

U3 Grundlagen der AVR-Programmierung

- Makros
- Bitoperationen

U3-1 Makros

- Makros sind Textersetzungen, welche vom Präprozessor aufgelöst werden. Dies Passiert bevor der Compiler die Dateien verarbeitet.
- Aufbau: `#define Suchwort Ersetzung`
 - ◆ Anweisungsende ist der Zeilenumbruch (kein Strichpunkt!)
- Ersetzung:

```
#define MEINE_KONST 7
[...]  
a = b + MEINE_KONST; // a = b + 7
```

```
#define MEINE_ERSETZUNG = b + 7
[...]  
a MEINE_ERSETZUNG; // a = b + 7
```

U3-1 Makros (2)

■ Funktionen:

```
#define POW2(a) (a * a)
[...]  
a = POW2(4); // a = (4 * 4);
```

■ Achtung:

```
#define SUB(a, b) a - b
[...]  
a = SUB(7, 5 + 2) * 5; // a = 7 - 5 + 2 * 5 = 12
```

◆ Berechnungen bei Makros in Klammern setzen

```
#define SUB(a, b) ((a) - (b))
[...]  
a = SUB(7, 5 + 2) * 5; // a = ((7 - (5 + 2)) * 5 = 0
```

U3-2 Logische Operatoren

■ Logische Operatoren:

"nicht"		"und"		"oder"	
!		&&	f w		f w
f	w	f	f f	f	f w
w	f	w	f w	w	w w

U3-2 Bit-Operatoren

■ Bit-Operatoren

"nicht"	"und"	"oder"	"x-oder"
~	&		^
f w	f f f	f f w	f f w
w f	w f w	w w w	w w f

■ Beispiel:

	1100	1100	1100
~	&		^
1001	1001	1001	1001
0110	1000	1101	0101

U3-2 Shift-Operatoren

➡ Bits werden im Wort verschoben

<<	Links-Shift
>>	Rechts-Shift

■ Beispiel:

```
uint8_t x = 0x9d;
```

x	1	0	0	1	1	1	0	1
x <<= 2	0	1	1	1	0	1	0	0
x >>= 2	0	0	0	1	1	1	0	1

■ Achtung! Bei signed-Variablen ist das >>-Verhalten meist nicht 100% definiert. Im Normalfall(!) werden bei negativen Werten 1er geschiftet.