

# Info- Blatt zur Formatierung und Kommentierung des Quellcodes

## Die Kommentierung des Quelltextes

Der Quelltext eines Programms oder Skriptes sollte folgendermaßen kommentiert werden:

- Übersichtlich
- Kurz
- Aussagekräftig
- Verständlich

### 1.Methode: Die Zeilenkommentierung

Der Kommentar wird durch ein `//` vom Code getrennt. Nach dem `//` wird die Kommentierung gesetzt und gilt bis zum Zeilenende. Die nächste Zeile wird wieder als Quellcode interpretiert.

Beispiel:      = Quellcode                           = Kommentar

```
P1M1 = 0xFE;                                      // P1^0 bidirektional
IT0 = 1;                                         // Interrupt 0 (int 0) ausgelöst bei fallender Flanke
```

Diese Methode wird genutzt um einzelne Zeilen zu kommentieren.

### 2.Methode: Die Blockkommentierung

Des Weiteren ist es möglich einen Kommentarblock mit `/*` zu beginnen und mit `*/` zu beenden. Alles zwischen diesen beiden Zeichen wird vom Compiler ignoriert.

Beispiel:      = Quellcode                           = Kommentar

```
/******
* Dieses Programm soll bei einmaligem Tasten-                      *
* druck eine LED im Sekundentakt zum Blinken                      *
* bringen.                                                              *
*****/
#include <reg932.h>
```

## Formatierungsregeln der Programmierung

### Horizontaler whitespace

Hierbei gibt es horizontalen whitespace und semi-whitespace. Als semi-whitespace werden Zeichen wie ein Punkt, Komma oder Unterstrich bezeichnet. Man verwendet diese Art von whitespace um Variablen- oder Funktionsnamen leichter lesbar zu machen.

Beispiel:

```
void v_int1(void) interrupt 2
{
    ...
}
```

oder:

```
sbLED_1 = P1^0;
```

Mit dem Einrücken am Zeilenanfang ist es einfach, dem Programm eine logische Struktur zu geben. Des Weiteren wird auch zwischen Operatoren wie =, +, -, ... immer ein horizontaler whitespace verwendet.

Beim klassischen Stil wird nach der Bedingungszeile eine geschweifte Klammer gesetzt, die gegenüber der oberen Zeile nicht eingerückt wird. Der nachfolgende Anweisungsblock wird jedoch 4 Leerzeichen nach rechts eingerückt. Die schließende Klammer wird nach dem Anweisungsblock, wieder auf dieselbe horizontale Linie gesetzt, wie die öffnende Klammer.

Beispiel:     = whitespace zwischen Operatoren

    = Einrücken

```
if( sbReset == 0 )
{
   btNotaus == 0;
}
```

```
if( sbTaste == 0 )
{
   sbLED_1 = 0;
   sbLED_2 = 1;
}
```

öffnende Klammer  
Anweisungsblock 4 Zeilen eingerückt  
schließende Klammer

Diesen Stil verwendet man auch bei Funktionen:

```
void main (void)
{
    IT1 = 1;
    ...
}
```

Hierbei sollte darauf geachtet werden nicht mit Tabs einzurücken, sondern mit Leerzeichen!!! Denn Tabs können von jedem Editor oder Druckertreiber unterschiedlich lang ausgegeben werden.

### Vertikaler whitespace

Ein anderes Wort für vertikalen whitespace wäre eine Leerzeile. Anhand von diesen Leerzeilen ist es leicht möglich, Anweisungsblöcke zu gruppieren und optisch von anderen Anweisungen zu trennen. Mainprogramm und Funktionen sollten auch durch Leerzeilen voneinander getrennt werden.

Beispiel:

```
EX1 = 1;           // int 1 freigeben
EA = 1;           // Interruptsperre aufheben
```

```
btNotaus = 0;
```

```
while (1)
{
```