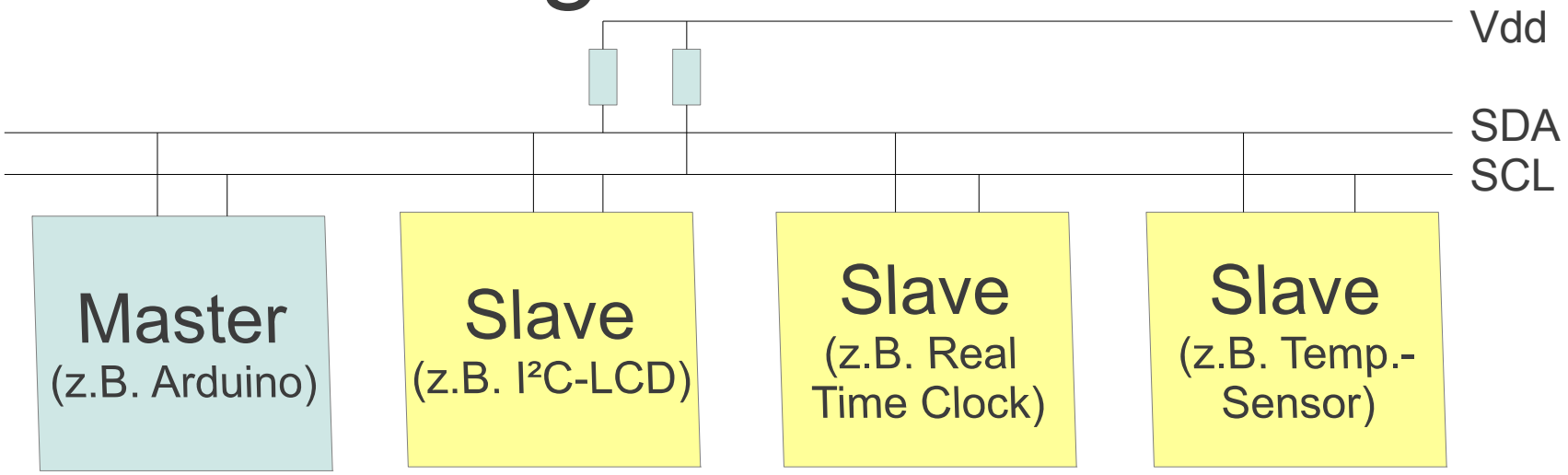
Der I²C-Bus

- Vorstellung des „Inter-Integrated Circuit“-Bus
- Aufbau und Funktionsweise
- Beispiel PortExpander am Arduino
- Weitere Anwendungsmöglichkeiten

Was ist der I²C-Bus

- entwickelt von Philips Anfang der 80er Jahre zur Verwendung in TV-Geräten
- Seit 01.10.2006 lizenzfrei da Patent von Philips abgelaufen
- Leicht zu realisieren, da nur zwei Signalleitungen
- theoretisch bis zu 112 Slaves (bei 7Bit-Adressierung) möglich
- mehrere Master möglich
- Open-Collector-Signale
- Hot-Plug-fähig
- Störanfällig bei größeren Entfernungen, daher meist nur innerhalb eines Gerätes verwendbar.

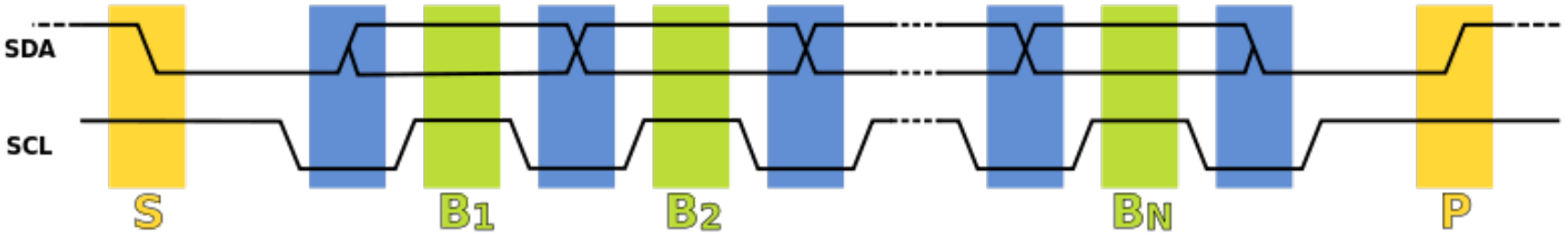
Leitungen des I²C-Bus



Beide Leitungen müssen mit einem Pullup-Widerstand (10k) auf Vdd angehoben werden.

I²C-Ports beim Arduino	SDA	SCL
Arduino < Uno	A4	A5
Arduino Leonardo	2	3
Arduino Mega, Due	20	21

Die Signale des I²C-Bus



Quelle: de.wikipedia.org/wiki/I2c

Start (S): SCL = HIGH , SDA = HIGH→LOW

Bit-Übertragung (B1-Bn):

1. SCL = LOW , SDA = HIGH/LOW
2. SCL = HIGH , (Takt abwarten)
3. SCL = LOW

Stop (P): SCL = HIGH , SDA = LOW→HIGH

HACKERSPACE BREMEN E.V.
make | hack | modify

Taktraten des I²C-Bus

Mode	**Max. Taktrate**
Standard	100 kHz
Fast Mode	400 kHz
Fast Mode Plus	1 MHz
High Speed Mode	3.4 MHz

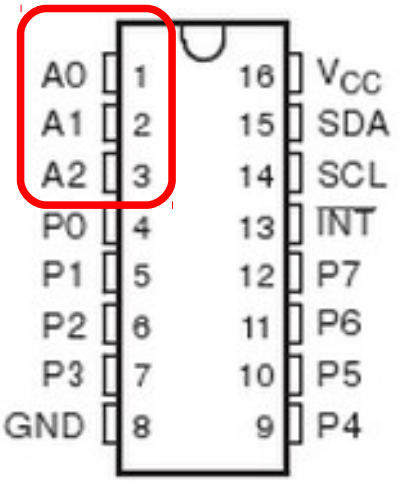
1 MHz und 3.4 MHz sind eher selten, 3.4MHz ist aufwendiger zu implementieren.

Langsamere Slaves verlängern die LOW-Flanke von SCL, um die Geschwindigkeit des Masters anzupassen (Clock-Stretching).

Adressierung im I²C-Bus

- 7 Bit – Adressierung + 1 R/W-Bit
  von 128 Adressen sind jedoch 16 reserviert:
  0000xxx = 0x00 – 0x07 und 1111xxx = 0x78 – 0x7F
- 10 Bit – Adressierung ist im gleichen System möglich
- Die 4 oberen Bits werden vom Hersteller vorgegeben
- Die 3 unteren Bits kann man selbst wählen.
  Dadurch können bis zu 8 baugleiche Slaves
  im gleichen Bus angesprochen werden.

PortExpander
PCF8574



HACKERSPACE
BREMEN E.V.
make | hack | modify

Kommunikation im I²C-Bus

START	A6	A5	A4	A3	A2	A1	A0	RW	ACK	D7	D6	D5	D4	D3	D2	D1	D0	ACK	...	STOP
	Adresse + R/W-Bit									Datenbits o. Registeradresse									...	
	Wird vom Master immer am Anfang gesendet									Kann mehrfach von Master oder Slave wiederholt werden										

Start: SDA = HIGH→LOW-Flanke bei SCL = HIGH

Gesendet wird immer „Most Significant Bit First“

R/W: HIGH = read , LOW = write

ACK = Der Empfänger setzt SDA auf LOW

Stop: SDA = LOW→HIGH-Flanke bei SCL = HIGH

HACKERSPACE BREMEN E.V.
make | hack | modify

Kommunikation im I²C-Bus

Beispiel:
Am Portexpander PCF8574 sollen alle Ports auf HIGH gesetzt werden.
Laut Datenblatt hat dieser die Adresse 0x20 (wenn A0-A2=LOW)

Die Ansteuerung sieht nun wie folgt aus:

Start	Adresse (A6-A0)	R/W	ACK	Daten (D7-D0)	ACK	STOP
1 → 0	010 0000	0	0	1111 1111	0	0 → 1
Master	Master	M	Slave	Master	Slave	Master

Kommunikation im I²C-Bus

Beispiel:
Am Portexpander PCF8574 sollen alle Ports gelesen werden.

Die Ansteuerung sieht nun wie folgt aus:

Start	Adresse (A6-A0)	R/W	ACK	Daten (D7-D0)	ACK	STOP
1 → 0	010 0000	1	0	XXXX XXXX		0 → 1
Master	Master	M	Slave	Slave		Master

Da der Master seine gewünschten Daten erhalten hat, kann das ACK entfallen.
Möchte man jedoch kontinuierlich die Daten lesen, kann der Master mit ACK signalisieren, dass weitere Daten gesendet werden sollen.

Kommunikations im I²C-Bus

Register eines Slaves beschreiben (z.B. EEPROM, RTC)

→ Start (SDA = HIGH->LOW)

→ **HW-Adresse senden** → **R/W=0** → ACK vom Slave

→ (MSB-)**Registeradresse senden** → ACK vom Slave
→ ggf. LSB-Registeradresse senden → ACK vom Slave

→ **Daten schreiben** → ACK vom Slave
→ ggf. mehrere Male wiederholen

→ Stop (SDA = LOW->HIGH)

Kommunikations im I²C-Bus

Register eines Slaves lesen (z.B. EEPROM, RTC)

→ Start (SDA = HIGH->LOW)

→ **HW-Adresse senden** → **R/W=0** → ACK vom Slave
→ **Registeradresse senden** → ACK vom Slave (evtl. MSB+LSB)
→ **Stop** (SDA = LOW->HIGH) oder „Repeated Start“

→ wieder **Start** (SDA = HIGH->LOW)
→ **HW-Adresse senden** → **R/W=1** → ACK vom Slave
→ **Byte lesen** welches der Slave vom aktuellen Register sendet
→ mit ACK vom Master ggf. folgende Register auslesen

→ Stop (SDA = LOW->HIGH)

HACKERSPACE BREMEN E.V.
make | hack | modify

Kommunikation im I²C-Bus

Beispiel Lesen der RealTimeClock DS1307:
HW-Adresse ist 0x68
Register: 0x00=sek; 0x01=min; 0x02=std; usw. ;)

→ START → Adresse 0x68 → RW = 0 → ACK abwarten
→ Registeradresse 0x00 senden → ACK abwarten → STOP

→ Start → Adresse 0x68 → RW = 1 → ACK abwarten
→ Sekunden empfangen → ACK senden
→ Minuten empfangen → ACK senden
→ Stunden empfangen → STOP

(Sekunden, Minuten und Stunden müssen noch in sinnvolle Werte Umgerechnet werden)

HACKERSPACE BREMEN E.V.
make | hack | modify

Kommunikation im I²C-Bus

10bit-Adressierung im 7bit-Bus

Der 10bit Adresse wird ein dafür reserviertes „11110“ vorangestellt.

START → HW-Adresse „11110“ + A9-A8 → RW-Bit
→ ACK(s) abwarten
→ HW-Adresse A7-A0 senden → ACK abwarten
→ ab hier alles so wie immer ;)

Es lassen sich dadurch alle neueren Typen mit 10bit-Adressen im alten 7bit-Adressbereich parallel mit alten Typen benutzen - auch mit Atmels „TWI“-Geräten, auch wenn diese selbst es nicht sprechen können.

Zusammenfassung 1.Teil

- Erst Adresse + R/W-Bit, dann Daten
- Zum Lesen von Registern erst die Registeradresse als Datenbyte senden, dann aus der HW-Adresse lesen.
- Alles nicht schwierig, aber ohne Datenblatt geht es auch nicht. (Adresse, Register, Befehle usw.)

Nach der Pause geht es praktischer weiter ;)

I²C am Arduino

I²C-Ports beim Arduino	SDA	SCL
Arduino < Uno	A4	A5
Arduino Leonardo	2	3
Arduino Mega, Due	20	21

Pullup-Widerstände nicht vergessen ;)

HACKERSPACE BREMEN E.V.
make | hack | modify

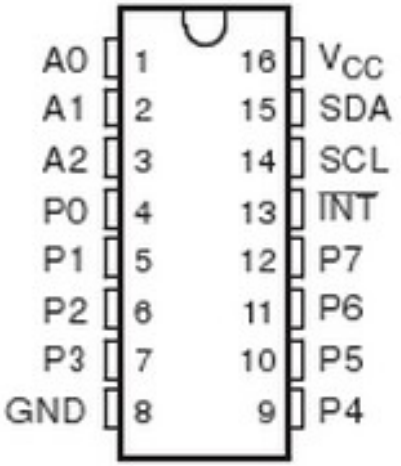
Die Wire-Library

Schreiben:

- Wire.begin(eigeneAdresse)
  In setup() - Bei Weglassen der Adresse arbeitet der Arduino als Master
- Wire.beginTransmission(HW-Adresse)
  Start, öffnet HW-Adresse schreibend;
- Wire.write(Byte / String / Array, Länge)
  Schreibt Byte, String oder Array ; gibt Anzahl der Bytes zurück
- Wire.endTransmission()
  Stop

Die Wire-Library

Codebeispiel zum Schreiben an PCF8574




```
#include Wire.h

byte adresseExpander = 32; // = 0x20 = B0100000
byte daten = 0;

void setup(){
  Wire.begin();
}

void loop(){

  Wire.beginTransmission(adresseExpander);
  Wire.write(daten);   // ehem. Wire.send
  Wire.endTransmission();

  daten++;
  if (daten > 255){daten = 0};
  delay(200);
}
```

HACKERSPACE BREMEN E.V.
make | hack | modify

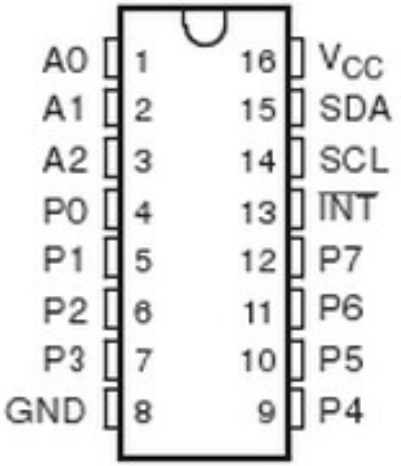
Die Wire-Library

Lesen:

- Wire.begin() in setup()
- Wire.requestFrom(HW-Adresse, Anzahl der Bytes)
  Start, öffnet HW-Adresse lesend;
  Anzahl der Bytes bestimmt Master-ACKs vor Stop.
- Wire.available()
  Gibt ein HIGH zurück, wenn Bytes empfangen wurden
- Wire.read()
  Gibt das empfange Byte zurück

Die Wire-Library

Codebeispiel zum Lesen des PCF8574




```
#include "Wire.h"
byte adresseExpander = 32; // = 0x20 = B010000
byte daten = 0;  // gelesenes Byte
byte expanderPort = 0; // zu lesendes Port-Bit
boolean portStatus = LOW;

void setup(){
  Wire.begin();
  Serial.begin(9600);
}

void loop(){
  Wire.requestFrom(adresseExpander, 1u);
  if (Wire.available()){
    daten = Wire.read();  //ehem. Wire.receive()
  }
  portStatus = bitRead(daten, expanderPort);
  Serial.println(portStatus ,BIN);
  delay(300);
}
```

HACKERSPACE BREMEN E.V.
make | hack | modify

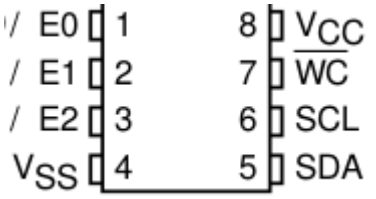
Die Wire-Library

Codebeispiel zum Schreiben eines EEPROMs 24Cxx

```
#include Wire.h
// E0, E1, E2, !WriteControl = GND
byte adresseEep = 80; // = 0x50 = B1010000
unsigned int startRegister = 10; // 16-Bit-Adr.
byte data = 123;

void setup(){
  Wire.begin();
  Serial.begin(9600);
}

void loop(){
 Wire.beginTransmission(adresseEep);
 Wire.write((int)(startRegister >> 8)); // MSB
 Wire.write((int)(startRegister & 0xFF)); // LSB
 Wire.write(data); // Daten schreiben
 Wire.write(data); // kann wiederholt werden
 Wire.endTransmission();
}
// von arduino.cc, Autor: hkhijhe, 01/10/2010
```



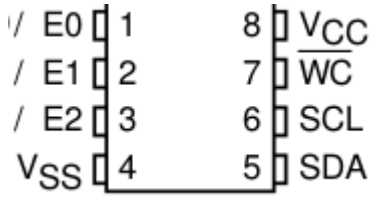
HACKERSPACE BREMEN E.V.
make | hack | modify

Die Wire-Library

Codebeispiel zum Lesen eines EEPROMs 24Cxx

```
#include „Wire.h“
byte adresseEep = 80; // = 0x50 = B1010000
unsigned int startRegister = 10; // 16-Bit-Adr.
byte data = 123;

void setup(){
  Wire.begin();
  Serial.begin(9600);
}
void loop(){
  Wire.beginTransmission(adresseEep);
  Wire.write((int)(startRegister >> 8)); // MSB
  Wire.write((int)(startRegister & 0xFF)); // LSB
  Wire.endTransmission(); //beendet Schreiben
  Wire.requestFrom(adresseEep, 1u); //1x Lesen
  if (Wire.available()) data = Wire.read();
  Serial.println(data, DEC);
  delay(1000);
}
```



HACKERSPACE BREMEN E.V.
make | hack | modify