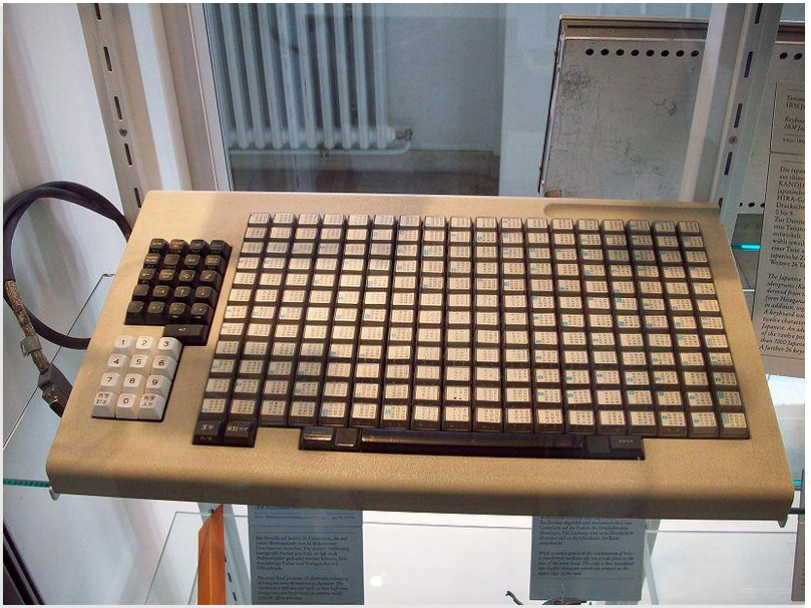


A master reads a slave immediately after the first byte

Microcontroller

PS/2 – Keyboard

Thomas Fischer



http://www.marjorie.de/ps2/start.htm
http://www.computer-engineering.org/
http://www.schatenseite.de/mamecontrol.html
http://de.wikipedia.org/wiki/Tastatur

PS/2 Keyboard

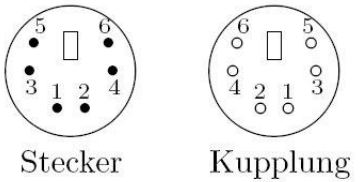
> If every key would be connected to one pin you would need a controller with 100pins. Within an infinite loop you could poll every pin. -> not the best solution!

> Better solution is to use the keys as connectors between rows and columns (matrix), 10 each. If a key is pressed down there will be a connection between one row and one column. Within an infinite loop you set one row to zero and ask all columns if there level is forced to zero. Now you need only 20 pins!

> A microcontroller (XT-keyboards an 8042) is sending this information to the PC using a Scancode.

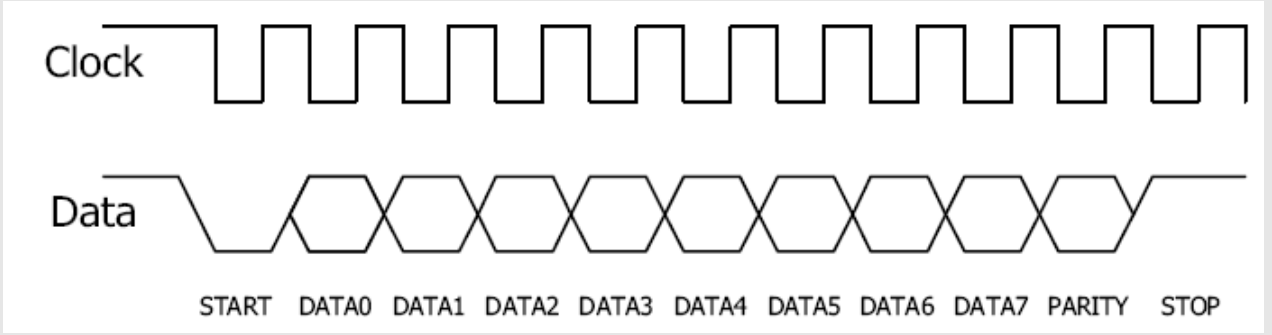
PS/2 Keyboard

- > Clock is zero when data is valid
- > Data line – transmit data bit by bit (serial transmission)



6-pin Mini-DIN (PS/2):

1 - Data
2 - nicht belegt
3 - Ground
4 - V_{CC} (+5 V)
5 - Clock
6 - nicht belegt



Serial Communication