Experiment 2: Digital-Analog-Wandler über I2C

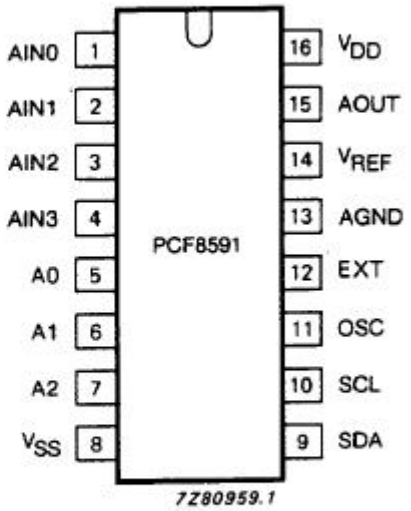
DK4AQ, 14.05.2013

V1.2

Preis derzeit: 2,45€
(Reichelt)

- 4 AD-Wandler + 1 DA-Wandler
- Auflösung: 8bit → 2^8 = 256 Stufen
- Ausgangsspannung: 0,01V – VDD-0,04V
- Betriebsspannung: 2,5 – 6V
- Ausgangsstrom: max.10mA

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I^2C-bus data input/output
SCL	10	I^2C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage



Anschlüsse des Bausteins

Slave-Adresse des Bausteins:

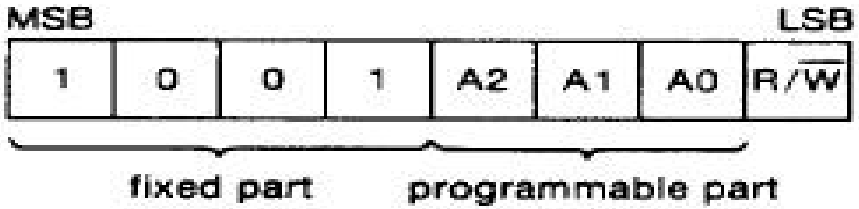
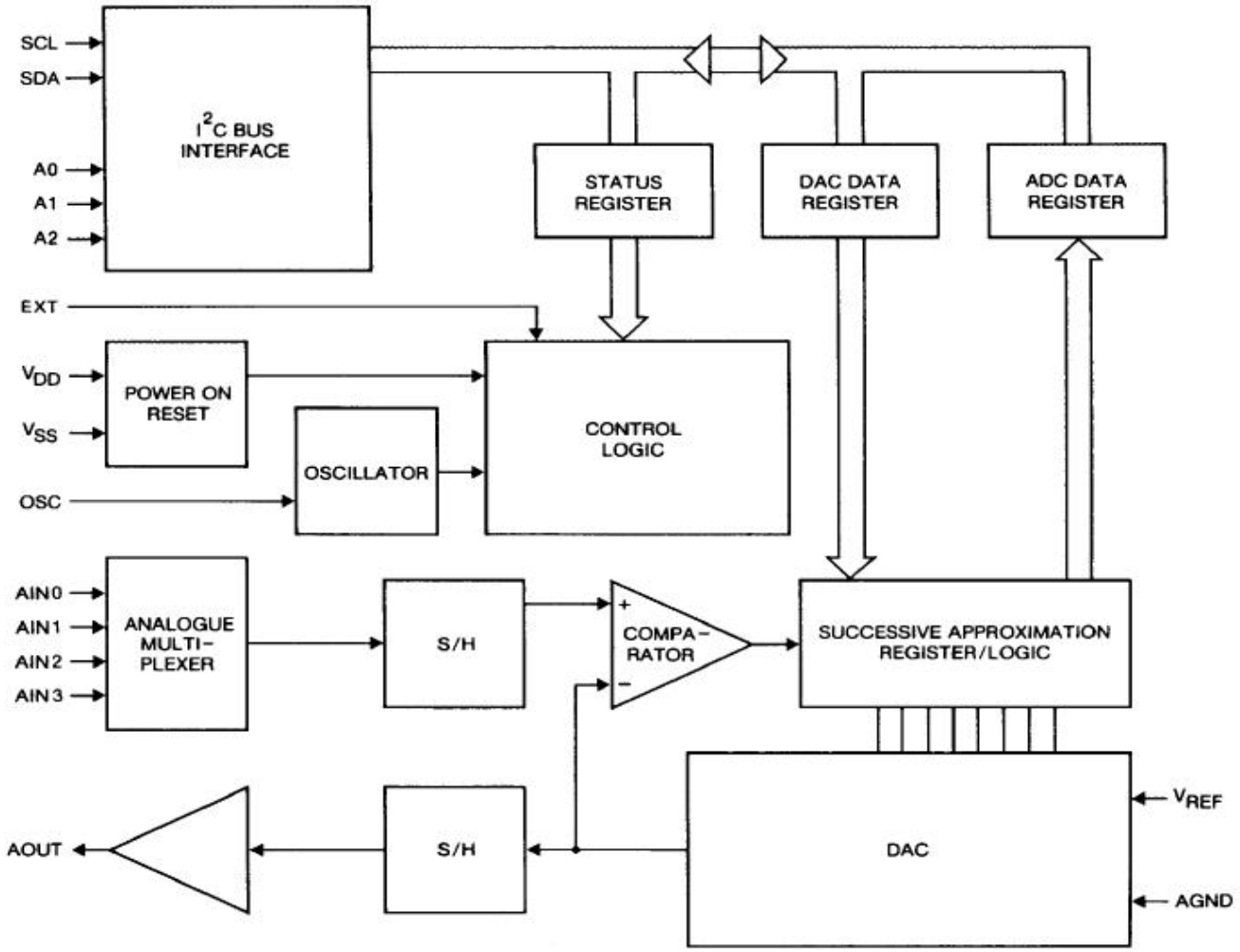


Fig.3 Address byte.

Die Grundadresse ist 1001000b = **48h**

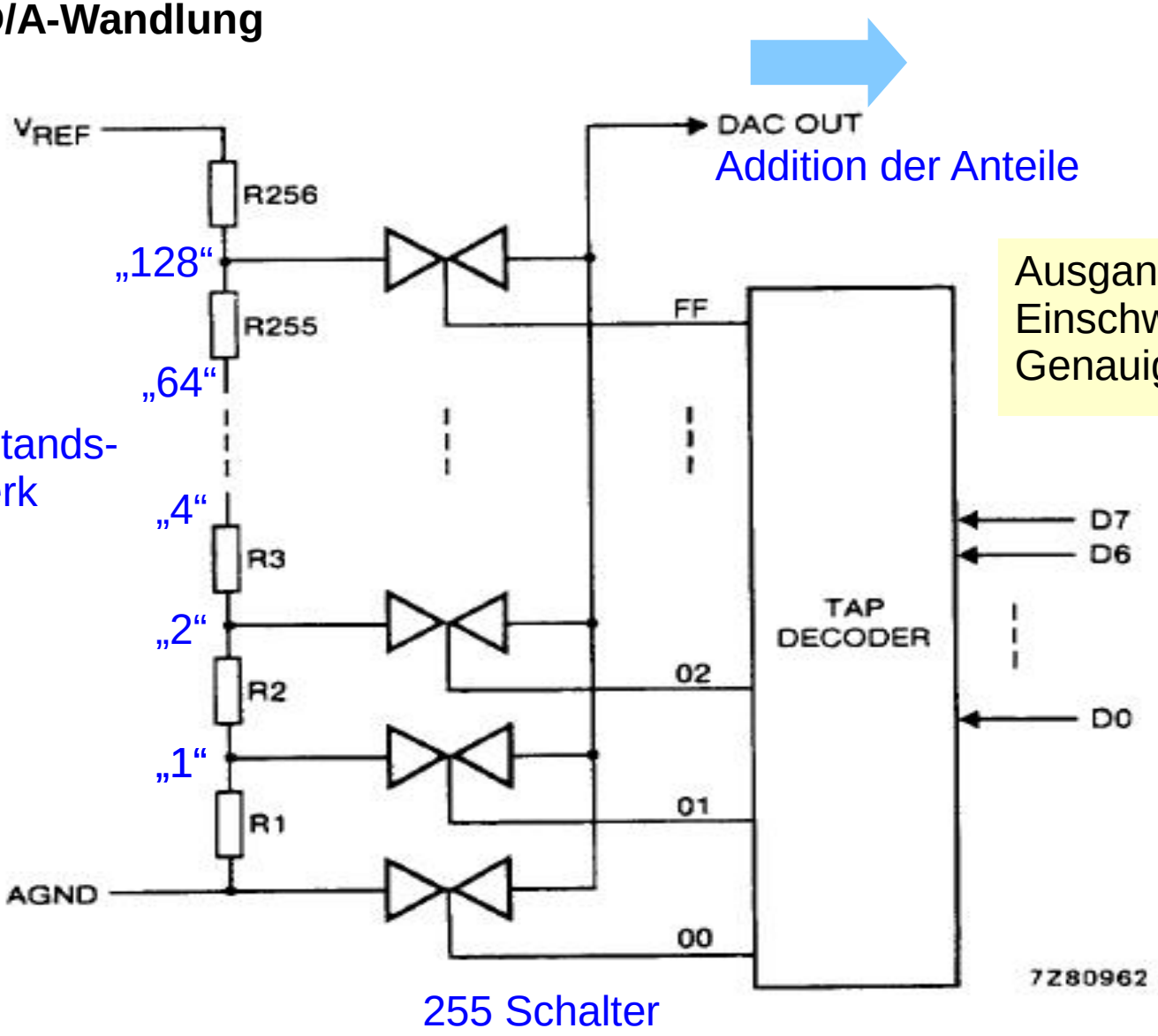
Die Bits A0-A3 sind Hardware-Anschlüsse, die von aussen am IC mit festen Pegeln belegt werden klönnen. Damit kann die Adresse von 48h bis 4Fh eingestellt werden.



Blockbild PCF8591

Der PCF8591 ist ein Baustein, der basierend auf einem System in dem mit Hilfe der „sukzessiven Aproximation“ Analog-/Digital-Wandler und Digital-/Analog-Wandler.

D/A-Wandlung



Addition der Anteile

Widerstands-
netzwerk

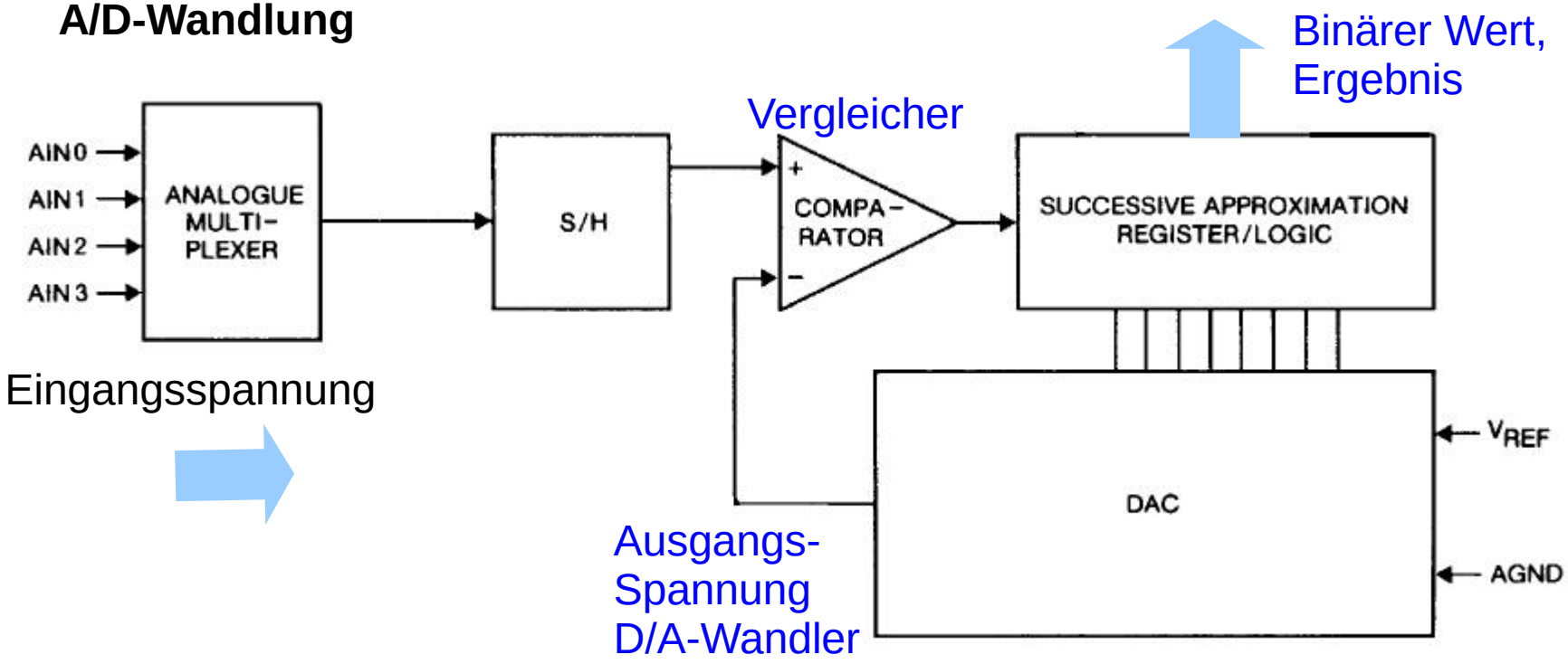
Ausgangsspannung: 0V-VREF,
Einschwingzeit < 90µs,
Genauigkeit: +/-1,5 LSB

8-Bit-Binärwert

255 Schalter

Der D/A-Wandler wird durch die Auswahl von Spannungsbeiträgen aus einem Widerstandsnetz erreicht. Jeder Beitrag entspricht seiner Gewichtung im binären Zahlensystem.

A/D-Wandlung



Der A/D-Wandler nutzt die Struktur des D/A-Wandlers. Die A/D-Wandlung wird durch Hochzählen des Registers erreicht. Mit dem Hochzählen steigt die Ausgangsspannung des D/A-Wandlers. Der Vergleicher stellt fest, wann die D/A-Spannung den Engangs-wert erreicht. Dann wird das Hochzählen gestoppt und der bis dahin gezählte Wert ist das Ergebnis der Wandlung.

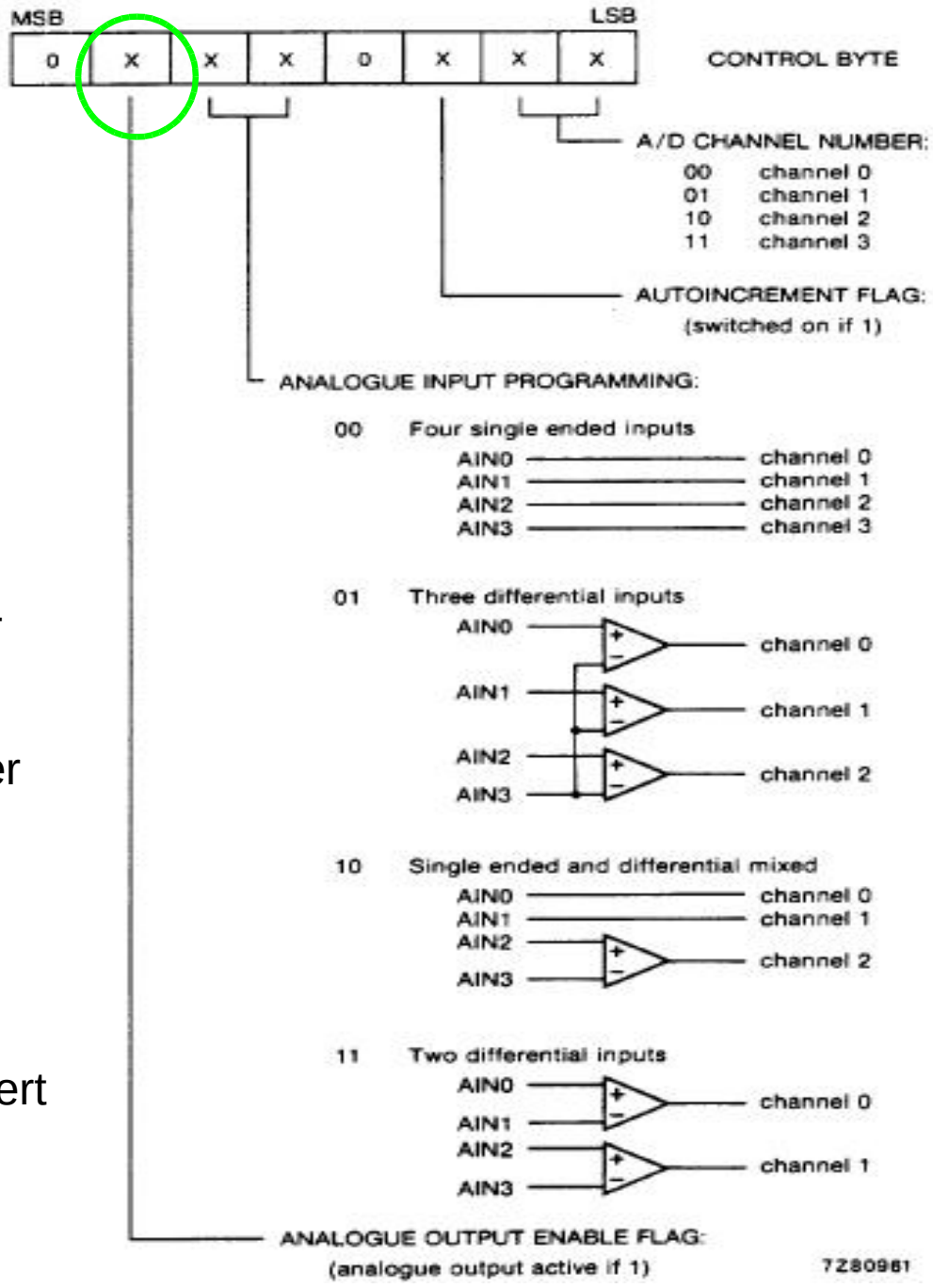
Messspannung: = 0V-VREF, Genauigkeit (Linearität): +/- 1,5 LSB, Wandlungszeit < 90µs

Struktur des Control-Bytes

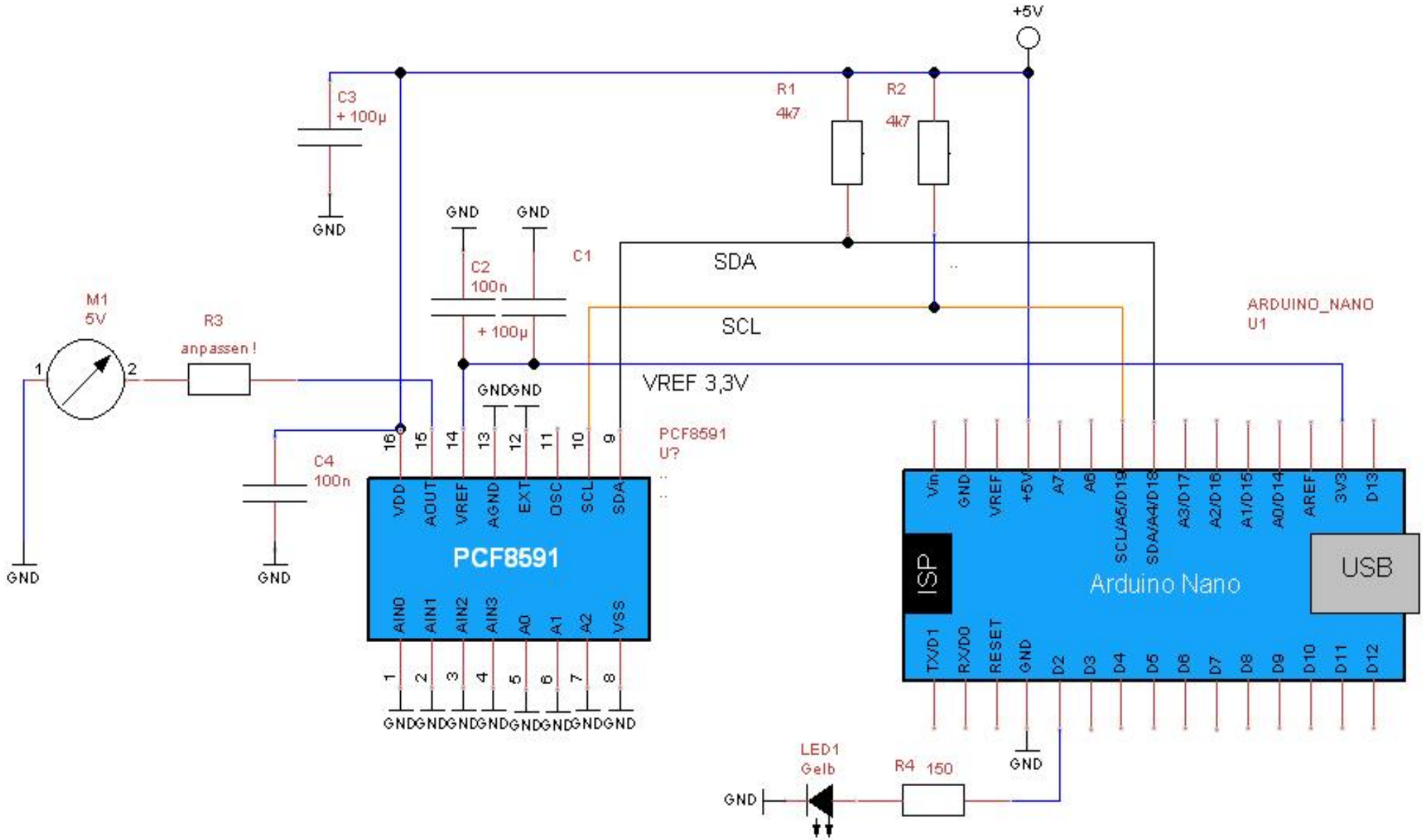
Der Baustein verfügt über ein Control-Register und ein Daten-Register (Lesen = A/D-Wert, Schreiben = D/A-Wert).

Im Control-Register kann man die Betriebsart einstellen.
Das Bit „Autoinkrement“ sorgt dafür, dass nach Lesen des eines Analog-Kanals automatiscvh der nächste Kanal eingelesen wird. So können über I2C alle KaNnäle hintereinander gelesen werden.

Für die D/A-Wandlung ist das Bit „Analog Output Enable“ wichtg. Es muß auf 1 gesetzt werden. Also bekommt das Control-Wwort den Wert 0100000b oder 40h.



Schaltung DA-Wandler steuern über I2C



Details zur Hardware-Beschaltung des PCF8591

Adressierungseingänge A0-A2: GND => log. 0, Adresse des Chips ist damit die Grundadresse 48h (s.o.)

Eingang EXT: GND , Es soll nicht ein externer Oszillator benutzt werden, sondern der interne Oszillator

VREF: Der Baustein hat keine eigene Referenzspannung zur Spannungsmessung. Daher holen wir die Referenzspannung vom Arduino (3,3V). Ergebnis: der DA-Wert von 0-255 bestreicht den Bereich von 0V bis 3,3V. Auch A/D-Messungen hätten den Eingangsbereich 0-3,3V

AIN0-AIN3: Analogeingänge, werden in diesem Experiment nicht benutzt, GND

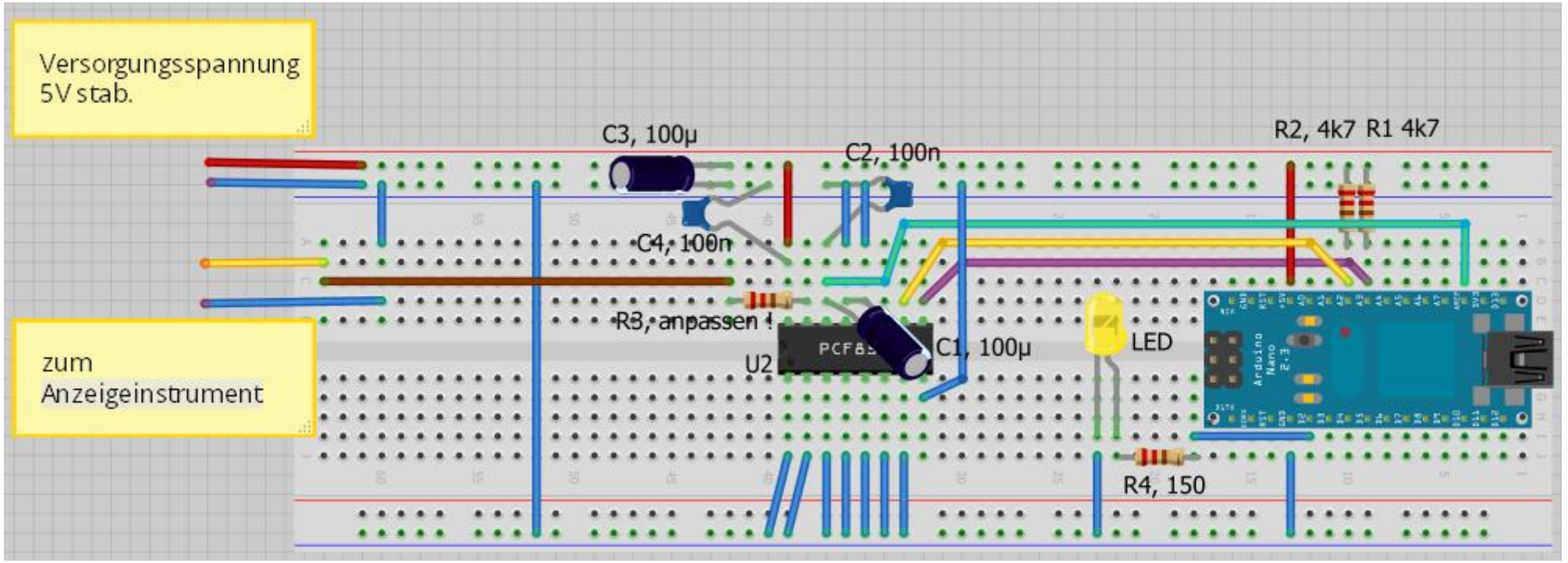
AGND: Masse-Referenzpunkt für alle Analog-Operationen, in diesem Experiment einfach auf GND

C3,C4: Abblockkondensatoren Betriebsspannung, möglichst dicht am Bausteinanschluss

C1,C2: Abblockkondensatoren Referenzspannung, möglichst dicht am Bausteinanschluss

Messinstrument: z.B. Vielfachmessgerät zur Anzeige der Ausgangsspannung

Aufbauvorschlag I2C_DA_Wandl_Exp2



Kurze Drähte oberhalb des Wandlers und Kondensator möglichst dicht am IC sind wichtig.
Der Widerstand R3 wird an das verwendete Messinstrument angepasst.

Programm I2C_DA_Wandl_Exp2 (1)

```
//I2C_DA_Wandl_Exp2
// mit PCF8591
#include <Wire.h>

#define DAC_ADR 0x48
#define ENABLE_OUTPUT B01000000
#define ledPin 2

byte stat = 0;
byte n = 0;

void setup()
{
    // Verwendete Bibliotheken
    Wire.begin();
    Serial.begin(9600);

    Serial.print("Start ");
    // Ausgang für LED deklarieren
    pinMode(ledPin, OUTPUT);
}
```

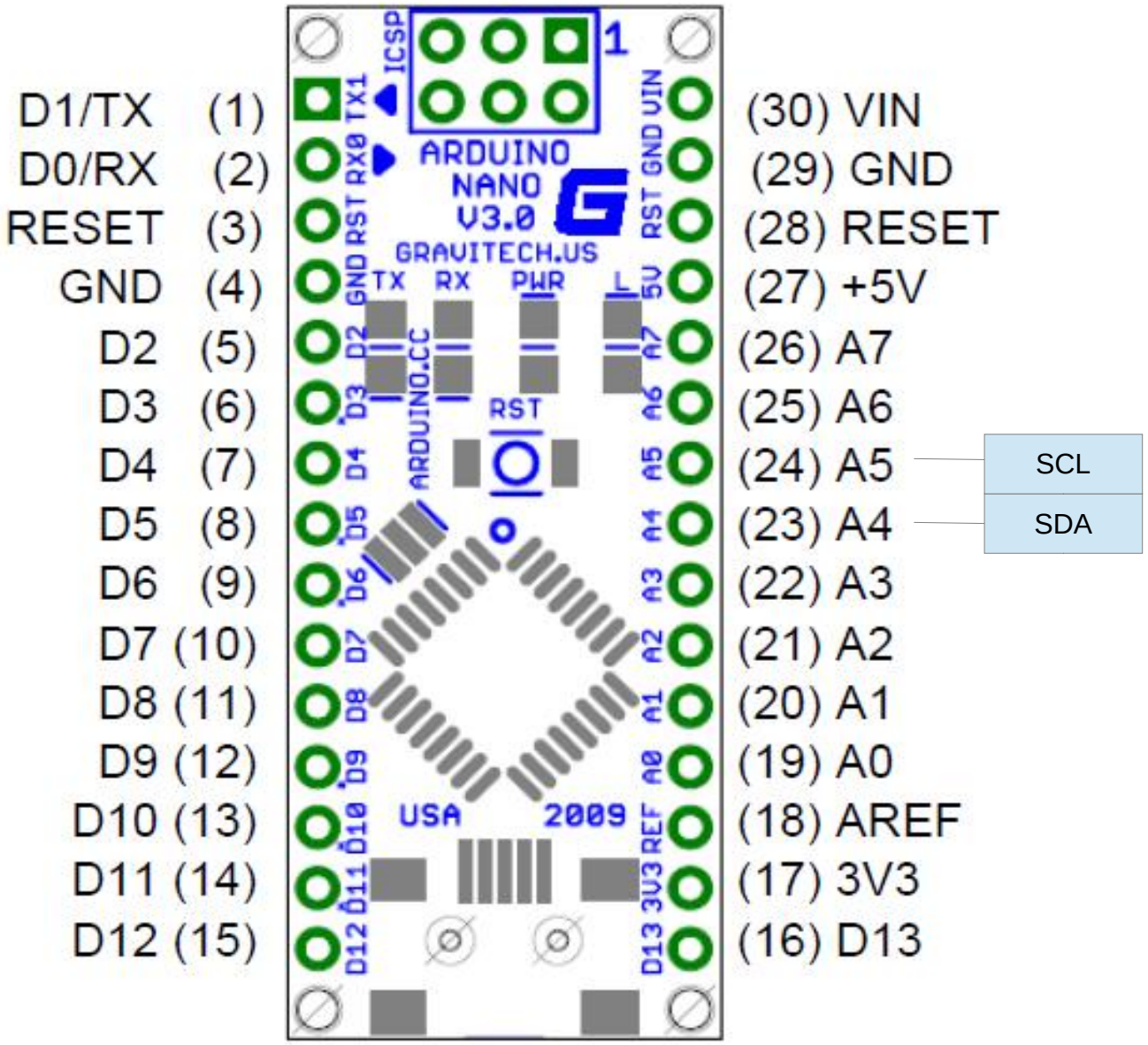
Programm I2C_DA_Wandl_Exp2 (2)

```
void loop()
{
    // Daten werden staendig erhoeht und ausgegeben (Rampe)
    // Nach Ueberlauf der Bbyte-Variablen startet die Rampe wieder bei null

    Wire.beginTransmission(DAC_ADR);        // Telegramm eröffnen zu Slave 48h
    Wire.write(ENABLE_OUTPUT);              // Control-Register in Puffer fuellen
    Wire.write(n);                          // Daten-Byte in Puffer füllen
    stat = stat | Wire.endTransmission();   // Telegramm absenden und
                                            // Fehlermeldungen kumulieren
    Serial.print(n);
    Serial.println(" ");

    n++;                // Datenwert erhöhen für nächste Ausgabe
    delay(10);          // Pause
```

Programm I2C_DA_Wandl_Exp2 (3)

```
    // Nach Hochlaufen der Daten bis 255 wird bei 0 wieder angefangehn
    if(n == 255)
    {
      //LED einschalten, wenn im letzten Durchlauf ein Fehler im Fehlermerker steht
      if(stat != 0)
      {
        digitalWrite(ledPin,1);
      }
      else
      {
        // Kein Fehler, dann LED loeschen
        digitalWrite(ledPin,0);
      }
      // Fehlermerker loeschen
      stat = 0;
      n = 0;
      delay(1000);
    }
}
```

Anhang


D1/TX (1)
D0/RX (2)
RESET (3)
GND (4)
D2 (5)
D3 (6)
D4 (7)
D5 (8)
D6 (9)
D7 (10)
D8 (11)
D9 (12)
D10 (13)
D11 (14)
D12 (15)
(30) VIN
(29) GND
(28) RESET
(27) +5V
(26) A7
(25) A6
(24) A5
(23) A4
(22) A3
(21) A2
(20) A1
(19) A0
(18) AREF
(17) 3V3
(16) D13
SCL
SDA
ARDUINO
NANO
V3.0
GRAVITECH.US
ARDUINO.CC
USA
2009