# MICROCONTROLLER PROGRAMMIERUNG

- Eine kurze Anleitung -

## Bit Arithmetik

**Logisches UND "&":** Das logische & setzt nur Bits, die rechts- UND linksseitig logisch `1` sind.
`0101 & 0011` → `0001`

**Logisches ODER "|":** Das logische Oder setzt alle Bits die links- und/oder rechtsseitig gesetzt sind.
`0101 | 0011` → `0111`

**Invertierung "∼":** Die Tilde (∼) negiert das folgende Argument.
∼`0101` → `1010`

**Das XOR "^":** Die XOR Funktion setzt alle Bits, die nur links- ODER rechtsseitig gesetzt sind. Beidseitig gleich gesetzte Bits werden zu `0`.
`0101^0011` → `0110`

**Der Shift-Operator ">>" und "<<":** Mit dem Shift Operator lassen sich Bits nach Links oder Rechts verschieben. Das Argument rechtsseitig des Operators gibt dabei an, wie oft geschoben werden soll. Überstehende Bits fallen dabei weg und neue Bits werden auf `0` gesetzt.
`0101<<1` → `1010`
`0101>>2` → `0001`

## Operatoren und Variablen

Es ist auch möglich, alle Operatoren auf eine Variable oder Port anzuwenden. Die Instruktion "`PORTA>>=2`" verschiebt alle Bits des Port A um 2 Stellen nach rechts.
Die Instruktion "`PORTA| =4`" setzt das 3. Bit von rechts auf `1`.

## Das Byte

Ein Byte besteht aus 8 Bits:

| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|

## Hexadezimalschreibweise

In C ist es nicht möglich, direkt Bit-Variablen zu setzen. Daher ist es nötig, diese entweder im Dezimalsystem oder Hexadezimalsystem anzugeben. Das Hexadezimalsystem eignet sich besonders gut, da man relativ einfach umrechnen kann.

| Dez. | Hex | Bit | Dez. | Hex | Bit |
|---|---|---|---|---|---|
| 0 | 0 | 0000 | 8 | 8 | 1000 |
| 1 | 1 | 0001 | 9 | 9 | 1001 |
| 2 | 2 | 0010 | 10 | A | 1010 |
| 3 | 3 | 0011 | 11 | B | 1011 |
| 4 | 4 | 0100 | 12 | C | 1100 |
| 5 | 5 | 0101 | 13 | D | 1101 |
| 6 | 6 | 0110 | 14 | E | 1110 |
| 7 | 7 | 0111 | 15 | F | 1111 |

## Beispiele

| | |
|---|---|
| `0xA5` ↔ `1010 0101` | `0x17` ↔ `0001 0111` |
| `0x00` ↔ `0000 0000` | `0xFF` ↔ `1111 1111` |
| `0x0F` ↔ `0000 1111` | `0x64` ↔ `0110 0100` |

## Ports & Pins

Als ein *Port* werden immer 8 (oder weniger) Pinne zusammengefasst. Bei AVR Controllern sind diese "`PORTA, PORTB`" usw. benannt.

- Pin A1 auf Ausgang schalten:
  `DDRA |= _BV(PA1);`
- Pullup an Pin D0 anschalten:
  `PORTD |= _BV(PD0)`
  *Wichtig: Der Pullup kann erst eingeschaltet werden, NACHDEM der Pin auf Eingang geschaltet wurde*
- Pin A4 auslesen:
  `if (PINA & _BV(PA4)) { ...`
- Pin A3 auf high schalten:
  `PORTA |= _BV(PA3)`

- Pin A3 auf low schalten:
  `PORTA &= ~(_BV(PA3))`
- Pin A3 umschalten (toggle):
  `PORTA ^= _BV(PA3)`

### Interrupts

Interrupts unterbrechen den Fluss des Hauptprogramms. Sie können durch verschiedenste Quellen ausgelöst werden (z.B. Timer oder Signal am Interrupt Pin). Um einen Interrupt zu benutzen muss man in der Regel folgende Dinge tun:

1. Interrupt Routine definieren und mit Code füllen:
   `ISR(...) { ... }`
2. Entsprechende Bits in entsprechenden Registern setzen (→ siehe Datenblatt). z.B. für den ext. Interrupt 0:
   `MCUCR |= _BV(ISC01);`
   `GICR |= _BV(INT0);`
   Meist muss man hier 2 oder 3 Register setzen.
3. Interrupts anschalten: `sei();`

### Toolchain benutzen

1. Einstellungen für Deinen Programmer setzen → `Makefile` editieren
2. Compilieren mit dem Befehl `make`.
3. Controller flashen (entweder mit avrdude oder `make flash` (je nach Makefile))

### Fuse Bits

Fuse bits setzen die wichtigsten Einstellung des Controllers - z.B. mit welchem Takt er laufen soll und aus welcher Quelle er seinen Takt beziehen soll (intern, externer Quartz, ...).
Wie die Fuse bits zu setzen sind, ist dem Datenblatt zu entnehmen. Vorsicht: *Falsche Einstellungen können den Controller unbrauchbar machen.*

### Fuse Bits mit avrdude setzen

```
avrdude -p m32 -c usbasp \
-U lfuse:w:0x9F:m -U hfuse:w:0xC9:m
```

### Checkliste mit den häuftigsten Fehlern

- *Nutzereingaben funktionieren nicht wie gewünscht.* → Taster entprellt? Pullup an? Liest Du auch wirklich das `PIN` Register (und nicht `PORT`) aus?
- *Programm läuft langsam / Ausgänge verhalten sich komisch.* → Fuses richtig gesetzt?
- *Der ADC verhält sich komisch / misst nicht richtig / "zappelt".* → Kondensator an AVCC geschaltet?
- *Der Timer funktioniert nicht.* → Interrupts eingeschaltet (`sei()`)? Bit im `TIMSK` Register gesetzt?

### Wichtige Links

**Laborboard Dokumentation**:
`https://das-labor.org/wiki/Laborboard`

**Laborboard Beispielcode**:
`https://www.das-labor.org/trac/browser/microcontroller/src-atmel/tests/helloboard`