Übersicht zur AVR – C Programmierung

Schlüsselworte und Bezeichner (reservierte Worte):

auto	**double**	**int**	**struct**	**break**	**else**	**long**
switch	**case**	**enum**	**register**	**typdef**	**char**	**extern**
return	**union**	**const**	**float**	**short**	**unsigned**	**continue**
for	**signed**	**void**	**default**	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**			

wichtige Binäre arithmetische Operatoren (zwei Operanten nötig)
`+` Addition, `-` Subtraktion, `*` Multiplikation, `/` Division, `%` Modulo

Unäre arithmetische Operatoren (nur ein Operant nötig)
`+,-` Vorzeichen, `++` Inkrement, `--` Dekrement

Bit-Operatoren
& and, **|** or, **~** not, **^** xor, **>>** shift right, **<<** shift left

Vergleichsoperatoren

==	ist gleich	**>**	größer als	**<**	kleiner als
>=	größer gleich	**<=**	kleiner gleich	**!=**	ungleich

Verknüpfungsoperatoren
&& 2 Bedingungen ver-*und*-en **||** 2 Bedingungen ver-*oder*-n **!** Bedingung verneinen

Trennzeichen

;	Befehlsende	**,**	Trennzeichen Parameter
{	Blockanfang Geltungsbereich	**}**	Blockende Geltungsbereich
(	Anfang Parameterliste	**)**	Ende Parameterliste

Zeichenketten
“Hallo“ konstanter String, **’A’** konstantes Zeichen

Zahlen
123 Ganzzahl Dezimal-Darstellung **1.23** Gleitkommazahl
0xA0 Ganzzahl Hexa-Darstellung **0b001** Ganzzahl Binär-Darstellung

Kommentare

//	bis Zeilenende	*/**	Komentaranfang	**/*	Kommentarende

Codebeispiele / Codestruktur

```
#include <avr\io.h>
// ----------------------------------------------------------------------
char function1 (int para1) // Funktion mit Parameter und Rückgabewert
{
    // hier Unterprogrammcode
    return ...;
}
// ----------------------------------------------------------------------
ISR (INT0_vect) // Interrupt Service Routine
{
    // hier Interruptbehandlung
}
// ----------------------------------------------------------------------
int main (void) // Hauptprogramm, startet bei Reset
{
    // hier Initalisierungen
    while (1)
    {
       // hier Verarbeitungscode
    }
}
// ----------------------------------------------------------------------
```

Die wichtigsten Datentypen in AVR C:

char Datentyp für eine Zeichen (Character, Buchstabe). 8 Bit
Beispiel: char buchstabe = 'K';

int Datentyp für eine Ganzzahl (Integer). 16 Bit
Beispiel: int alter = 37;

float Datentyp für eine Kommazahl (Komma=Punkt). 32 Bit, Wertebereich: $3.4*10^{-38}...3.4*10^{38}$
Beispiel: float alter = 37.5;

unsigned vorzeichenloser Datentyp (nur positive Zahlen)
signed vorzeichenbehafteter Datentyp

Beispiele:

```
signed char wert1;    // 8 Bit, Wertebereich: -128...127
unsigned char wert2; // 8 Bit, Wertebereich: 0...255
signed int  wert3;    // 16 Bit, Wertebereich: -32768...+32767
unsigned int wert4;  // 16 Bit, Wertebereich: 0...+65535
```

volatile (flüchtig) legt fest das der Wert einer Variablen durch nebenläufige Ereignisse außerhalb der Funktion verändert werden kann (z.B. globale Daten werden durch ISR verändert) **Beispiel:**

```
volatile int  wert5; // 16 Bit, von Optimierung ausgeschlossen
```

Die wichtigsten Kontrollstrukturen in AVR C:

Anweisung; (wird immer mit einem Semikolon abgeschlossen)

```
PORTB=0xFF;
```

if (Bedingung) Anweisung; **if** (Bedingung) Anweisung; **else** Anweisung;

```
if (!(PIND&0x04)) PORTB=0xFF;
if (!(PIND&0x04)) PORTB=0xFF; else PORTB=0x00;
```

while (Bedingung) Anweisung ; /* kopfgesteuerte Schleife */

```
while (i<5000) i++ ;          // zählen
while (i<5000) { ...    }     // mehr als eine Anweisung wiederholt ausführen
```

do {Anweisung} **while** (Bedingung); /* fußgesteuerte Schleife */

```
do i++; while (i<500); // zählen
do {...} while (1); // mehr als eine Anweisung wiederholt ausführen
```

for (Initialisierung; Wiederholbedingung; Schrittweite) Anweisung; /* Zählschleife */

```
for (int i = 0; i < 10; i++);
for (int i = 10; i > 0; i--) { ... }
```

switch (switch_variable) /* Fallunterscheidung */

```
{
 case Konstante1: {Anweisungs1;} break;
 case Konstante2: {Anweisungs2;} break;
 default : {Anweisung_X;} break;
}
```

```
switch ( PIND & 0x0C )      // 2 Taster an PortD 2 und 3
       {
         case 0x08 : PORTB=0x01 ; // Taster1 -> LED1
                     break;
         case 0x04 : PORTB=0x02 ; // Taster2 -> LED2
                     break;
         default : PORTB=0x00 ;    // kein Taster
                     break;}
```

Interruptbezeichner für den ATmega16:

INT0_vect, INT1_vect, TIMER2_COMP_vect, TIMER2_OVF_vect, TIMER1_CAPT_vect, TIMER1_COMPA_vect, TIMER1_COMPB_vect, TIMER1_OVF_vect, TIMER0_OVF_vect, SPI_STC_vect, USART_RXC_vect, USART_UDRE_vect, USART_TXC_vect, ADC_vect, EE_RDY_vect, ANA_COMP_vect, TWI_vect, SPM_RDY_vect