Formelsammlung

1.5.2 TG Informationstechnik

Lokale Formelsammlung HW: Arduino-IDE

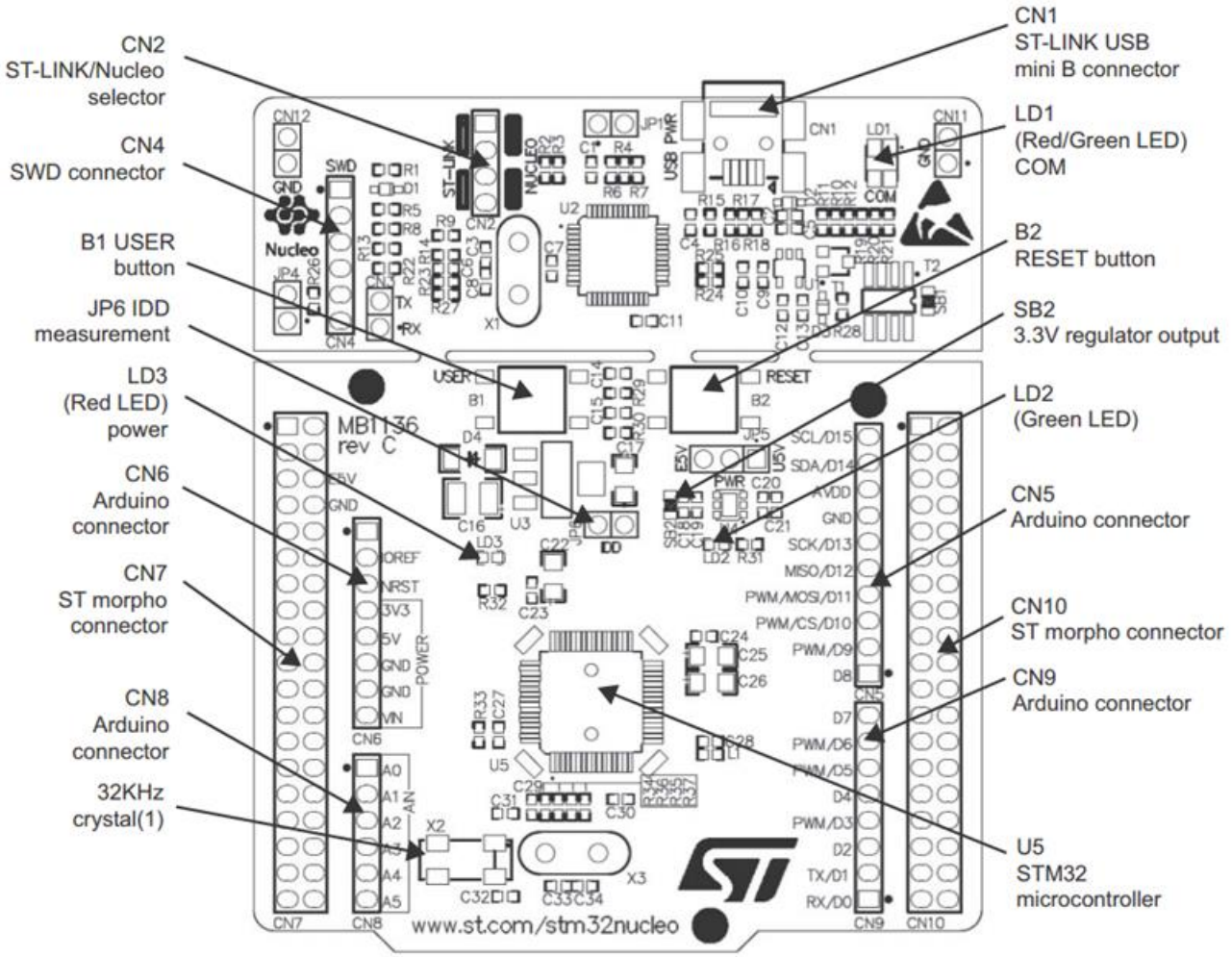
Version: V 4.40

Gültig ab Abitur 2024

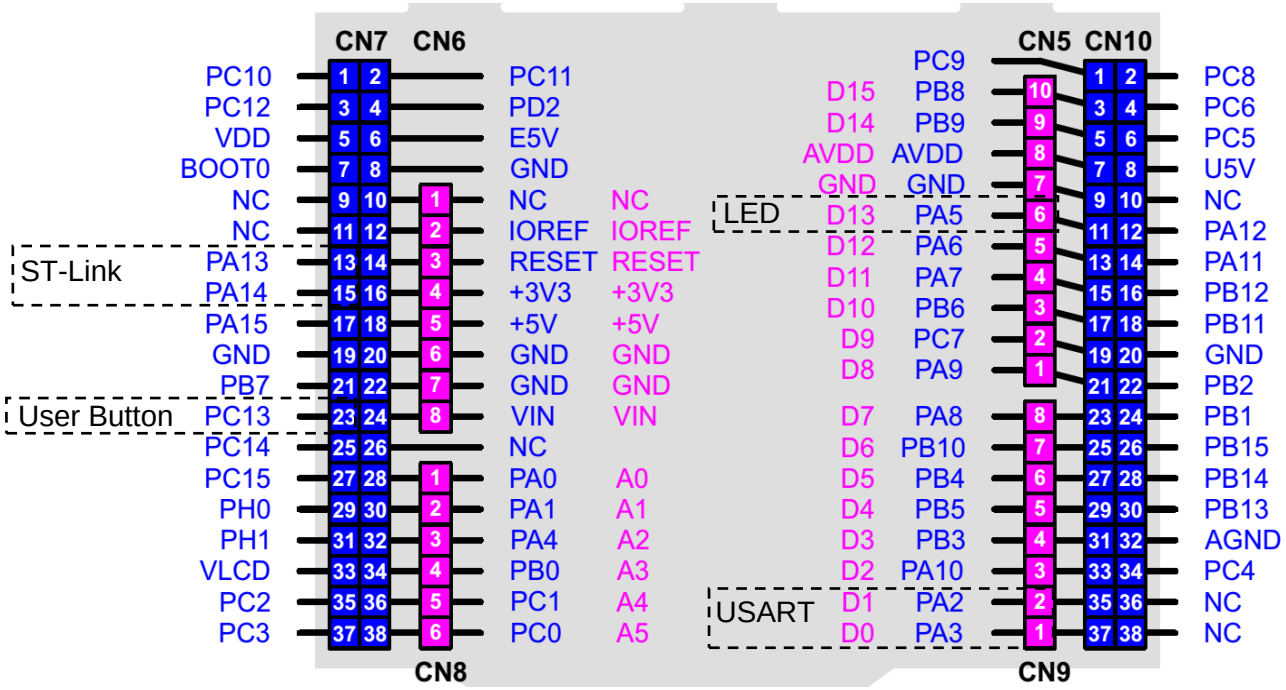
Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Inhaltsverzeichnis:

1 Trainingsplatine STM32 Nucleo L152RE

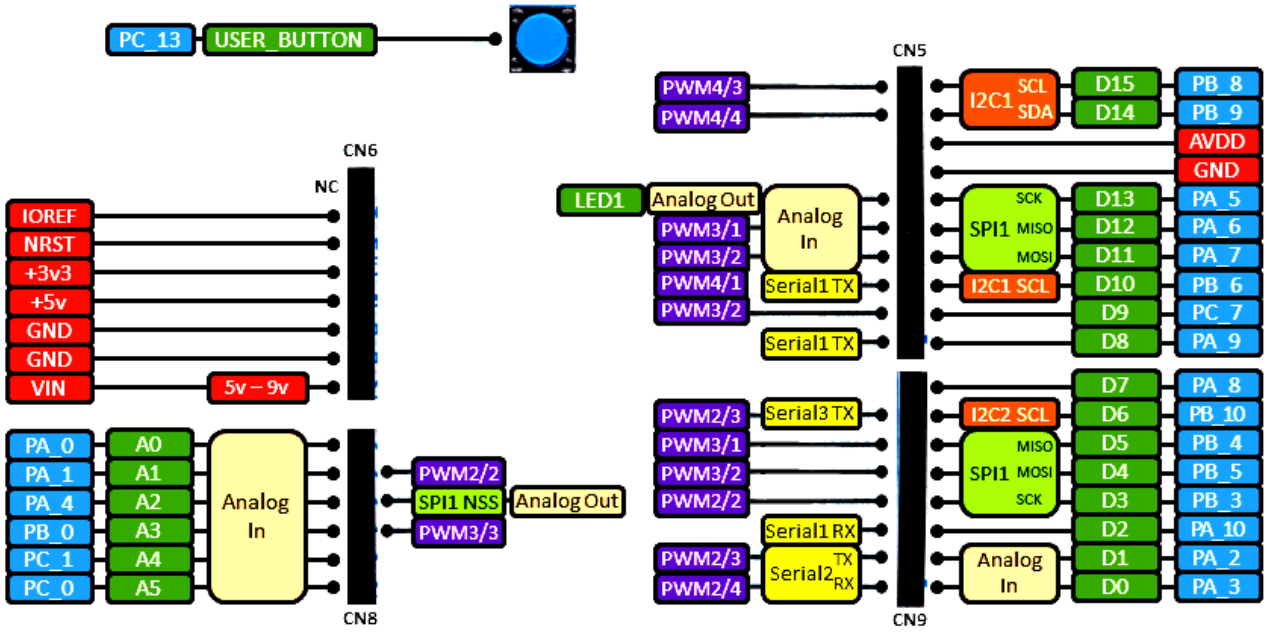


1.1 Pinbelegung

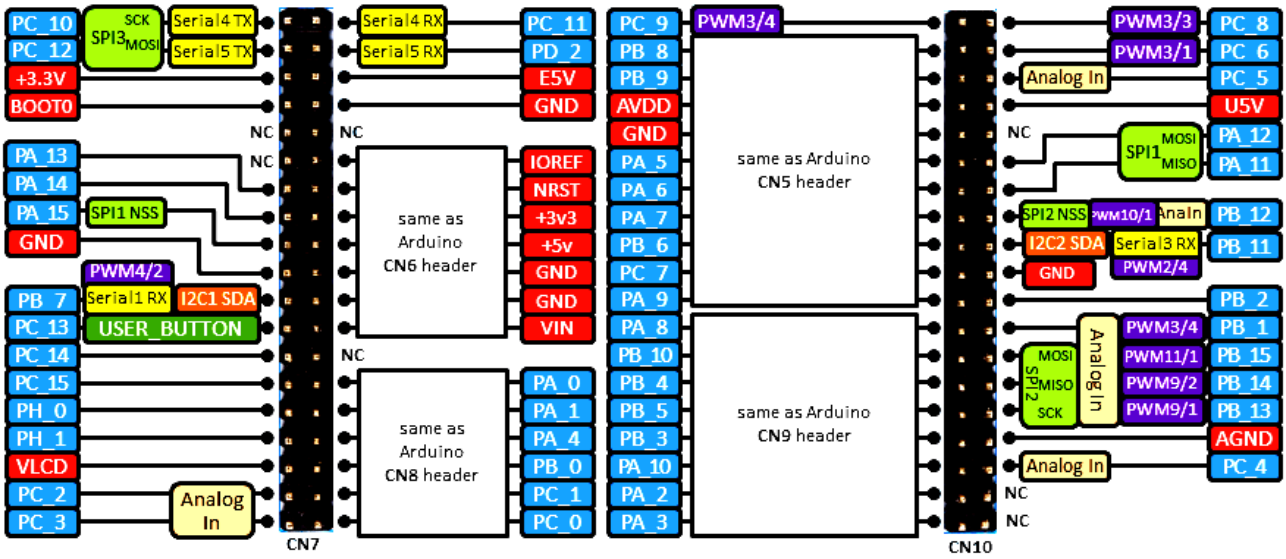


1.2 Layout

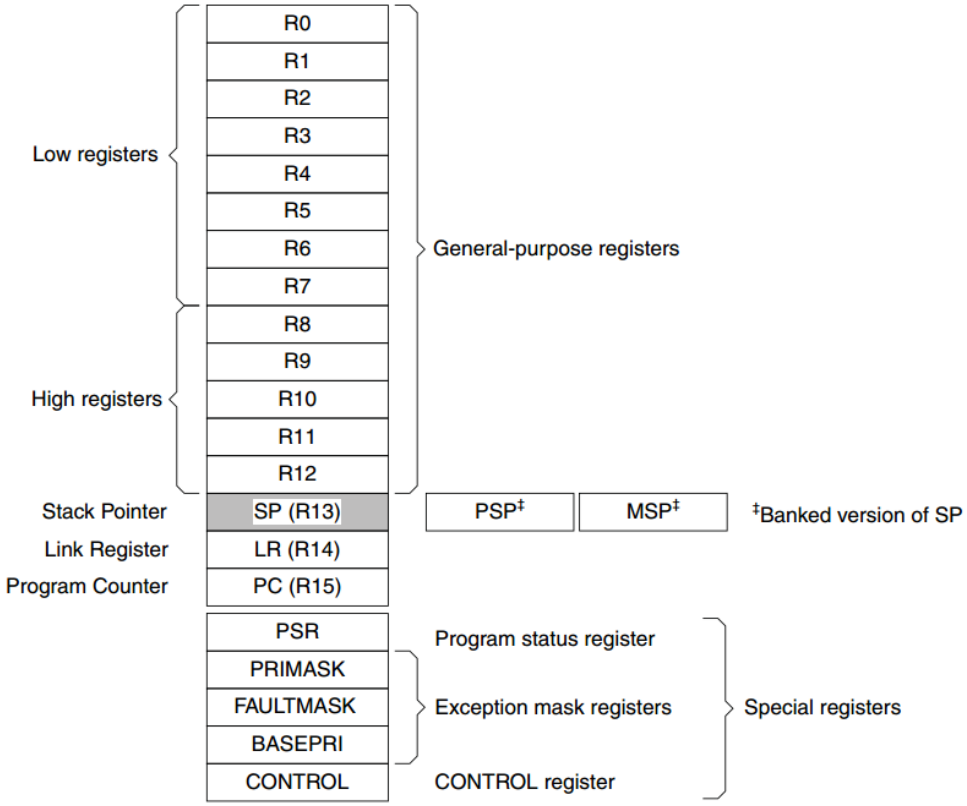
Nucleo L152RE
Arduino Headers



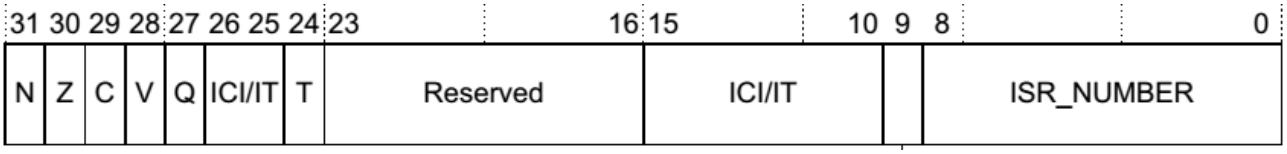
Nucleo L152RE
Morpho Headers



1.3 Core Register mit PSR

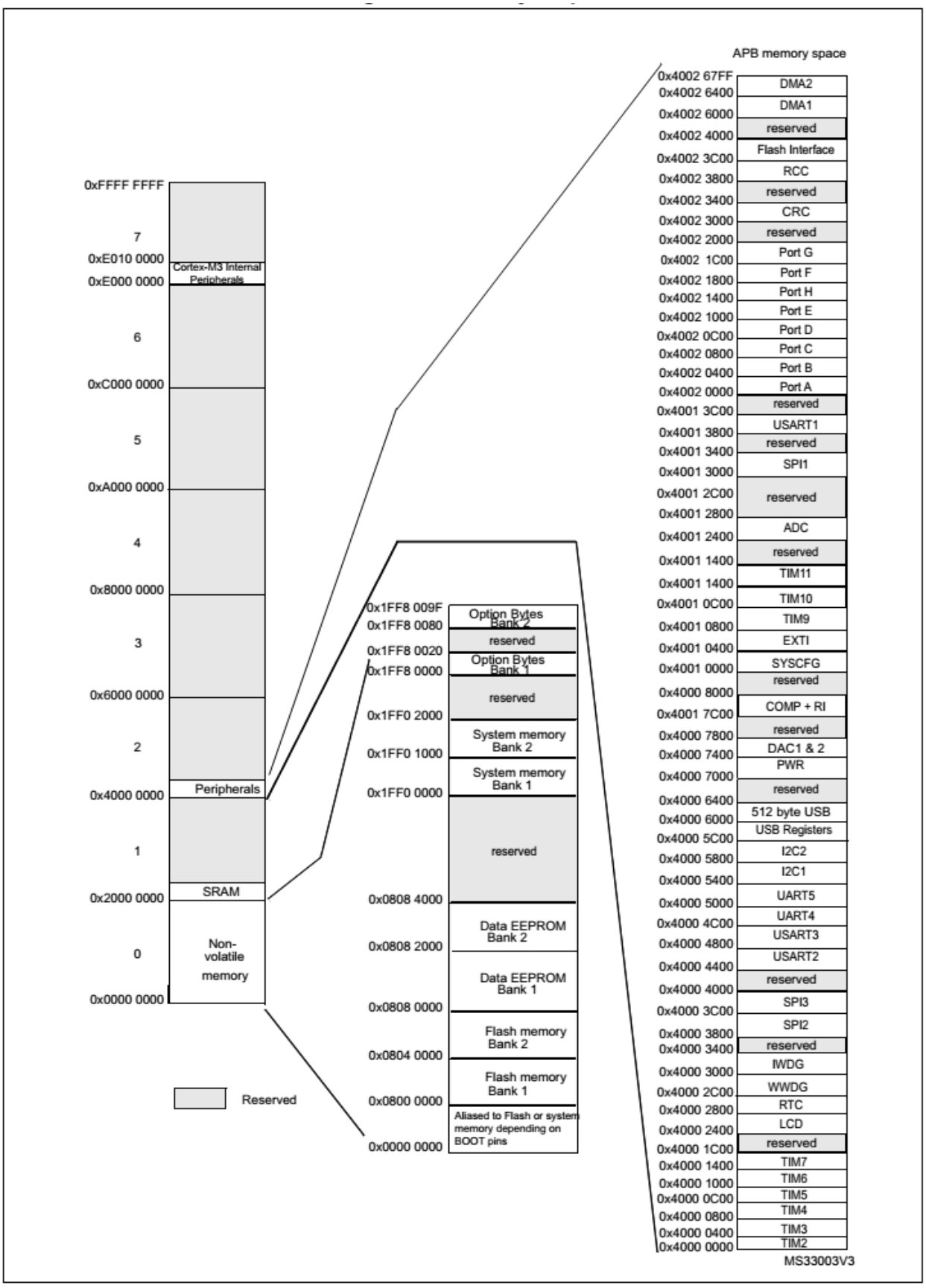


R-Nr	Zweck
R0-12	General Purpose Register für Datenoperationen (Load-Store-Arithmethic-Logic)
R13	Stack-Pointer SP
R14	Link-Register LR
R15	Programm Counter PC (halfword aligned-2er Schritte)



PSR	Programm Status Register
Bit 31	N: Negative or less flag 0: Ergebnis Operation positiv, 0, oder größergleich als … 1: Ergebnis Operation negativ oder kleiner als …
Bit 30	Z: Zero flag 0: Ergebnis Operation war ungleich Null 1: Ergebnis Operation war gleich Null
Bit 29	C: Carry or borrow flag 0: Addition ergibt keinen Übertrag, Subtraktion ergibt Ergebnis <0 1: Addition ergibt einen Übertrag, Subtraktion ergibt Ergebnis >=0
Bit 28	V: Overflow flag 0: Operation ergibt keinen Überlauf Wertebereich 1: Operation ergibt einen Überlauf Wertebereich

1.4 Memory Map



2 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
`long`	32	+	0 .. 4294967295
`float`	32	- +	-3,4E38 .. 3,4E38
`enum` Aufzählungstyp			enum {AUTOMATK, HAND} Zustand = AUTOMATIK;

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	
++	Inkrement
--	Dekrement
-	Vorzeichen
*	Multiplikation
/	Division
%	Modulo, Rest der Division
+	Plus
-	Minus

Priorität
Höchste
Niedrigste

Vergleichs- und logische Operatoren	
!	NOT
>	Größer
>=	Größer gleich
<	Kleiner
<=	Kleiner gleich
==	Gleich
!=	Ungleich
&&	AND
||	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`

2.1 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);
```

2.2 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>;      … break;
}
```

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. Diese 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;
}
```

2.3 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
}
```

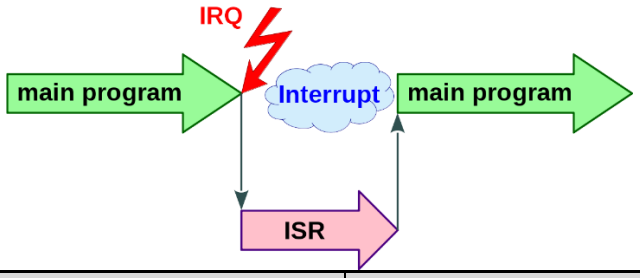
2.4 Beispiel eines C/CPP-Programms

```
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;
        }
}
```

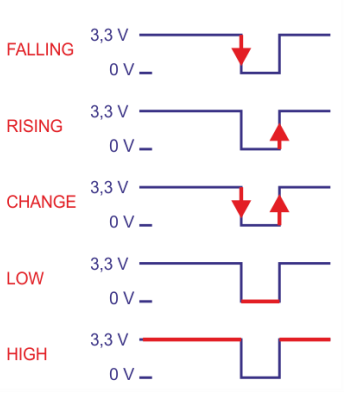
2.5 Portpin: Eingabe und Ausgabe

	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 für `konfig: = INPUT_PULLUP, INPUT_PULLDOWN, INPUT`	
Verwendung	Var = Pin Zustand lesen	`buttonState = digitalRead(S2);`

2.6 Externer Interrupt

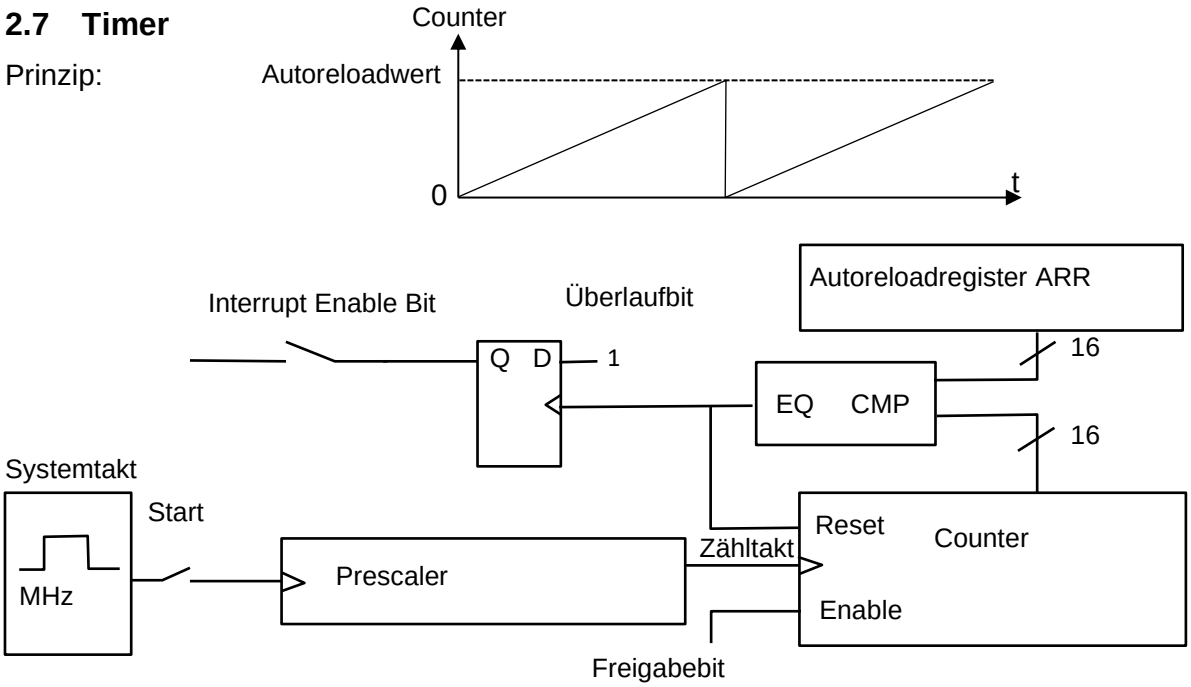


	Befehl	
Externer Interrupt		
Deklaration	`#define Name Pin`	`#define S2  PA4`
	Mögliche Werte für `Portpin: PA0..PA15, PB0..PB15, PC0..PC15, oder D1, A0…`	
Konfiguration	attachInterrupt (digitalPinToInterrupt (PIN), ISR_Name, Aktion);	
Bsp.:	attachInterrupt (digitalPinToInterrupt (S2), ISR_EXT_IR, FALLING);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



2.7 Timer

Prinzip:



Beispiel STM32L152

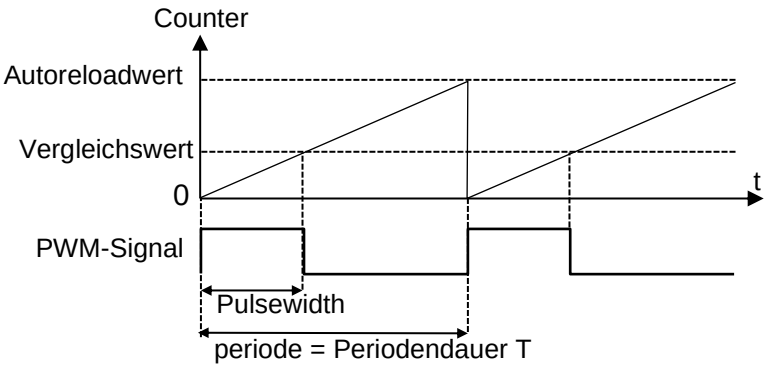
Maßnahme	Syntax
Timerauswahl:	`static HardwareTimer mytimer = HardwareTimer(TIM3);`

TimerÜberlauf nach Zeit konfigurieren	`mytimer.setOverflow(2000, MICROSEC_FORMAT); [<= 32 Bit]`
	⇨ Wenn Counter-Reg und ARR gleich sind
	`Prescaler automatisch konfiguriert!`

TimerÜberlauf nach Frequenz konfigurieren	`mytimer.setOverflow(2000, HERTZ_FORMAT); [<= 32 Bit]` `Prescaler automatisch konfiguriert!`
TimerÜberlauf nach f-Ctr	`Mytimer.setOverflow(50000); Prescaler Konfigurierbar!`
zusätzlich Vorteiler nutzen:	`mytimer.setPrescaleFactor(32); [<= 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

2.8 Puls-Weiten-Modulation (PWM)

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`
Pulsewidth Bitbreite	`analogWriteResolution(8-16);`	`analogWriteResolution (16);`
Verwendung	analogWrite(Pinname, Pulsweite);	`analogWrite(RGB_r, 200);`

2.9 LCD

```
lcd.begin(16, 2);         // 16 Zeichen, 2 Zeilen
lcd.clear();              // löschen
lcd.setBacklight(255);    // Hintergrundlicht
lcd.setCursor(0,0);       // Cursorposition (Spalte, Zeile)
lcd.print("SOS_IR...");   // Textausgabe
```

Ausgabevariante am LCD mit zusammengesetzten Strings:

```
I2C_LCD(" " + String(Std) + ":" + String(Min) + ":" + String(Sec));
```

Ausgabevariante am LCD mit Standard-Funktion sprintf();

```
char buf [16];
sprintf(buf, "%02u : %02u : %02u", Std, Min, Sec);
I2C_LCD(buf);
```

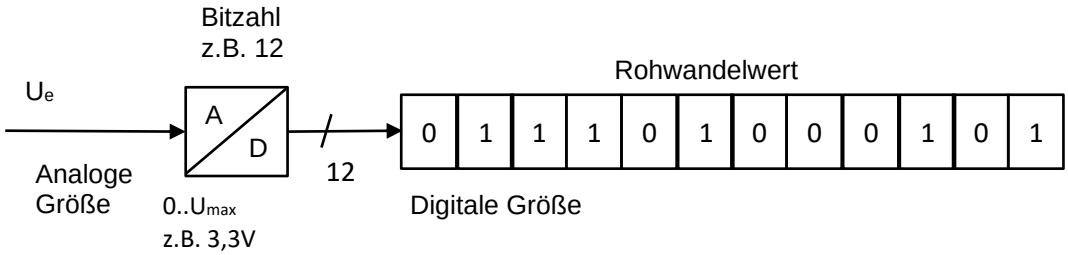
Bei Umlauten und Sonderzeichen müssen diese jeweils im Displaycode eingefügt werden:
Bsp.: Würfelz\xe1hler

Hinweis: **\xe1** ist der Displaycode für den Umlaut **ä**. Weitere Zeichencodes:

ö: \xef	ß: \xe2	σ: \xe5
ü: \xf5	µ: \xe4	α: \xe0
°: \xdf	€: \xd3	ε: \xe3

2.10 Analog – Digital – Wandlung

AD-Wandler	Befehl	Beispiel
Deklaration	`Datentyp Name Pin`	`#define A0_Pin A0`
	analogReadResolution(10, 12, 16); //Bitbreite	analogReadResolution(12);
	Mögliche Werte für `Portpin`: A0-A5	
Verwendung	`Var = analogRead(Pin)`	`sensorValue = analogRead(A0_Pin);`



Berechnungsformeln:

- Anzahl der Stufen: 2^n
- Stufen: 0 - (2^n - 1)
- kleinste Spannungsstufe => U_{LSB}: U_{VCC} / 2^n bzw. U_{ref} / 2^n
- Aktueller Wandelwert (Rohwandelwert): $x = U_e/U_{VCC} * (2^n - 1)$

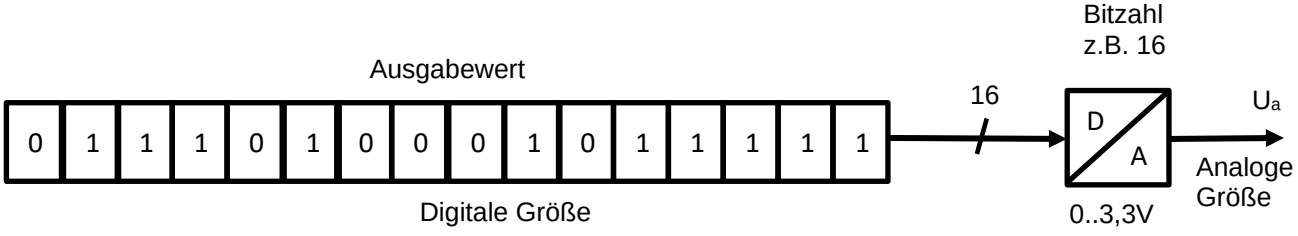
Werteanpassung / Skalierung mit map():

Werteanpassung, Bsp.:

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

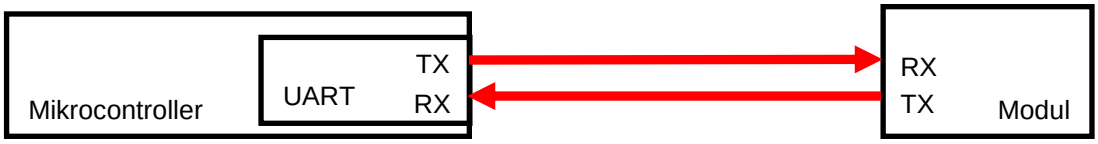
2.11 Digital – Analog – Wandlung

	Befehl	Beispiel
Digital-Analog Wandler		
Deklaration	`AnalogOut meinAnalogOut(Portpin);`	`AnalogOut ausgang(ledPin);`
	Mögliche Werte für `Portpin`: PA5	
Verwendung	`analogWrite(Pin, Wert)`	`analogWrite(ledPin, outputValue);`



2.12 Externe Kommunikationsmöglichkeiten

Universal Asynchronous Receiver Transmitter (UART)



	Befehl	Beispiel
Universal Asynchronous Receiver Transmitter (UART)		
#include <SoftwareSerial.h> => Software Serial Modus		

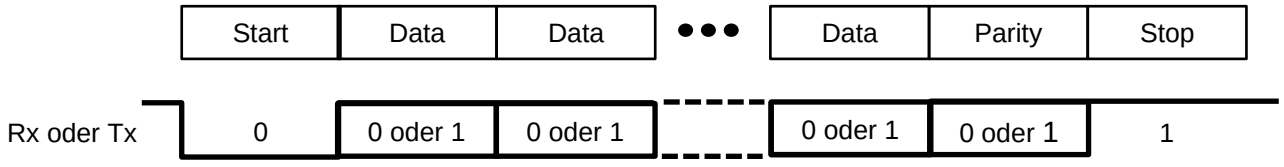
Deklaration	`#define softserial`
	`#define RX Pin?`
	`#define TX Pin?`	

	SoftwareSerial SerialBsp(RX, TX);	
Verwendung	SerialBsp.begin(9600); // Baudrate	

Daten empfangen	`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



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

- 5-8 **Data-Bits** (Standard = 8)
- 0 oder 1 **Parity-Bit** (Standard = 0 none)
- 1 oder 2 **Stop-Bbit** (Standard =1)

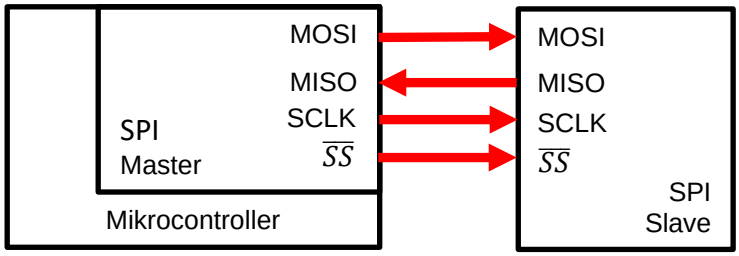
Falls ein Paritybit programmiert wurde, kann es gerade Parity (even) oder ungerade Parity (odd) anzeigen. Bsp:

Informationswort	Summe der Einsen	Paritätsbit / Codewort	
		bei Even-Parity	bei Odd-Parity
0011.1010	gerade	0 / 0011.1010 **0**	1 / 0011.1010 **1**
1010.0100	ungerade	1 / 1010.0100 **1**	0 / 1010.0100 **0**

Serial Peripheral Interface (SPI)

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

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



Signale

Master/Slave (OLD)	Controller/Peripheral (NEW)
Master In Slave Out (MISO)	Controller In, Peripheral Out (CIPO)
Master Out Slave In (MOSI)	Controller Out Peripheral In (COPI)
Slave Select pin (SS)	Chip Select Pin (CS)

Sendeleitung

Empfangsleitung

Auswahl Slaves/Chip (Lowaktiv)

SCLK (Serial Clock): Taktleitung

Mode	Clock Polarity (CPOL)	Clock Phase (CPHA)	Output Edge	Data Capture
SPI_MODE0	0	0	Falling	Rising
SPI_MODE1	0	1	Rising	Falling
SPI_MODE2	1	0	Rising	Falling
SPI_MODE3	1	1	Falling	Rising

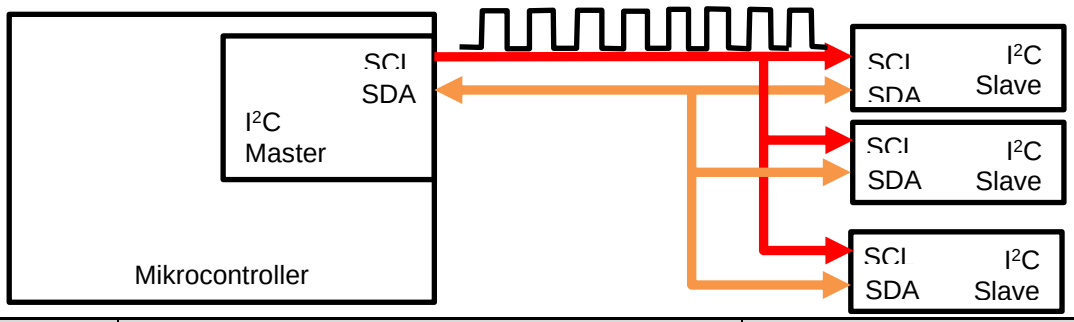
	Befehl	Beispiel
`#include <SPI.h>  //Bibliothek einbinden`		
Initialisiere SPI	`SPI.begin()`	

	`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	`Bsp: Schreiben:`
	`digitalWrite(CS, LOW);`
	`SPI.transfer(address_w);`
	`SPI.transfer(0x09);`
	`SPI.transfer(Bitmuster);`
	`digitalWrite(CS, HIGH);`
	`SPI.end();`
		`Bsp.: Lesen:`
		`digitalWrite(CS, LOW);`
		`ADC_H = SPI.transfer(0x00);`
		`ADC_L = SPI.transfer(0x01);`
		`digitalWrite(CS, HIGH);`

Inter-Integrated Circuit (I^2C) SCL (Serial Clock): Taktleitung SDA (Serial Data): Datenleitung

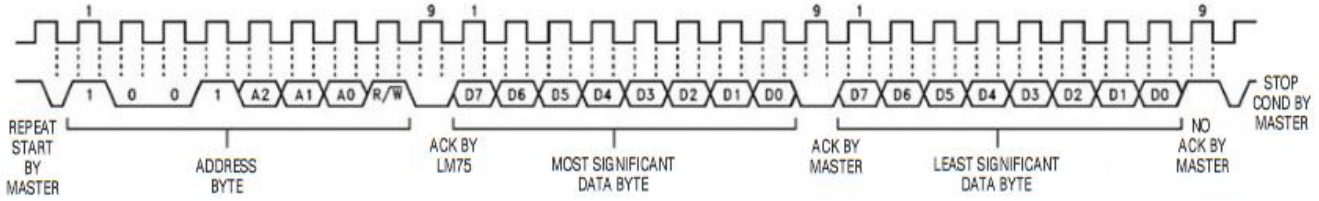


I^2C	Befehl	Beispiel
	#include <Wire.h> => I^2C-Library	JE NACH Datenblatt!
Deklaration	#define adress => 7 Bit!	address 0b0100000
Initialisierung	Bibliothek starten I^2C-Frequenz einstellen	Wire.begin(); Wire.setClock(10000);.
Verwendung	I^2C-Übertragung starten inkl. Adresse Baustein	Wire.beginTransmission(address);
Daten senden	Daten auf Bus schreiben	Wire.write(0x09);
	Übertragung abschließen	Wire.endTransmission();
Daten empfangen	I2C-Übertragung starten inkl. Adresse Baustein	Wire.beginTransmission(address);
	Daten auf Bus schreiben	Wire.write(0x00);
		Wire.endTransmission();
	Nach Empfangenen Daten fragen…	Wire.requestFrom(address, 2);
	Var = daten lesen	Temp_H = Wire.read();
	Übertragung beenden	Temp_L= Wire.read();
		Wire.endTransmission();
	Daten aus einem bestimmten Register lesen:	Wire.beginTransmission(address);
	Slave Registeradresse mitteilen	int WR_Byte = Wire.write(0x??);
		Wire.endTransmission();
	Inhalt des Registers in var lesen	Wire.requestFrom(address, WR_Byte);
		var = Wire.read();
		Wire.endTransmission();
ReStart gemäß Datenblatt	Wenn Restart gemäß I^2C-Frame benötigt:	Wire.endTransmission();

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

LM 75:

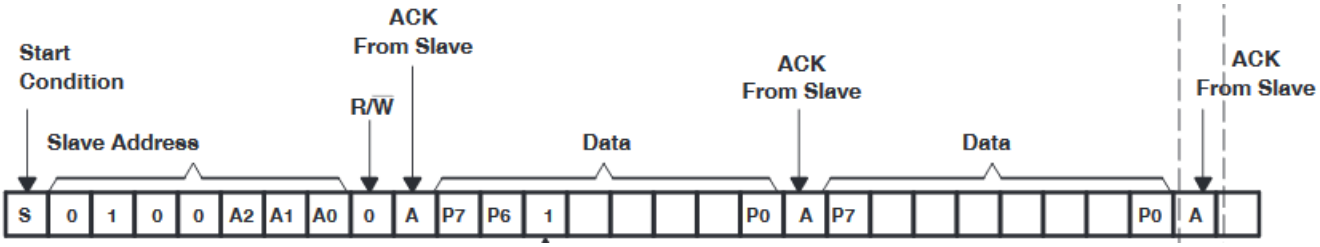
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
1	0	0	1	A2	A1	A0	RD/$\overline{W}$



UPPER BYTE								LOWER BYTE							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Sign bit 1= Negative 0 = Positive	MSB 64°C	32°C	16°C	8°C	4°C	2°C	1°C	LSB 0.5°C	X	X	X	X	X	X	X

X = *Don't care.*

PCF 8574



UPPER BYTE								LOWER BYTE							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Sign bit 1= Negative 0 = Positive	MSB 64°C	32°C	16°C	8°C	4°C	2°C	1°C	LSB 0.5°C	X	X	X	X	X	X	X

X = *Don't care.*

2.13 Glossar

Acknowlege	Quittierung
ALU	Arithmetisch-Logische Einheit
Ax	Analogeingang Pin
PWM	Analogausgang Pin
BCD	Binär Codiert Dezimal
BLDC-Motor	Bürstenloser Gleichstrommotor, Brushless DC-Motor
Bluetooth	Funkstandard zur Datenübertragung
Carry	Übertrag
CISC	Complex Instructionset Computer

CPU	Central Processing Unit
CS	Steuerleitung für Chip Select
CTR	Counter
DA-Wandler	Digital-Analog-Wandler
DEMUX	Demultiplexer
%2	Modulo 2
EN	Enable, Freigabe
EPROM	Erasable Programmable Read Only Memory
EVA	Eingabe Verarbeitung Ausgabe
Even	gerade
Frame	Rahmen
GPIO	General Purpose Input Output
Hardware Timer	16-Bit Timer
LED	Light Emitting Diode Leuchtdiode
LOAD	laden
MUX	Multiplexer
NVIC	Nested Vector Interrupt Controller
Odd	Ungerade
OE	Steuerleitung für Output Enable
Overflow	Überlauf
Parity	Geradzahligkeit
Periode	Periodendauer
Poti	Potentiometer Einstellwiderstand für analoge Eingabe
Pulsewidth	Pulsbreite
PWM	Puls-Weiten-Modulation
R0 usw.	Prozessorregister
RAM	Random Access Memory
RD	Steuerleitung für lesen
RISC	Reduced Instructionset Computer
ROM	Read Only Memory
Rx	Receive
SPI	Serial Peripheral Interface
SRG	Schieberegister
SS	Slave Select
Stack	Stapel
Stackpointer	Stapelzeiger
Tx	Transmit
UART	Universal Synchronous Asynchonous Receiver Transmitter
WR	Steuerleitung für schreiben