

# Angewandte Informatik

Prof. Dr. rer. nat. Peter Gerwinski

17. Dezember 2015

# Angewandte Informatik

- 1 Einführung
- 2 Einführung in C
- 3 Bibliotheken
- 4 Algorithmen
  - 4.1 Differentialgleichungen
  - 4.2 Rekursion
  - 4.3 Stack und FIFO
  - 4.4 Aufwandsabschätzungen
- 5 Hardwarenahe Programmierung
  - 5.1 Bit-Operationen
  - 5.2 I/O-Ports
  - 5.3 Interrupts
  - 5.4 volatile-Variable
  - 5.5 Software-Interrupts

...

...

# 5 Hardwarenahe Programmierung

## 5.1 Bit-Operationen

### 5.1.1 Zahlensysteme

Oktal- und Hexadezimal-Zahlen lassen sich ziffernweise  
in Binär-Zahlen umrechnen:

000	0	0000	0	1000	8
001	1	0001	1	1001	9
010	2	0010	2	1010	A
011	3	0011	3	1011	B
100	4	0100	4	1100	C
101	5	0101	5	1101	D
110	6	0110	6	1110	E
111	7	0111	7	1111	F

**Auswendig lernen!**

## 5.1.2 Bit-Operationen in C

C-Operator	Verknüpfung	Anwendung
&	Und	Bits gezielt löschen
	Oder	Bits gezielt setzen
^	Exklusiv-Oder	Bits gezielt invertieren
~	Nicht	Alle Bits invertieren
<<	Verschiebung nach links	Maske generieren
>>	Verschiebung nach rechts	Bits isolieren

Beispiele:

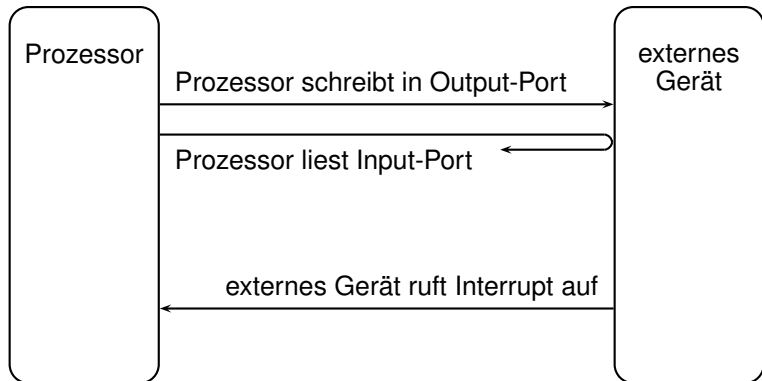
- Bit Nr. 5 gezielt auf 1 setzen: `PORTB |= 1 << 5;`
- Bit Nr. 6 gezielt auf 0 setzen: `PORTA &= ~(1 << 6);`
- Ist Bit Nr. 4 gesetzt? `if (PINC & (1 << 4) ...`
- Umschalten zwischen Ein- und Ausgabe: `DDR`  
Bit = 1: Ausgabe; Bit = 0: Eingabe

**Details abhängig von  
Prozessor und Compiler!**

## 5.2 I/O-Ports

## 5.3 Interrupts

Kommunikation mit externen Geräten



## 5.2 I/O-Ports

In Output-Port schreiben = Leitungen ansteuern

Datei: `RP6Base/RP6Base_Examples/RP6Examples_20080915/  
RP6Lib/RP6base/RP6RobotBaseLib.c`

Suchbegriff: `setMotorDir`

```
void setMotorDir(uint8_t left_dir, uint8_t right_dir)
```

```
{
```

```
    /* ... */
```

```
    if(left_dir)
```

```
        PORTC |= DIR_L;
```

```
    else
```

```
        PORTC &= ~DIR_L;
```

```
    if(right_dir)
```

```
        PORTC |= DIR_R;
```

```
    else
```

```
        PORTC &= ~DIR_R;
```

```
}
```

Manipulation einzelner Bits

→ Steuerung der Motordrehrichtung

Output-Port

## 5.2 I/O-Ports

In Output-Port schreiben = Leitungen ansteuern

Datei: `RP6Base/RP6Base_Examples/RP6Examples_20080915/  
RP6Lib/RP6base/RP6RobotBaseLib.c`

Suchbegriff: `updateStatusLEDs`

```
DDRB &= ~0x83;    ← Data Direction Register: auf Input(!) setzen
PORTB &= ~0x83;   ← internen Pull-Up-Widerstand ausschalten
if(statusLEDs.LED4){ DDRB |= SL4; PORTB |= SL4; }
if(statusLEDs.LED5){ DDRB |= SL5; PORTB |= SL5; }
if(statusLEDs.LED6){ DDRB |= SL6; PORTB |= SL6; }
DDRC &= ~0x70;
PORTC &= ~0x70;
DDRC |= ((statusLEDs.byte << 4) & 0x70);
PORTC |= ((statusLEDs.byte << 4) & 0x70);
```

Manipulation einzelner Bits

3 Bits gemeinsam

**union** statusLEDs: Bit-Felder vs. Byte

## 5.3 Interrupts

Externes Gerät ruft (per Stromsignal) Unterprogramm auf  
Zeiger hinterlegen: „Interrupt-Vektor“

Datei: [RP6Base/RP6Base\\_Examples/RP6Examples\\_20080915/  
RP6Lib/RP6base/RP6RobotBaseLib.c](#)

Suchbegriff: ISR

„Dies ist ein Interrupt-Handler.“

Interrupt-Vektor 0 darauf zeigen lassen

Schreibweise abhängig von Prozessor und Compiler!

```
ISR (INT0_vect)
{
    mleft_dist++;
    mleft_counter++;
    /* ... */
}
```

Aufruf durch Sensor an Encoder-Scheibe  
→ Entfernungsmessung



## 5.4 volatile-Variable

```
volatile uint16_t mleft_counter;
```

```
/* ... */
```

„Immer lesen und schreiben. Nicht wegoptimieren.“

```
volatile uint16_t mleft_dist;
```

```
/* ... */
```

```
ISR (INT0_vect)
```

```
{  
    mleft_dist++;  
    mleft_counter++;  
    /* ... */  
}
```

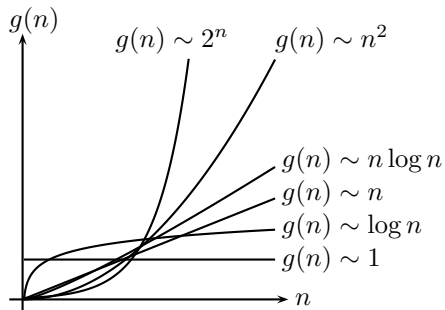
```
int main (void)
```

```
{  
    int prev_mleft_dist = mleft_dist;  
    while (mleft_dist == prev_mleft_dist)  
        /* just wait */;  
}
```

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen



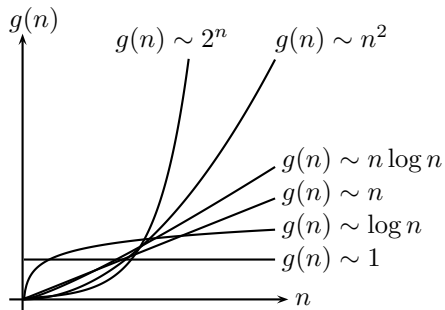
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$



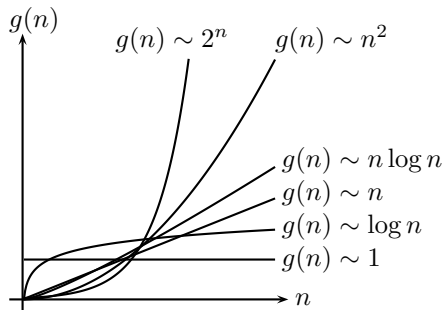
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort



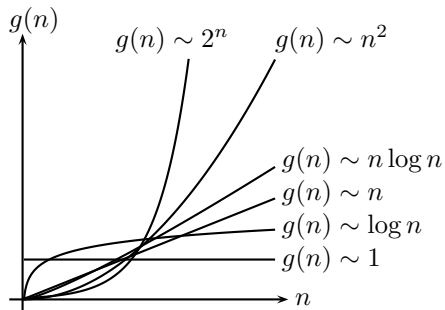
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$



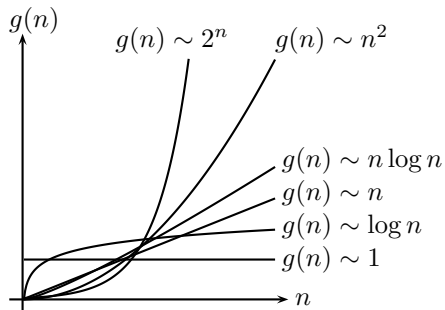
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen,  
abbrechen, falls schon sortiert  
→ Bubblesort



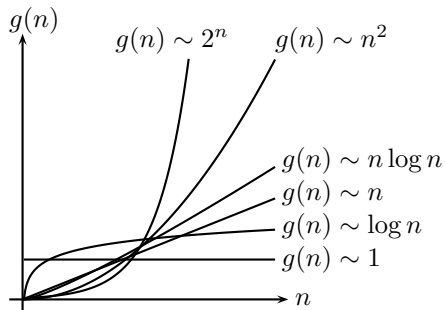
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen, abbrechen, falls schon sortiert  
→ Bubblesort:  $\mathcal{O}(n)$  bis  $\mathcal{O}(n^2)$



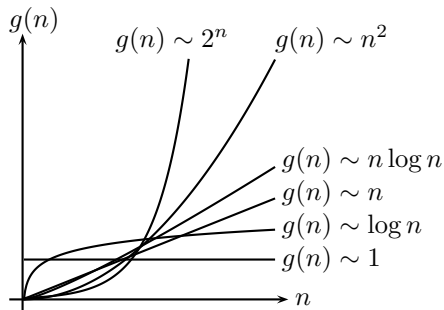
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen, abbrechen, falls schon sortiert  
→ Bubblesort:  $\mathcal{O}(n)$  bis  $\mathcal{O}(n^2)$
- Rekursiv sortieren  
→ Quicksort



$n$ : Eingabedaten

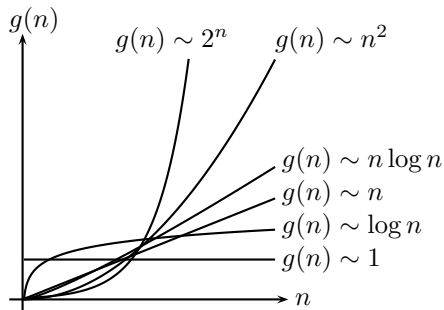
$g(n)$ : Rechenzeit



## 5.5 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen, abbrechen, falls schon sortiert  
→ Bubblesort:  $\mathcal{O}(n)$  bis  $\mathcal{O}(n^2)$
- Rekursiv sortieren  
→ Quicksort:  $\mathcal{O}(n \log n)$  bis  $\mathcal{O}(n^2)$



$n$ : Eingabedaten

$g(n)$ : Rechenzeit

# Angewandte Informatik

## 1 Einführung

## 2 Einführung in C

## 3 Bibliotheken

## 4 Algorithmen

4.1 Differentialgleichungen

4.2 Rekursion

4.3 Stack und FIFO

4.4 Aufwandsabschätzungen

## 5 Hardwarenahe Programmierung

5.1 Bit-Operationen

5.2 I/O-Ports

5.3 Interrupts

5.4 volatile-Variable

5.5 Software-Interrupts

...

...