

Formelsammlung

1.5.2 TG Informationstechnik

Lokale Formelsammlung HW: Arduino-Framework

Version: V 5.1

Gültig **ab** Abitur 2026

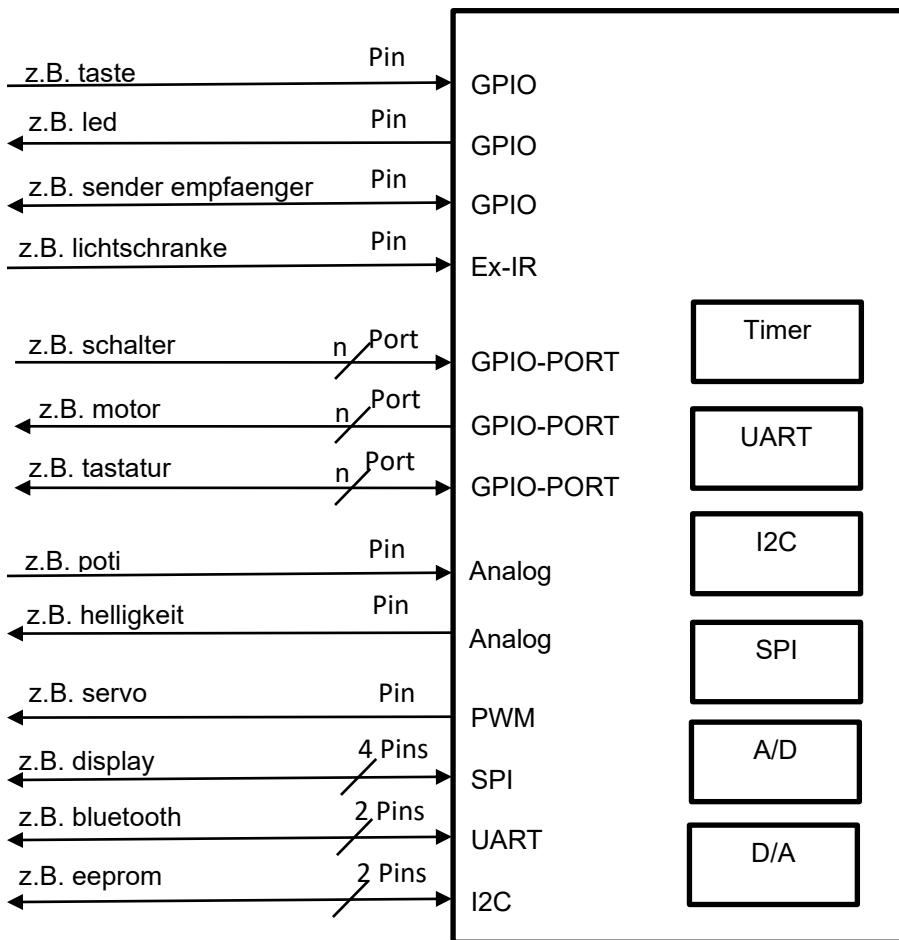
Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Inhaltsverzeichnis:

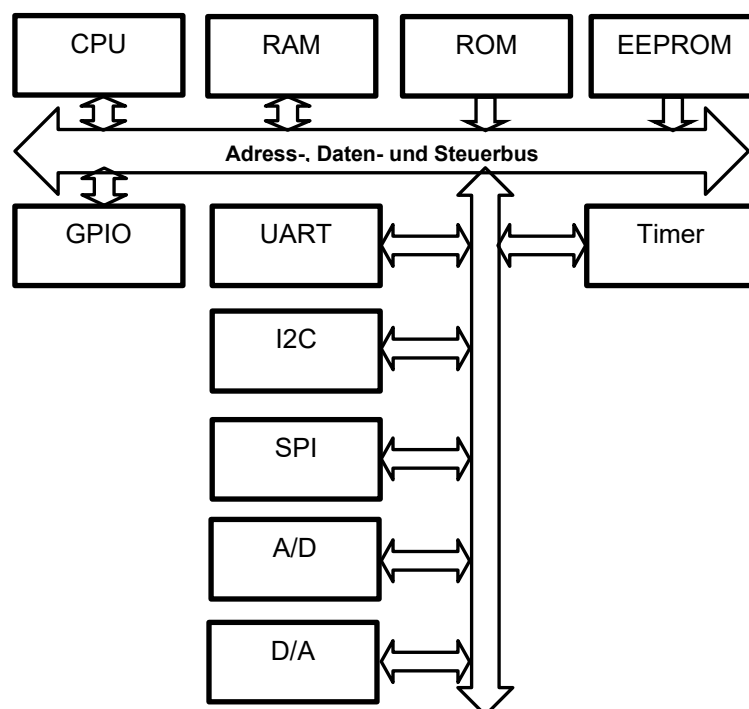
Hardware – Mikrocontrollertechnik Blockschaltbild „Prüfungscontroller“	3
Trainingsplatine STM32 Nucleo L152RE	4
Pinbelegung	4
Layout	4
Hochsprache C/CPP	6
Datentypen	6
Zeiger und Referenzen	6
Operatoren	6
Schleifen	7
Programmverzweigungen	8
Operationen (Unterprogramme, Funktionen)	10
String-Klasse – Methodenübersicht	10
Beispiel eines C/CPP-Programms	11
Portpin: Eingabe und Ausgabe	12
Externer Interrupt	12
Timer	13
Puls-Weiten-Modulation (PWM)	14
LCD	14
Analog – Digital – Wandlung	15
Digital – Analog – Wandlung	15
Externe Kommunikationsmöglichkeiten	16
Universal Asynchronous Receiver Transmitter (UART)	16
Serial Peripheral Interface (SPI)	17
Inter-Integrated Circuit (I ² C)	18
Allgemeine Coding Empfehlung/Vorschlag für HW-Programmierung C/C++	20

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Hardware – Mikrocontrollertechnik Blockschaltbild „Prüfungscontroller“

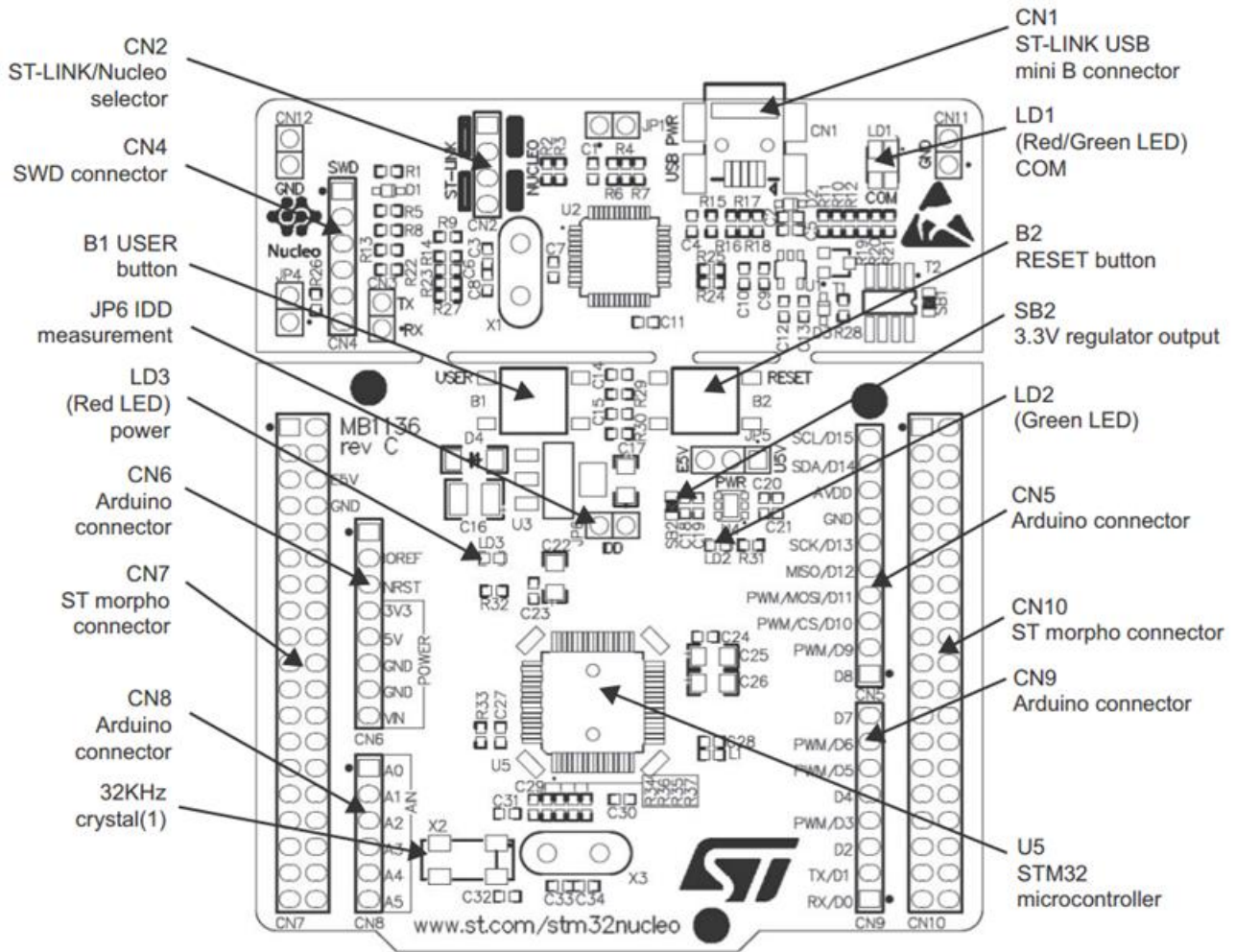


Blockbild Mikrocontroller:

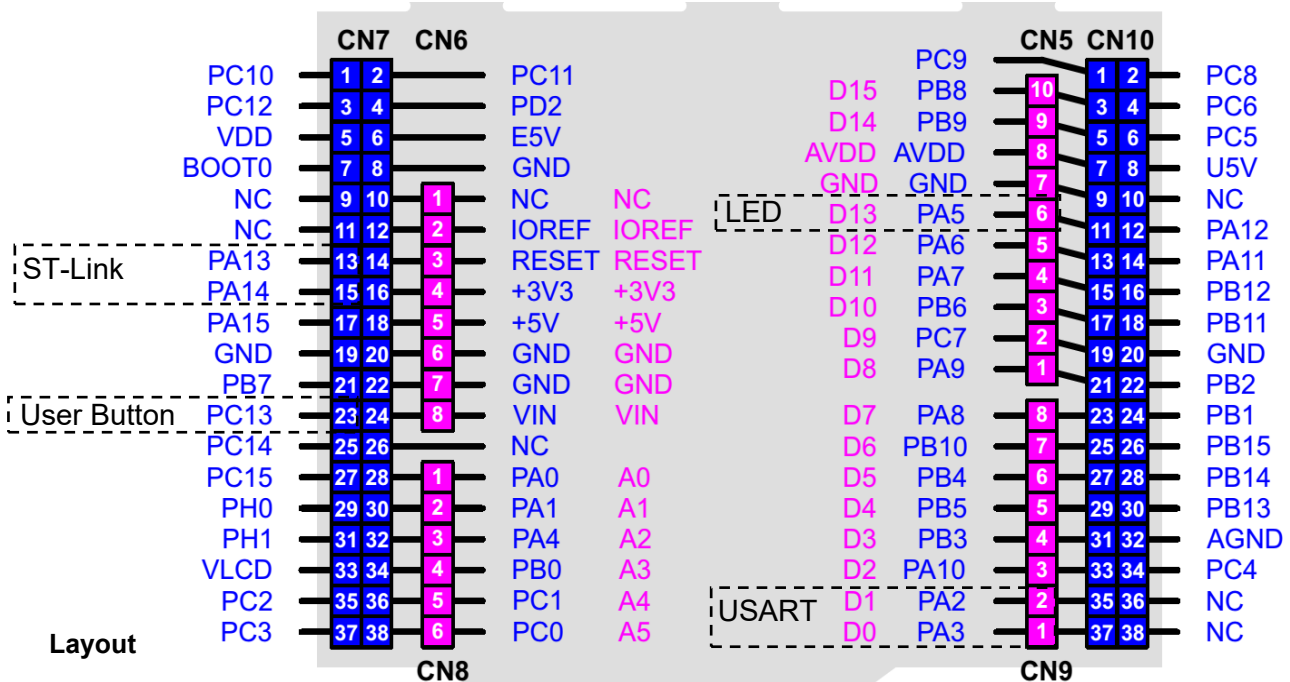


Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Trainingsplatine STM32 Nucleo L152RE

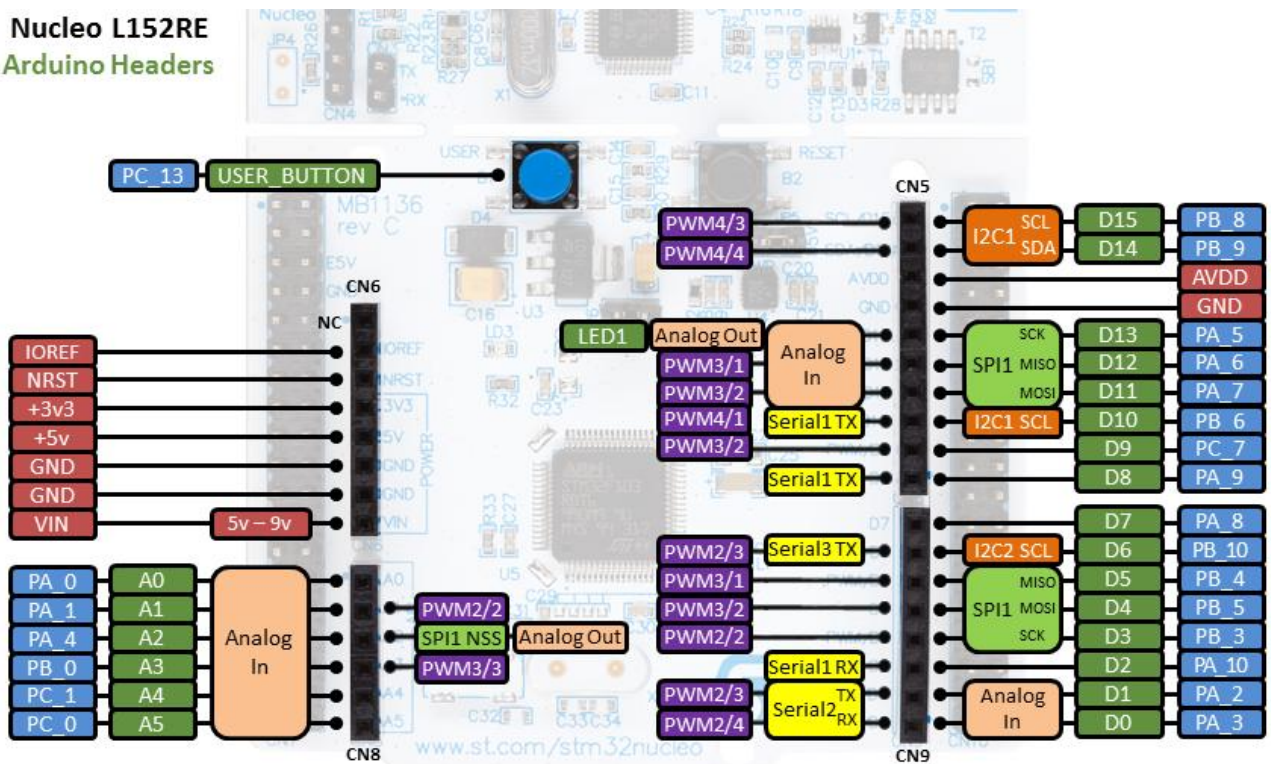


Pinbelegung

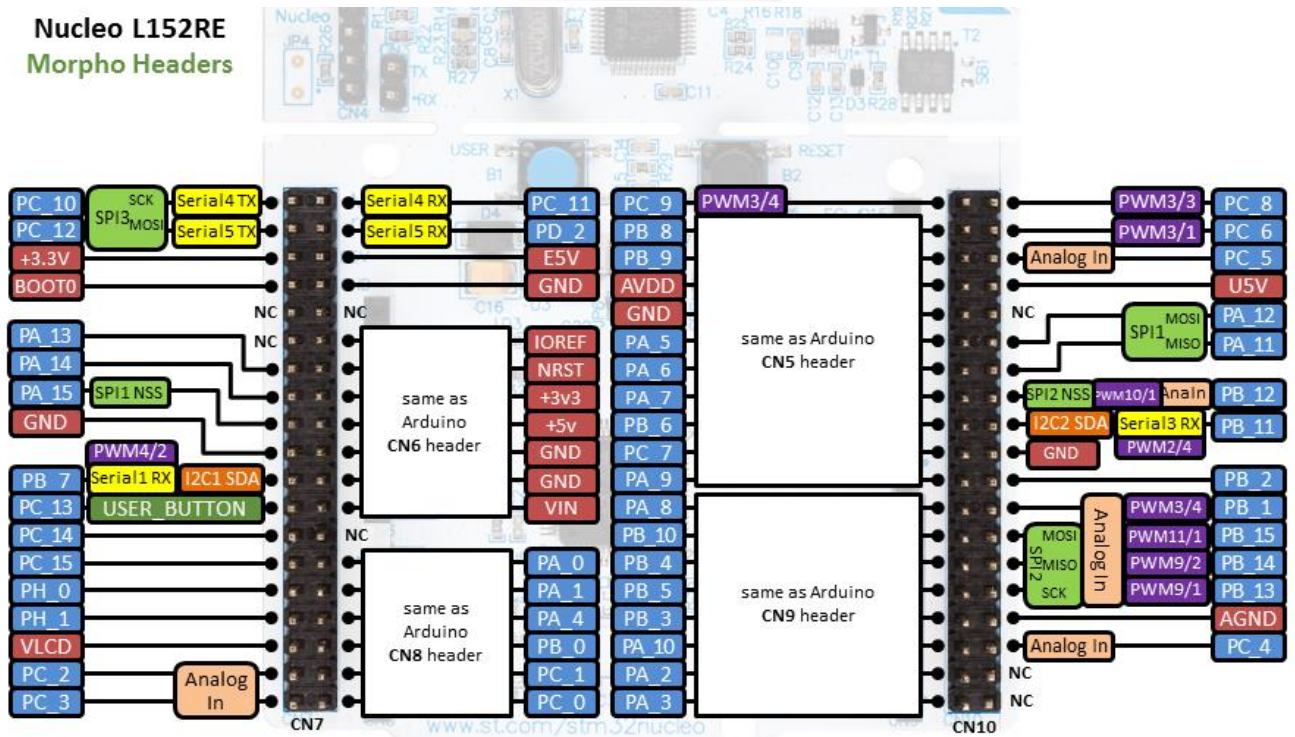


Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Nucleo L152RE
Arduino Headers



Nucleo L152RE
Morpho Headers



Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

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		Priorität	Vergleichs- und logische Operatoren	
++	Inkrement		Höchste	!
--	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

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

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:

```
// int toggle
for (int i=0; i<10; i++) {
    toggle = !toggle;
}
```

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 P_BUTTON gedrückt ist, wird der Ausgang invertiert

while (digitalRead(P_Button) == true) {
    ausgang = !ausgang;
    digitalWrite(P_OUT, 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 P_BUTTON gedrückt wird.
x = 100;
do {
    x--;
} while ((x>0) && (digitalRead(P_BUTTON) ==0));
```

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

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 P_BUTTON_1 gedrückt ist, sollen die Variablen ausgang1 eins und ausgang2
// null werden. Drückt man dagegen P_BUTTON_2, wird nur ausgang2 zu eins.
if ((digitalRead(P_BUTTON_1) ==1) {
    ausgang1 = 1;        // Block mit mehreren Anweisungen wird ausgeführt,
    ausgang2 = 0;        // wenn die Bedingung hinter if wahr ist
}
if ((digitalRead(P_BUTTON_2) ==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 P_BUTTON gedrückt ist, soll ausgang1 eins und ausgang2 null werden,
// andernfalls soll ausgang1 null und ausgang2 eins werden.
if ((digitalRead(P_BUTTON) ==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
}
```

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

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:                break;
    case 0x10:                break;
    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;
}
    
```

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

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
}
```

String-Klasse – Methodenübersicht

Methode	Beschreibung	Beispiel
charAt(index)	Gibt das Zeichen am angegebenen Index zurück.	char c = s.charAt(1); // bei "Hallo" → 'a'
compareTo(string)	Vergleicht zwei Strings lexikografisch.	s1.compareTo(s2); // < 0, == 0 oder > 0
concat(value)	Hängt einen Wert an den String an.	s.concat(" Welt");
c_str()	Gibt ein const char*-Array zurück.	Serial.println(s.c_str());
endsWith(suffix)	Prüft, ob der String mit dem Suffix endet.	s.endsWith(".txt");
equals(string)	Prüft auf exakte Übereinstimmung.	s.equals("Hallo");
equalsIgnoreCase(string)	Prüft auf Übereinstimmung (ohne Groß-/Kleinschreibung).	s.equalsIgnoreCase("hallo");
indexOf(char)	Gibt den Index des ersten Vorkommens zurück.	s.indexOf('l'); // bei "Hallo" → 2
lastIndexOf(char)	Gibt den Index des letzten Vorkommens zurück.	s.lastIndexOf('l'); // bei "Hallo" → 3
length()	Gibt die Länge des Strings zurück.	int len = s.length();
remove(index)	Entfernt alles ab dem Index.	s.remove(3); //"Hallo" → "Hal"
replace(find, replace)	Ersetzt ein Zeichen oder Substring.	s.replace("alt", "neu");
startsWith(prefix)	Prüft, ob String mit Präfix beginnt.	s.startsWith("Hal");
toCharArray(buf, len)	Kopiert Inhalt in char-Array.	s.toCharArray(buf, len);
toDouble()	Wandelt in double um.	double d = s.toDouble();
toFloat()	Wandelt in float um.	float f = s.toFloat();
toInt()	Wandelt in int um.	int n = s.toInt();

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Beispiel eines C/CPP-Programms

```
// Pin Definitionen
const int P_S1 = PA4;
const int P_LED_rt= D1, P_LED_ge= D2, P_LED_gn = D3;

volatile bool anforderungFussgaenger = false; //globale Variablen

char phasen[4]= { //array
    0b001, //0 rot
    0b101, //1 rotgelb
    0b010, //2 grün
    0b100}; //3 gelb

void setup()
{
    pinMode(P_LED_RT, OUTPUT);
    pinMode(P_LED_ge, OUTPUT);
    pinMode(P_LED_gn, OUTPUT);
    pinMode(P_S1, INPUT_PULLUP);
    z=0;
}

void loop()
{
    if (digitalRead(P_S1) == LOW)
    {
        anforderungFussgaenger = true;
        digitalWrite(P_LED_rt,HIGH);
        digitalWrite(P_LED_gn,LOW);
        digitalWrite(P_LED_ge,LOW);
        delay(500);
        digitalWrite(P_LED_ge,HIGH);
        delay(500);
        digitalWrite(P_LED_gn, HIGH);
        digitalWrite(P_LED_RT, LOW);
        digitalWrite(P_LED_ge, LOW);
        delay(500);
        digitalWrite(P_LED_gn, LOW);
        digitalWrite(P_LED_ge, HIGH);
        delay(500);
    }
    if (digitalRead(P_S1) == HIGH)
    {
        anforderungFussgaenger = false;
        for (int i=0; i<5; i++) {
            writeAmpelphasen(phasen[i]);
            delay(1000);
        }
    }
}
```

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

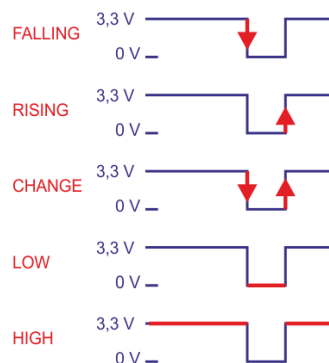
Portpin: Eingabe und Ausgabe

	Befehl	Beispiel
Portausgabe: Bsp.: <code>GPIOA->MODER = 0x5555; // Konfiguration Pin0..Pin7 als Output = 0b01010101010101; //Wobei 01 für ein Pin gilt GPIOA->ODR = 0x00FF; // LowByte wird auf HIGH gesetzt...</code>		
Einzelbitausgabe		
Deklaration	<code>#define / const int Name Pin</code> Name = Pin-Bezeichnung	<code>const int P_LED_b1 = D12;</code>
Konfiguration	<code>pinMode(P_LED_b1, OUTPUT);</code>	
Verwendung	<code>digitalWrite(name, WERT)</code> WERT=HIGH, LOW	<code>digitalWrite(P_LED_b1, HIGH);</code>
Porteingabe: Bsp.: <code>GPIOA->MODER = 0x0000; // Konfiguration Pin0..Pin7 als Input unsigned int portInA = GPIOA->IDR; // Portwert eingelesen...</code>		
Einzelbiteingabe		
Deklaration	<code>#define / const int Name Pin</code> Mögliche Werte für Pin: PA0..PA15, PB0..PB15, PC0..PC15 oder D1, A0...	<code>const int P_S2 = PA4;</code>
Konfiguration	<code>pinMode(name, konfig);</code> Mögliche Werte für konfig: = INPUT_PULLUP, INPUT_PULLDOWN, INPUT	<code>pinMode (P_S2, INPUT);</code>
Verwendung	Var = Pin-Zustand lesen	<code>buttonState = digitalRead(P_S2);</code>

Externer Interrupt

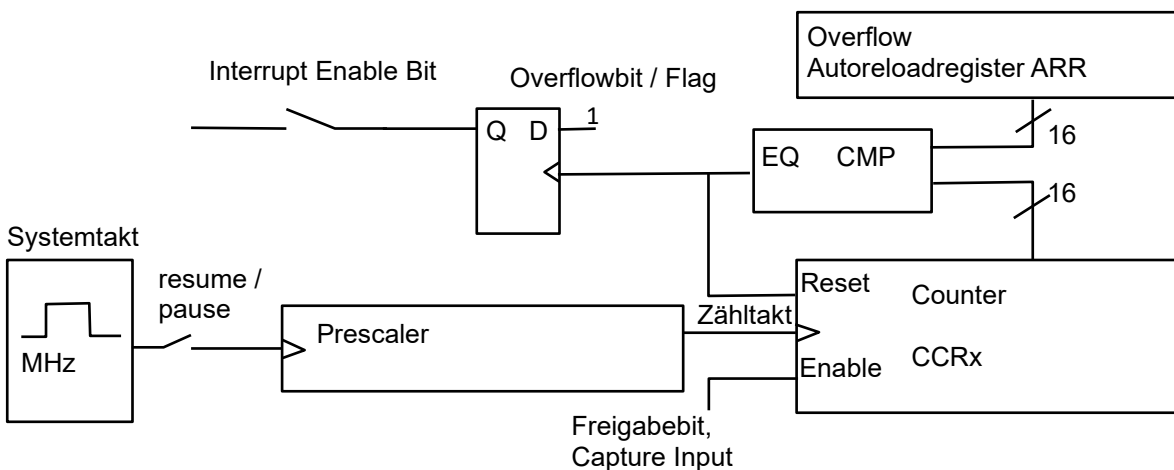
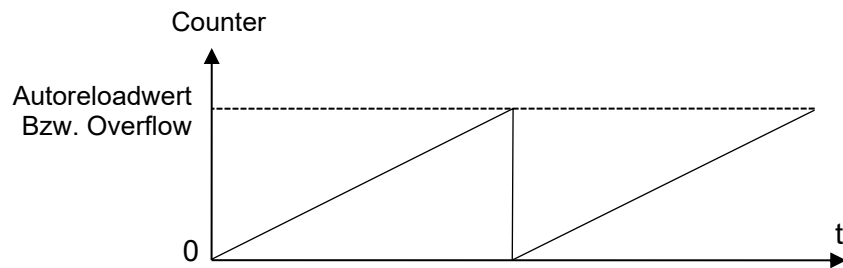


	Befehl	
Externer Interrupt		
Deklaration	<code>#define / const int Name Pin</code> Mögliche Werte für Portpin: PA0..PA15, PB0..PB15, PC0..PC15, oder D1, A0...	<code>const int P_S2 = PA4;</code>
Konfiguration	<code>attachInterrupt (digitalPinToInterrupt (PIN), ISR_Name, Aktion);</code>	
Bsp.:	<code>attachInterrupt (digitalPinToInterrupt (P_S2), ISR_EXT_IR, FALLING);</code>	
	Aktion: FALLING, RISING, CHANGE, HIGH, LOW	
	<code>detachInterrupt (digitalPinToInterrupt (P_S2))</code>	
Hinweis:	Variable(n) in der ISR sollten als <code>volatile</code> deklariert werden	



Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Timer
Prinzip:



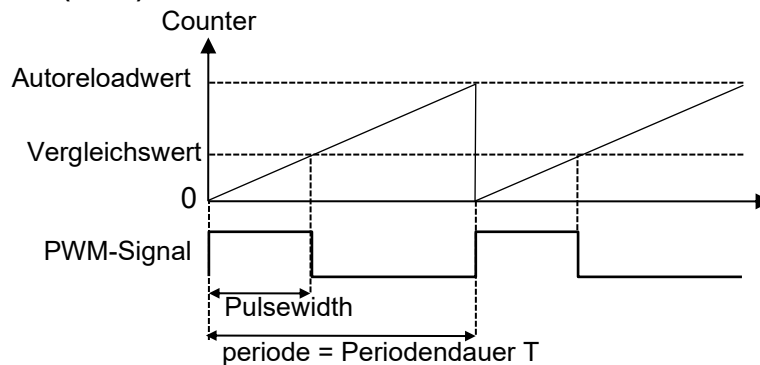
Beispiel STM32L152

Maßnahme	Syntax
Timerauswahl:	<code>static HardwareTimer mytimer = HardwareTimer(TIM4);</code>
Prescaler nutzen	<code>mytimer.setPrescaleFactor(32);</code> [≤ 16 Bit]
Prescaler lesen	<code>vorteilerWert = mytimer.getPrescaleFactor();</code>
Overflow nach f-Ctr _{ARR}	<code>mytimer.setOverflow(50000);</code> Prescaler Konfigurierbar!
Overflow lesen _{ARR}	<code>overflowWert = mytimer.getOverflow();</code>
TimerLR aktivieren und ISR	<code>mytimer.attachInterrupt(isr_Timer);</code>
TimerInterrupt deaktivieren	<code>mytimer.detachInterrupt();</code>
Timer starten	<code>mytimer.resume();</code>
Timer stoppen	<code>mytimer.pause();</code>
Timerzähler auslesen _{Counter}	<code>zaehlWert = mytimer.getCount();</code>
Timerzähler setzen _{Counter}	<code>mytimer.setCount(5);</code> [≤ 16 Bit]
Timer CaptureCompare CCRx-Register Hinweis=> mögliche Pins ermitteln, Bsp.: TIM2_CH1: PA0, TIM3_CH3: PB0	<code>mytimer.setMode(1, TIMER_INPUT_CAPTURE_RISING, CCPin);</code> //Mode: (Kanal 1, ...RISING FALLING BOTHEDGE, GPIO) <code>mytimer.attachInterrupt(1, isr_...);</code> //Kanal 1, isr_... <code>captureWert = mytimer.getCaptureCompare(1);</code>
Hinweis:	Variable(n) in der ISR sollten als volatile deklariert werden

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Puls-Weiten-Modulation (PWM)

Prinzip:



PWM	Befehl	Beispiel
Deklaration	#define const int Name Pin	const int P_RGB_b1 = 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(P_RGB_b1, 200);
Berechnung Tastgrad: Pulsewidth / Periodendauer [in %]		

LCD

```
void initLCD(void){
    lcd2x16.begin(16, 2);    // 16 Zeichen, 2 Zeilen
    lcd2x16.clear();        // löschen
    lcd2x16.setBacklight(255); // Hintergrundlicht
    lcd2x16.setCursor(0,0); // Cursorposition (Spalte, Zeile)
    lcd2x16.print("eScooter..."); //Textausgabe}
```

Ausgabevariante am LCD mit zusammengesetzten Strings, Bsp.:

```
lcdWrite(("Temp:" + String(var) + "C"), "Feuchte:" + String(var));
```

```
void lcdWrite(String z1, String z2)
{ lcd2x16.setCursor(0, 0);
  lcd2x16.print(z1);
  lcd2x16.setCursor(0, 1);
  lcd2x16.print(z2); }
```

Ausgabevariante am LCD mit Standard-Funktion sprintf();, Bsp:

```
char buf [16];
sprintf(buf, "%02u : %02u : %02u", std, min, sec);
lcd2x16.print(buf);
```



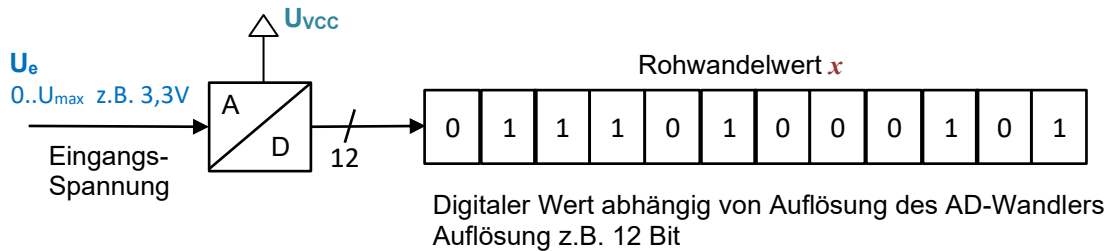
Bei Umlauten und Sonderzeichen müssen diese jeweils im Displaycode eingefügt werden:
Bsp.: Würfel\xe1hler

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

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Analog – Digital – Wandlung

AD-Wandler	Befehl	Beispiel
Deklaration	Datentyp Name Pin	const int P_A0 = A0;
	analogReadResolution(8, 10, 12); //Bitbreite	analogReadResolution(12);
	Mögliche Werte für Portpin: A0-A5	
Verwendung	var = analogRead(Pin)	sensorValue = analogRead(P_A0);



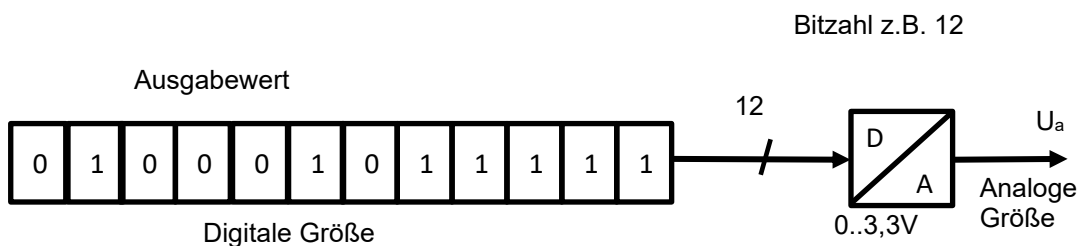
Berechnungsformeln:

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

Werte Anpassung / Skalierung mit der map() Funktion:
Allgemeine Formel: $map(x, vonMin, vonMax, nachMin, nachMax) = \frac{(x - vonMin) \cdot (nachMax - nachMin)}{vonMax - vonMin} + nachMin$
Operation Arduino Framework (Bsp.): <code>map(x, 0, 1023, 0, 255); // Werte von 0-1023 werden auf 0-255 angepasst</code>

Digital – Analog – Wandlung

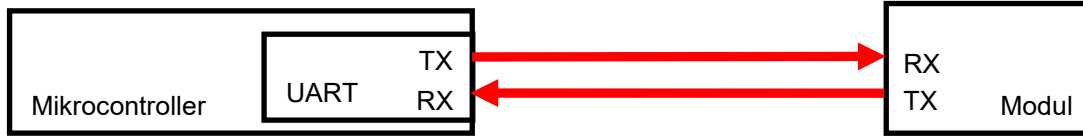
	Befehl	Beispiel
Digital-Analog Wandler		
Deklaration	Datentyp Name Pin	const int P_DAC = D13;
	AnalogWriteResolution()	AnalogWriteResolution(12);
	Mögliche Werte für Portpin: 0-4095	
Verwendung	analogWrite(Pin, Wert)	analogWrite(P_DAC, outputValue);



Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

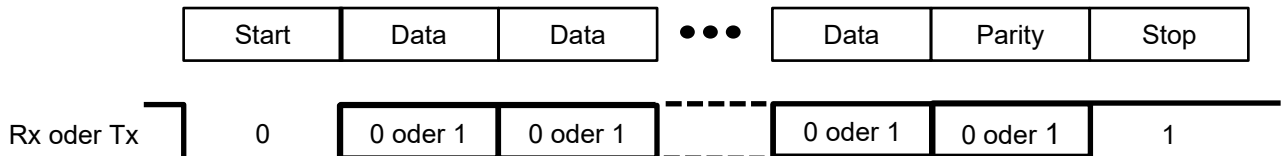
Externe Kommunikationsmöglichkeiten

Universal Asynchronous Receiver Transmitter (UART)



	Befehl	Beispiel
Universal Asynchronous Receiver Transmitter (UART)		
Deklaration	<pre>const int P_RX = Pin?; const int P_TX = Pin?;</pre>	
Verwendung	<pre>Serial.begin(115200); // Baudrate</pre>	
Daten empfangen	Fragen nach daten im Serial-Buffer Wenn ja, dann in Variable lesen...	<pre>if (Serial.available()) { incomingByte = Serial.read(); } </pre>
Daten senden	Mit print(ln) String schreiben + Zeilenumbruch Mit print Varwert schreiben	<pre>Serial.println("LED an"); Serial.print(LED);</pre>

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:

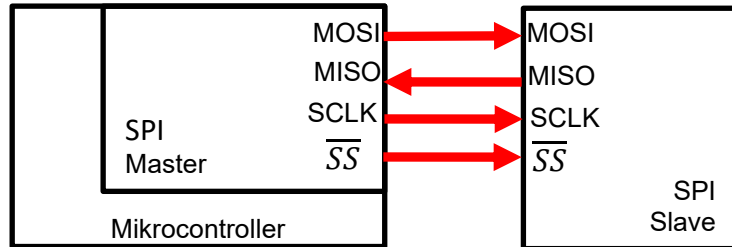
Informationswort	Summe der Einsen	Paritätsbit / Codewort	
		bei Even-Parity	bei Odd-Parity
0011.1010	gerade	0 / 0011.1010 0	1 / 0011.1010 1
1010.0100	ungerade	1 / 1010.0100 1	0 / 1010.0100 0

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Serial Peripheral Interface (SPI)

Das **Serial Peripheral Interface (SPI)** dient der Kommunikation des Mikrocontrollers mit **Modulen** auf der Platine. Ideal für schnelle Datenübertragung über kurze Distanzen. 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)

Empfangsleitung
 Sendeleitung
 Auswahl Slaves/Chip (Lowaktiv)

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
	<pre>#include <SPI.h> //Bibliothek einbinden Für SPI_1 gelten folgende Pins als Standard: SPI.setMOSI(D11); SPI.setMISO(D12); SPI.setSCK(D13);</pre>	
Initialisiere SPI	<pre>SPI.begin(); SPISettings(8000000, MSBFIRST, SPI_MODE0); //f, dataOrder, Modus SPI_MODE0)); SPI.setBitOrder(MSBFIRST); SPI.setClockDivider(SPI_CLOCK_DIV32); SPI.setDataMode(SPI_MODE0); SPI.end(); //NUR bei Beendigung des SPI-Busses</pre>	
Konfiguration CS-Pin	<pre>const int P_CS = D5?; // chipSelectPin pinMode(CS, OUTPUT);</pre>	
Verwendung	<pre>Bsp: Schreiben: digitalWrite(P_CS, LOW); SPI.transfer(0x40); SPI.transfer(0x09); SPI.transfer(Bitmuster); digitalWrite(P_CS, HIGH);</pre>	<pre>Bsp.: Lesen: digitalWrite(P_CS, LOW); ADC_H = SPI.transfer(0x00); ADC_L = SPI.transfer(0x01); digitalWrite(P_CS, HIGH);</pre>

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

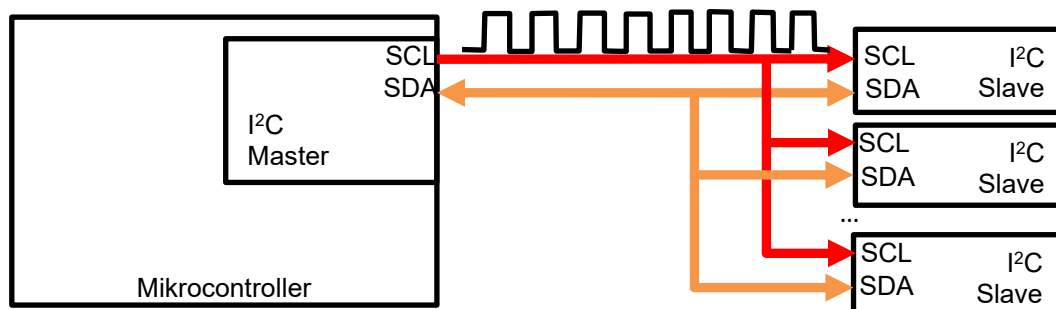
Inter-Integrated Circuit (I²C)

Das Inter-Integrated Circuit (I²C) ist ein serielles Kommunikationsprotokoll dient der Kommunikation mit Modulen auf der Platine. Für mittlere Distanzen und Systeme mit vielen Geräten. Module sind:

- Sensoren (Temperatur, Druck, Beschleunigung)
- Echtzeituhr-Module
- LCD-Displays
- Digital / Analog-Wandler

Signale

SCL (Serial Clock): Taktleitung und SDA (Serial Data): Datenleitung



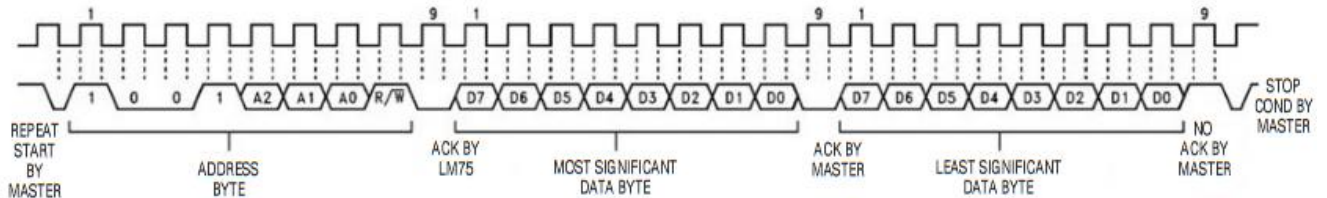
I ² C	Befehl	Beispiel
	<code>#include <Wire.h> //=> I²C-Library</code>	JE NACH Datenblatt!
Deklaration	<code>#define address => 7 Bit!</code>	<code>address 0b0100000</code>
Pindeklaration	<code>#define SCL PB8 //I²C1 => I²C2: PB10</code> <code>#define SDA PB9 //I²C1 => I²C2: PB11</code>	<code>Wire.setSCL(PB8);</code> <code>Wire.setSDA(PB9);</code>
Initialisierung	Bibliothek starten I ² C-Frequenz einstellen	<code>Wire.begin();</code> <code>Wire.setClock(10000);</code>
Verwendung	I ² C-Übertragung starten inkl. Adresse Baustein	<code>Wire.beginTransmission(address);</code>
Daten senden	Daten auf Bus schreiben Übertragung abschließen	<code>Wire.write(0x09);</code> <code>Wire.endTransmission();</code>
Daten empfangen	I ² C-Übertragung starten inkl. Adresse Baustein Daten auf Bus schreiben	<code>Wire.beginTransmission(address);</code> <code>Wire.write(0x00);</code> <code>Wire.endTransmission();</code>
	Nach Empfangenen Daten fragen... Var = daten lesen	<code>Wire.requestFrom(address,WR_Byte);</code> <code>Temp_H = Wire.read();</code> <code>Temp_L= Wire.read();</code>
	Übertragung beenden	<code>Wire.endTransmission();</code>
	Daten aus einem bestimmten Register lesen: Slave Registeradresse mitteilen Inhalt des Registers in var lesen	<code>Wire.beginTransmission(address);</code> <code>int WR_Byte = Wire.write(0x??);</code> <code>Wire.endTransmission();</code> <code>Wire.requestFrom(address, WR_Byte);</code> <code>var = Wire.read();</code> <code>Wire.endTransmission();</code>
ReStart gemäß Datenblatt	Wenn Restart gemäß I ² C-Frame benötigt:	<code>Wire.endTransmission();</code>

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

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

LM 75:

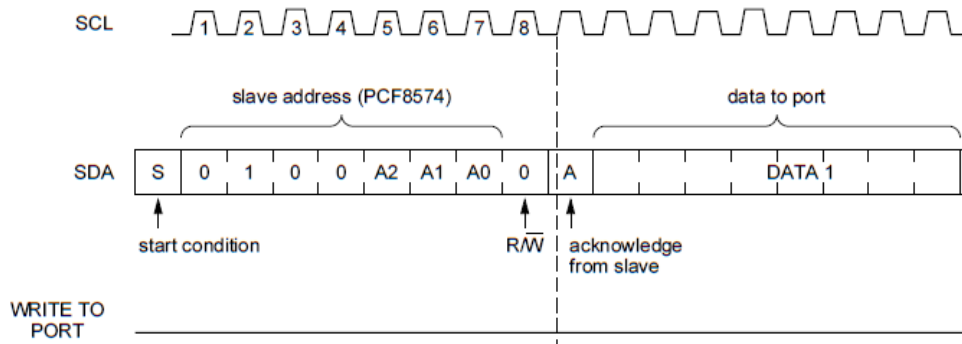
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
1	0	0	1	A2	A1	A0	RD/W



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:



R/W : Write=0, Read=1

Abiturprüfung ab 2024	Berufliches Gymnasium (TG)
Formelsammlung	1.5.2 Informationstechnik

Allgemeine Coding Empfehlung/Vorschlag für HW-Programmierung C/C++

Namenskonventionen

Verwenden Sie aussagekräftige Namen: Variablen, Funktionen und Konstanten sollten klar benennen, wofür sie stehen.

- Schlecht: `int x;`
- Gut: `int sensorValue;`

In C++ setzt man normalerweise den lower-CamelCase ein. CamelCase ist eine Namenskonvention, bei der Wörter oder Wortbestandteile ohne Leerzeichen oder Trennzeichen zusammengeschrieben werden, wobei der erste Buchstabe jedes Wortes (außer dem ersten Wort) großgeschrieben wird. Beispiel: `sensorValue`, `readTemperature`, `myVariable`

Bezeichnung Routinen	immer klein – lower Camelcase Bsp: <code>void readTemperature(); // Verb am Anfang</code>
Bezeichnung Interrupt Service Routinen	isr am Anfang. Sonst Camelcase. Bsp.: <code>void isr_ModPlus()</code>
Bezeichnung von Konstanten Konstanten sollten in Großbuchstaben mit Unterstrichen zwischen den Wörtern geschrieben werden	<code>const int MAX_TEMPERATURE = 90;</code> oder (Alternativ): <code>#define MAX_TEMPERATURE 100</code>
Zustände / Enumeratoren sollten in <u>Großbuchstaben</u> mit Unterstrichen zwischen den Wörtern geschrieben werden	<code>enum zustandstyp {BA1, BA2, BA3};</code> Vorteil: lässt sich in- bzw. dekrementieren <code>enum zustandstyp aktZustand = BA1;</code> oder (Alternativ): <code>#define RASENMAEHER_AUS 0</code> <code>#define RASENMAEHER_LINKS 2</code> <code>#define RASENMAEHER_VORWAERTS 1</code> <code>#define RASENMAEHER_RECHTS 3</code>
Bezeichnung von Klassen	immer groß und Einzahl Bsp.: <code>HardwareTimer</code>
Bezeichnung Objekten	Bsp.: <code>HardwareTimer timer3 = HardwareTimer(TIM3);</code> <code>LiquidCrystal_PCF8574 lcd2x16 (0x27);</code>
Pin Configuration	// Pin-Konfiguration, P steht für Pin. Bsp.: <code>//Der blaue Taster ist mit PC13 verbunden:</code> <code>const int P_BUTTON = PC13;</code> <code>//Die LED ist mit PA5 verbunden (On-Board LED):</code> <code>const int P_LED = PA5;</code> <code>//Nicht empfohlen, da nicht Debugfähig & mbed nicht umsetzbar:</code> <code>#define P_S1_TASTER D3</code>

! Keine Bewertungsgrundlage !