**Formelsammlung**

**LOKAL**

**Arduino-IDE**

1.5.2 TG Informationstechnik

Version: V 4.28

Gültig ab Abitur 2024

**Inhaltsverzeichnis:**

[1 Hardware - Mikrocontrollertechnik 3](#_Toc107818242)

[1.1 Blockschaltbild „Prüfungscontroller“ 3](#_Toc107818243)

[1.2 Hochsprache C/CPP 4](#_Toc107818244)

[Datentypen 4](#_Toc107818245)

[Zeiger und Referenzen 4](#_Toc107818246)

[Operatoren 4](#_Toc107818247)

[Schleifen 6](#_Toc107818248)

[Programmverzweigungen 7](#_Toc107818249)

[Beispiel eines C/CPP-Programms 10](#_Toc107818250)

[1.3 On Chip Peripherie 11](#_Toc107818251)

[Portpin: Eingabe und Ausgabe 11](#_Toc107818252)

[Externer Interrupt 11](#_Toc107818253)

[Timer 12](#_Toc107818254)

[Puls-Weiten-Modulation (PWM) 13](#_Toc107818255)

[Analog – Digital – Wandlung 14](#_Toc107818256)

[Digital – Analog – Wandlung 14](#_Toc107818257)

[1.4 Externe Kommunikationsmöglichkeiten 15](#_Toc107818258)

[Universal Asynchronous Receiver Transmitter (UART) 15](#_Toc107818259)

[Serial Peripheral Interface (SPI) 16](#_Toc107818260)

[Inter-Integrated Circuit (I2C) SCL (Serial Clock): Taktleitung SDA (Serial Data): Datenleitung 17](#_Toc107818261)

Weitere Hinweise <https://www.stm32duino.com>

# Hardware - Mikrocontrollertechnik

## Blockschaltbild „Prüfungscontroller“

DigitalIn

DigitalOut

PortIn

PortOut

InterruptIn

DigitalInOut

z.B. Taste

z.B. LED

z.B. Schalter

n

z.B. Motor

n

z.B. Lichtschranke

z.B. SenderEmpfaenger

PortInOut

z.B. Tastatur

n

AnalogIn

AnalogOut

z.B. Poti

z.B. Helligkeit

PWMOut

z.B. Servo

SPI

z.B. Display

BufferedSeriall

z.B. Bluetooth

MISO, MOSI,

SCK, SS

RX TX

I2Cl

z.B. EEPROM

Hardware

Timer

Hardware

Timer

Hardware

Ext. Int.

SCL SDA

Pin

Pin

Pin

Pin

Port

Port

Port

Hardware

USART

Pin

Pin

Pin

4 Pins

2 Pins

2 Pins

## Hochsprache C/CPP

### Datentypen

|  |  |  |  |
| --- | --- | --- | --- |
| **Datentyp** | **Bits** | **Vorzeichen** | **Wertebereich** |
| unsigned char | 8 | + | 0 .. 255 |
| (signed) char | 8 | - + | -128 ..127 |
| uint\_32t/uint16\_t | 32/16 | + | 0 .. 4294967295 bzw. 0 .. 65535 |
| int\_32t/int16\_t | 32/16 | - + | -2147483648 .. 2147483647 bzw. 32768 .. 32767 |
| float | 32 | - + | -3,4E38 .. 3,4E38 |

### Zeiger und Referenzen

int x=127; //Wert

int \*y; //Zeiger

\*y=x; //der Zeiger weist auf eine Variable mit dem Wert von x

y=&x; //der Zeiger bekommt die Adresse der Variable x im Speicher

Beispiel:

|  |  |  |
| --- | --- | --- |
|  | Adresse | RAM |
| x | 0x20000000 | 127 |
| y | 0x20000004 | 0x20000000 |
|  |  |  |

printf("%d %x %d\r\n",x,(int)y,\*y);

liefert folgende Ausgabe: 127 0x20000000 127

### Operatoren

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mathematische Operatoren** | | **Priorität** | **Vergleichs- und logische Operatoren** | |
| ++ | Inkrement | Höchste | ! | NOT |
| -- | Dekrement |  | > | Größer |
| - | Vorzeichen |  | >= | Größer gleich |
| \* | Multiplikation |  | < | Kleiner |
| / | Division |  | <= | Kleiner gleich |
| % | Modulo, Rest der Division |  | == | Gleich |
| + | Plus |  | != | Ungleich |
| - | Minus |  | && | AND |
|  |  | Niedrigste | || | OR |

Da ein Gleichheitszeichen in C ein Zuweisungsoperator ist, weist man z.B. mit x = 10; der Variablen x den Wert 10 zu.

|  |  |
| --- | --- |
| **Bitweise Operatoren** | |
| & | UND |
| | | ODER |
| ^ | EXOR |
| ~ | Einerkomplement |
| << | Nach links schieben |
| >> | Nach rechts schieben |

|  |  |
| --- | --- |
| **Kurzschreibweisen** | |
| += | x += 3; wie x = x + 3 |
| -= | x -= 3; wie x = x - 3 |
| \*= | x \*= 5; wie x = x \* 5 |
| /= | x /= 7; wie x = x / 7 |

|  |  |
| --- | --- |
| **Beispiele** | **Ergebnisse** |
| x = 10; |  |
| y = ++x; | y = 11 |
| y = x++; | y = 10 |
| y = 0x11; |  |
| y = y<<1; | y = 0x22 |
| (Bitweise um 1 nach links schieben) |  |

### Schleifen

#### FOR-Schleife (zählergesteuerte Schleife)

Schleife, mit einer genau berechenbaren Anzahl an Wiederholungen.

for (<zaehlvariable=startwert>;<bedingung>;<schrittweite>) {

…

}

* startwert Anfangswert der Zählvariablen
* bedingung Schleife wird so lange durchlaufen, wie die Bedingung wahr ist
* schrittweite Anweisung zum Erhöhen oder Erniedrigen der Zählvariablen

**Beispiel:**

// *DigitalOut ausgang(PC\_0)* 10x invertieren

for (int i=0; i<10; i++) {

ausgang = !ausgang;

}

#### WHILE-Schleife (kopfgesteuerte Schleife)

Schleife, die wiederholt wird, so lange die Bedingung am Schleifenanfang erfüllt ist.

while (<bedingung>) {

…

}

Solange die am Anfang stehende **Bedingung erfüllt ist,** wird die Schleife wiederholt. Die Prüfbedingung steht **vor den Anweisungen**, sie heißt deshalb **kopfgesteuerte Schleife.**

Wenn die am Schleifenanfang stehende **Bedingung nicht erfüllt ist,** dann wird die gesamte Schleife übersprungen.

**Beispiel:**

// Solange der Taster *DigitalIn taster(PA\_1)* gedrückt ist, wird der

// Ausgang *DigitalOut ausgang(PC\_0)* invertiert

while (taster==true) {

ausgang = !ausgang;  
}

#### Do-WHILE-Schleife (fußgesteuerte Schleife)

Schleife, die mindestens einmal durchlaufen wird, da erst am Ende der Schleife mit der Überprüfung der Bedingung entschieden wird, ob die Schleife wiederholt werden muss.

do {

…

} while (<bedingung>);

**Beispiel:**

// Die Schleife wird maximal 100 mal und minimal 1 mal durchlaufen. Sie wird früh-

// zeitig abgebrochen, wenn der Taster *DigitalIn taster(PA\_1)* gedrückt (=1) wird.

x = 100;

do {

x--;

} while ((x>0) && taster==0);

### Programmverzweigungen

#### Einfache Verzweigung mit if

Bei der if-Anweisung werden die Anweisungen innerhalb des if-Blocks nur dann ausgeführt, falls die Bedingung wahr ist.

if (<bedingung>) {

<anweisung1>;

<anweisung2>;

…

}

**Beispiel:**

// Wenn *taster1* gedrückt ist, soll *ausgang1* eins und *ausgang2* null werden.

// Drückt man dagegen *taster2*, wird nur *ausgang2* zu eins.

if (taster1==1) {

ausgang1 = 1; // Block mit mehreren Anweisungen wird ausgeführt,

ausgang2 = 0; // wenn die Bedingung hinter if wahr ist

}

if (taster2==1)

ausgang2 = 1; // Nur eine Anweisung, keine {} notwendig

#### Zweiseitige Verzweigung mit if

Bei der if/else-Anweisung kann zwischen **zwei Alternativen** entschieden werden. Ist die Bedingung wahr, so wird die erste Alternative (if-Block), ansonsten die zweite Alternative (else-Block) an Anweisungen ausgeführt.

if (<bedingung>) {

<anweisung1>;

<anweisung2>;

…

} else {

<anweisung3>;

<anweisung4>;

…

}

**Beispiel:**

// Wenn *taster1* gedrückt ist, soll *ausgang1* eins und *ausgang2* null werden,

// andernfalls soll *ausgang1* null und *ausgang2* eins werden.

if (taster1==1) {

ausgang1 = 1; // Block mit mehreren Anweisungen wird ausgeführt,

ausgang2 = 0; // wenn die Bedingung hinter if wahr ist

} else {

ausgang1 = 0; // Block mit mehreren Anweisungen wird ausgeführt,

ausgang2 = 1; // wenn die Bedingung hinter if nicht wahr ist

}

#### Mehrere Verzweigungen mit if

if (<bedingung1>) {

<anweisung1>;

…

} else if (<bedingung2>) {

<anweisung2>;

…

} else {

<anweisung3>;

…

}

#### Fallunterscheidung mit switch

Mit der switch-Anweisung kann aus einer **Reihe von Alternativen** ausgewählt werden. Es ist zulässig, dass mehrere Möglichkeiten gültig sind und dieselbe Wirkung haben. Sie werden nacheinander aufgelistet. Passt keine der Möglichkeiten, dann wird die **default**-Einstellung ausgeführt.

Achtung! Auf keinen Fall **break** vergessen!!!

switch (<vergleichswert>) {

case <wert1>:

<anweisung1>;

<anweisung2>;

…

break;

case <wert2>:

<anweisung3>;

<anweisung4>;

…

break;

…

default:

<anweisung5>;

…

}

**Beispiel:**

// In der Variablen *ergebnis* ist ein Messergebnis oder eine Zahl gespeichert.

// Abhängig vom genauen Wert sollen nun bestimmte Reaktionen erfolgen.

switch (ergebnis) {

case 0x00:

case 0x10:

case 0x20:

ausgang1 = 1;

break;

case 0x30:

ausgang1 = 0;

break;

case 0x40:

ausgang1 = ~ausgang1;

break;

default:

ausgang2 = 1;

break;

}

Hinweis: **Switch-Variablen** müssen einen **einfachen Datentyp** verwenden. Hinter **case** müssen **Konstanten** stehen. Dies können mit #define am Anfang des Programms deklariert werden.

**Beispiel:**

# define rechts 0x10 // ohne Semikolon!!

# define links 0x20

# define rechtskurve 0b0100

# define linkskurve 0b1000

unsigned char richtung;

…

switch (richtung) {

case rechts:

motor = rechtskurve;

break;

case links:

motor = linkskurve;

break;

default:

motor = vorwaerts;

break;

}

#### Operationen ( Unterprogramme, Funktionen)

**Deklaration von Operationen**

**Beispiele:**

void addieren(void); // ohne Rückgabewert, ohne Parameter

void zeitms(int msec); // ohne Rückgabewert, mit Parameter

float berechneQuadrat(float pQ); // mit Rückgabewert, mit Parameter

**Definition von Operationen**

**Beispiel:**

int a, result; // globale Variablen

void addieren(void) { // Operationsname

result = a + a; // Anweisung(en)

}

**Operationen mit Übergabewert**

**Beispiel:**

void zeitms(int msec) { // Übergabewert msec

int t1; // lokale Variable

for (t1=msec;t1!=0;t1--)

wait\_us(1000); // Zeitschleife;

}

**Operationen mit Rückgabewert**

**Beispiel:**

float berechneQuadrat(float pQ=10) { // Parameter mit Standardwert

return pQ\*pQ; // Rückgabewert

}

### Beispiel eines C/CPP-Programms

volatile bool Anforderung\_Fussgaenger = false; //globale Variablen

int z;

int S1 = PA4;

int LED\_rt = D1, LED\_ge = D2, LED\_gn = D3;

char phasen[4]= { //array

0b001, //0 rot

0b101, //1 rotgelb

0b010, //2 grün

0b100}; //3 gelb

void setup()

{

pinMode(LED\_rt, OUTPUT);

pinMode(LED\_ge, OUTPUT);

pinMode(LED\_gn, OUTPUT);

pinMode(S1, INPUT\_PULLUP);

z=0;

}

void loop()

{

if (digitalRead(S1) == LOW)

{

Anforderung\_Fussgaenger = true;

}

digitalWrite(LED\_rt,HIGH);

digitalWrite(LED\_gn,LOW);

digitalWrite(LED\_ge,LOW);

delay(500);

digitalWrite(LED\_ge,HIGH);

delay(500);

digitalWrite(LED\_gn, HIGH);

digitalWrite(LED\_rt, LOW);

digitalWrite(LED\_ge, LOW);

delay(500);

digitalWrite(LED\_gn, LOW);

digitalWrite(LED\_ge, HIGH);

delay(500);

if (Anforderung\_Fussgaenger) //Fußgängeranforderung auswerten

{

digitalWrite(LED\_rt, HIGH);

delay(500);

Anforderung\_Fussgaenger = false;

digitalWrite(LED\_rt, LOW);

}

z++;

if (z>8)

{

z=0;

}

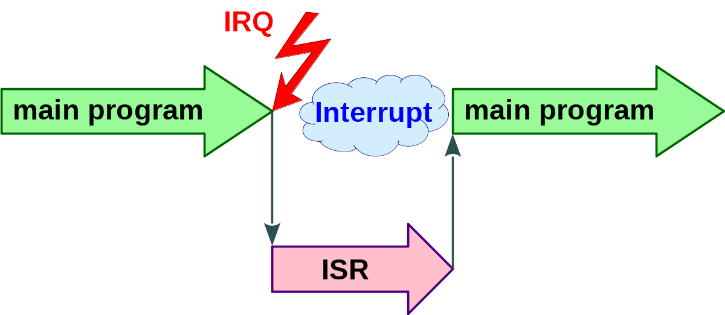
}

## On Chip Peripherie

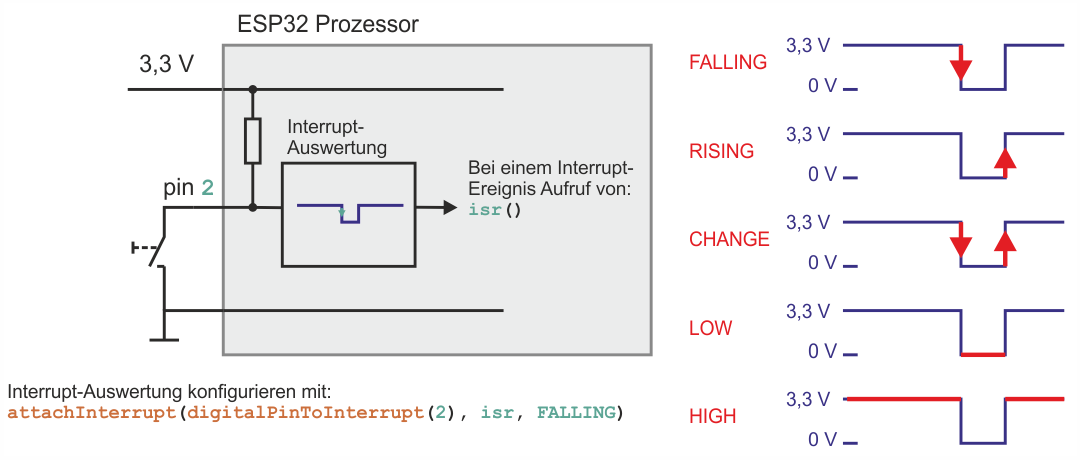
### Portpin: Eingabe und Ausgabe

|  |  |  |
| --- | --- | --- |
|  | **Befehl** | **Beispiel** |
| **Portausgabe:** Bsp.: GPIOA->ODR = 0xFF00; // Highbyte auf HIGH gesetzt… | | |
| **Einzelbitausgabe** | | |
| Deklaration | #define Name Pin | #define LED\_D12 D12 |
| Name = Pin-Bezeichnung | |
| Konfiguration | pinMode(LED\_D12, OUTPUT); | |
|  |  | |
| Verwendung | digitalWrite(name, WERT) WERT=HIGH, LOW | digitalWrite(LED\_rt,HIGH); |
| **Einzelbiteingabe** | | |
| Deklaration | #define Name Pin | #define S2 PA4 |
| Mögliche Werte für Pin: PA0..PA15, PB0..PB15, PC0..PC15 oder D1, A0… | |
| Konfiguration | pinMode(name, konfig); | pinMode (S2, INPUT); |
| Mögliche Werte fürkonfig: = INPUT\_PULLUP, INPUT\_PULLDOWN, INPUT | |
| Verwendung | Var = Pin Zustand lesen | buttonState = digitalRead(S2); |

### Externer Interrupt



|  |  |  |
| --- | --- | --- |
|  | **Befehl** |  |
| **Externer Interrupt** | | |
| Deklaration | #define Name Pin | #define S2 PA4 |
| Mögliche Werte für Portpin: PA0..PA15, PB0..PB15, PC0..PC15, oder D1, A0… | |
| Konfiguration | attachInterrupt (digitalPinToInterrupt (PIN), ISR\_Name, Aktion); | |
| Bsp.: | attachInterrupt (digitalPinToInterrupt (S2), ISR\_EXT\_IR, FALLING); | |
|  | Aktion: FALLING, RISING, CHANGE, HIGH, LOW | |
| Hinweis: | Variable(n) in der ISR sollten als volatile deklariert werden | |



### Timer

Autoreloadwert

Counter

0

t

Prinzip:

Systemtakt

Reset

Enable

Q D

1

16

16

Autoreloadregister ARR

Counter

EQ CMP

Prescaler

Freigabebit

Überlaufbit

Interrupt Enable Bit

MHz

Start

Zähltakt

Beispiel STM32L152

|  |  |
| --- | --- |
| Maßnahme | Syntax |
| Timerauswahl: | static HardwareTimer mytimer = HardwareTimer(TIM3); |
| TimerÜberlauf nach Zeit konfigurieren | mytimer.setOverflow(2000, MICROSEC\_FORMAT); [<= 16 Bit]   * Wenn Counter-Reg und ARR gleich sind |
| TimerÜberlauf nach Frequenz konfigurieren | mytimer.setOverflow(2000, HERTZ\_FORMAT); [<= 16 Bit] |
| zusätzlich Vorteiler nutzen: | mytimer.setPrescaleFactor(100); [<= 16 Bit] |
| TimerInterrupt aktivieren und ISR aufrufen | mytimer.attachInterrupt(ISR\_Timer); |
| TimerInterrupt deaktivieren | mytimer.detachInterrupt(); |
| Timer starten | mytimer.resume(); |
| Timer stoppen | mytimer.pause(); |
| Hinweis: | Variable(n) in der ISR sollten als volatile deklariert werden |

### Puls-Weiten-Modulation (PWM)

Autoreloadwert

Counter

0

t

Vergleichswert

PWM-Signal

Pulsewidth

periode = Periodendauer T

Prinzip:

|  |  |  |
| --- | --- | --- |
| **PWM** | **Befehl** | **Beispiel** |
| Deklaration | #define name (Portpin) | #define RGB\_bl D10 |
| Konfiguration f-Frequenz in Hz => 1/Periodendauer | analogWriteFrequency(var); | analogWriteFrequency(2000); //entspricht 2KHz |
| Bitbreite Pulsweite | analogWriteResolution(8-16); | analogWriteResolution (16); |
| Verwendung | analogWrite(Pinname, Pulsweite); | analogWrite(RGB\_r, 200); |

### Analog – Digital – Wandlung

|  |  |  |
| --- | --- | --- |
| **AD-Wandler** | **Befehl** | **Beispiel** |
| Deklaration | Datentyp Name Pin | #define A0\_Pin = A0; |
| Mögliche Werte für Portpin: A0-A5 | |
| Verwendung | Var = analogRead(Pin) | sensorValue = analogRead(A0\_Pin); |

A

D

Bitzahl z.B. 12

12

0

1

0

0

0

1

0

1

0

1

1

1

0..Umax

z.B. 3,3Vmax

Rohwandelwert

Ue

Analoge

Größe

Digitale Größe

**Berechnungsformeln**

z.B.

Wandelwert als Kommazahl: x = (Ue/Umax) z.B. x = (Ue/3.3)

Stufung (analoge Auflösung): Umax/4095 z.B. 3,3V/4095

Wandelwert als Ganzzahl linksbündig: unsigned short x = (Ue/Umax) \* 65535

### Digital – Analog – Wandlung

|  |  |  |
| --- | --- | --- |
|  | **Befehl** | **Beispiel** |
| **Digital-Analog Wandler** | | |
| Deklaration | AnalogOut meinAnalogOut(Portpin); | AnalogOut ausgang(ledPin); |
| Mögliche Werte für Portpin PA\_5 | |
| Verwendung | analogWrite(Pin, Wert) | analogWrite(ledPin, outputValue); |

Bitzahl z.B. 16

D

A

0

1

0

0

0

1

0

1

0

1

1

1

Ausgabewert

16

Ua

Analoge

Größe

1

1

1

1

0..3,3V

Digitale Größe

**Berechnungsformeln**

float x:

uint16\_t:

Werteanpassung / Skalierung mit map():

Werteanpassung = map(Wert, 0, 1023, 0, 255); // Werte von 0-1023 werden auf 0-255 angepasst

## Externe Kommunikationsmöglichkeiten

### Universal Asynchronous Receiver Transmitter (UART)

Mikrocontroller

UART

Modul

TX

RX

RX

TX

|  |  |  |
| --- | --- | --- |
|  | **Befehl** | **Beispiel** |
| **Universal Asynchronous Receiver Transmitter (UART)** | | |
| **#include <SoftwareSerial.h> => Software Serial Modus** | | |
| Deklaration | #define softserial  #define RX *Pin?*  #define TX *Pin?* |  |
| SoftwareSerial SerialBsp(RX, TX); | |
| Verwendung  Daten empfangen | SerialBsp.begin(9600); // Baudrate  Fragen nach daten im Serial-Buffer  Wenn ja, dann in Variable lesen… | if (SerialBsp.available())  {  msg = SerialBsp.readString(); |
| Daten senden | Mit print(ln) String schreiben  Mit print Var-wert schreiben | SerialBsp.print("LED an");  SerialBsp.print(LED); |
|  |  |  |
|  |  |  |

**Frame**

Start

Data

Data

Data

Parity

Stop

0 oder 1

0 oder 1

0 oder 1

0 oder 1

0

1

Rx oder Tx

Eine UART-Übertragung beginnt immer mit einem Startbit (Low). Darauf folgen

* 5-8 **Datenbits** (Standard = 8)
* 0 oder 1 **Paritybit** (Standard = 0 none)
* 1 oder 2 **Stopbits** (Standard =1)

Falls ein Paritybit programmiert wurde, kann es gerade Parity (even) oder ungerade Parity (odd) anzeigen.

### Serial Peripheral Interface (SPI)

Das **Serial Peripheral Interface** (**SPI**) dient der Kommunikation des Mikrocontrollers mit **Modulen** auf der Platine. Module sind

Mikrocontroller

SPI

Master

SPI

Slave

MOSI

MISO

SCLK

MOSI

MISO

SCLK

* Anzeigen,
* Speicher,
* LAN-Bausteine
* …

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung**Signale**

Sendeleitung

Empfangsleitung

Auswahl Slaves/Chip (Lowaktiv)

SCLK (Serial Clock): Taktleitung

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

|  |  |  |
| --- | --- | --- |
|  | **Befehl** | **Beispiel** |
| #include <SPI.h> //Bibliothek einbinden | | |
| Initialisiere SPI | SPI.begin() |  |
| SPISettings(14000000, MSBFIRST, SPI\_MODE0) f, dataOrder, Modus  Oder:  SPI.setBitOrder(MSBFIRST);  SPI.setClockDivider(SPI\_CLOCK\_DIV32);  SPI.setDataMode(SPI\_MODE3);  SPI.end(); | |
| Konfiguration CS-Pin | #define chipSelectPin (CS) = D5 | |
| pinMode(chipSelectPin, OUTPUT); | |
| Verwendung | SPI.begin();  digitalWrite(CS, LOW);  SPI.transfer(address\_w);  SPI.transfer(0x09);  SPI.transfer(Bitmuster);  digitalWrite(CS, HIGH);  SPI.end(); |  |
|  |  | |

### Inter-Integrated Circuit (I2C) SCL (Serial Clock): Taktleitung SDA (Serial Data): Datenleitung

.

Mikrocontroller

I2C

Master

I2C

Slave

SCL

SDA

SCL

SDA

I2C

Slave

SDA

SCL

I2C

Slave

SDA

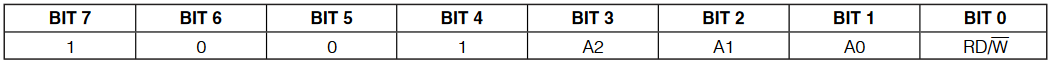
SCL

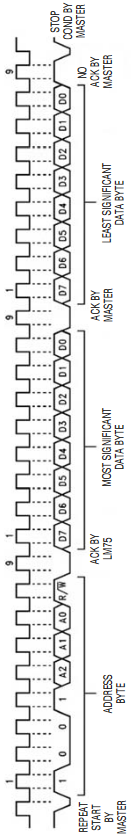
…

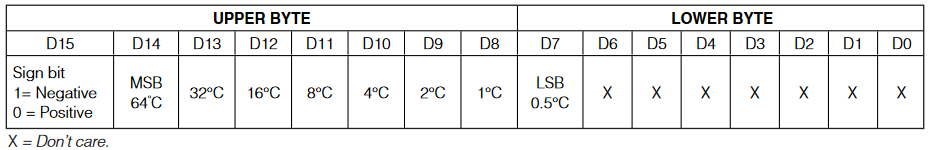
|  |  |  |
| --- | --- | --- |
| **I2C** | **Befehl** | **Beispiel** |
|  | #include <Wire.h> => I2C-Library | JE NACH Datenblatt! |
| Deklaration | #define adress => 7 Bit! | address 0b0100000 //0x40 |
| Initialisierung | Bibliothek starten  I2C-Frequenz einstellen | Wire.begin();  Wire.setClock(10000);. |
| Verwendung  Daten senden  Daten empfangen | I2C-Übertragung starten inkl. Adresse Baustein  Daten auf Bus schreiben  Übertragung abschließen  I2C-Übertragung starten inkl. Adresse Baustein  Daten auf Bus schreiben  Nach Empfangenen Daten fragen…  Var = daten lesen  Übertragung beenden | Wire.beginTransmission(address);  Wire.write(0x09);  Wire.endTransmission();  Wire.beginTransmission(address);  Wire.write(0x00);  Wire.endTransmission();    Wire.requestFrom(address, 2);  Temp\_H = Wire.read();  Temp\_L= Wire.read();  Wire.endTransmission(); |
|  |  |  |

Beispielhaft aufgeführte I2C-Bausteine bzw. Auszug Datenblätter Quelle: www.alldatasheet.com

**LM 75:**







**PCF 8574**

