Befehlsübersicht Leistungskurs Elektrotechnik Q3: Embedded Systems

Kommentare

```
// einzeiliger Kommentar

/*
   mehrzeiliger
   Kommentar
*/
```

Elementare Datentypen

Typ	Größe in Bit	Wertebereich
`int`	16	Ganzzahl (Integer): -32.768 bis 32.767
`long`	32	Ganzzahl: -2.147.483.648 bis 2.147.483.647
`float`	32	Fließkommazahl: -3.4028235E+38 bis 3.4028235E+38
`double`	64	Fließkommazahl mit doppelter Genauigkeit
`boolean`	8	logischer Wert (Wahrheitswert): `true` oder `false`
`char`	8	Zeichen (ASCII-Code)

Deklaration von Variablen und Konstanten, Wertzuweisung (Beispiele)

```
int a;      // Die Variable a wird deklariert, sie ist vom Typ int.
a = 5;      // Der Variablen a wird der Wert 5 zugewiesen.

//Die Variable b wird deklariert, ihr wird der Wert 0.815 zugewiesen.
double b = 0.815;

//Die Konstante g wird deklariert, ihr wird der Wert 9.81 zugewiesen.
const double g = 9.81;
```

Häufig sind sehr viele Bauelemente (Sensoren, Aktoren) an den Pins eines Mikrocontrollers angeschlossen. Für einen gut lesbaren Programmcode ist es dann wichtig, dass die Nummern der Pins durch aussagekräftige Bezeichner ersetzt werden. Beispiel:

```
//An Pin 9 ist ein Taster angeschlossen.
//Der Zugriff auf diesen Pin ist jetzt über den Bezeichner taster möglich.
const int taster = 9;
```

Eindimensionale Arrays (Beispiel)

```
// Das Array meineLottozahlen wird deklariert.
// Es kann 6 Werte vom Datentyp int speichern (Index 0 bis 5).
int meineLottozahlen[6];

//Im Array meineLottozahlen wird unter dem Index 0 die Zahl 8 gespeichert.
meineLottozahlen[0] = 8;
```

Strings

String-Variablen speichern Zeichenketten und sind keine elementaren Datentypen. Sie können aber ähnlich verwendet werden. Beispiel:

```
//Die Variable str wird deklariert und mit "Guten Tag" gefüllt.
String str = "Guten Tag";
```

Mathematische Operatoren

Operator	Bedeutung	Beispiel
`+`	Addition	`y = 3 + 4`
`-`	Subtraktion	`m = 7 - x`
`*`	Multiplikation	`n = a * b`
`/`	Division	`y = 10 / 2`
`%`	Modulo (liefert ganzzahligen Rest einer Division)	`a = 20 % 7`
`++`	Inkrement	`i++` (i wird um 1 erhöht)
`--`	Dekrement	`i--` (i wird um 1 verringert)

Vergleichsoperatoren

Operator	Bedeutung	Beispiel
`==`	gleich	`x == 5`
`!=`	ungleich	`x != 5`
`<`	kleiner als	`x < 5`
`>`	größer als	`x > 5`
`<=`	kleiner als oder gleich	`x <= 5`
`>=`	größer als oder gleich	`x >= 5`

Logische Operatoren

Operator	Bedeutung	Beispiel
`!`	Negation	`!(a > b)`
`&&`	logisches UND	`(a > b) && (c < 5)`
`||`	logisches ODER	`(a > b) || (c < 5)`

Kontrollstrukturen

<u>Ja/Nein-Abfrage (Beispiel)</u>

```
if(a < 5)
{
      // Anweisungen im "Ja-Block"
}
      else
      {
            // Anweisungen im "Nein-Block"
      }
```

Falls die Bedingung a < 5 wahr ist, werden die Anweisungen im "Ja-Block" ausgeführt. Falls nicht, werden die Anweisungen im "Nein-Block" ausgeführt.

Kopfgesteuerte Schleife (Beispiel)

```
while(a < 5)
{
      // Anweisungen im Schleifenkörper
}
```

Zuerst wird geprüft, ob die Bedingung a < 5 wahr ist. Falls ja, werden die Anweisungen im Schleifenkörper ausgeführt. Dieser Vorgang wiederholt sich und endet erst dann, wenn die Bedingung a < 5 nicht mehr wahr ist.

Fußgesteuerte Schleife (Beispiel)

```
do
{
      // Anweisungen im Schleifenkörper
}
while(a < 5);
```

Zuerst werden die Anweisungen im Schleifenkörper ausgeführt. Danach wird geprüft, ob die Bedingung a < 5 wahr ist.
Dieser Vorgang wiederholt sich und endet erst dann, wenn die Bedingung a < 5 nicht mehr wahr ist.

Zählschleife (Beispiel)

```
for(int i = 0; i <= 2; i++)
{
      // Anweisungen im Schleifenkörper
}
```

Die Anweisungen im Schleifenkörper werden dreimal (für i = 0, i = 1 und i = 2) ausgeführt.

Schleifen vorzeitig beenden

Alle Schleifen können durch den Befehl `break` vorzeitig beendet werden.

Wichtige Funktionen

pinMode()

Konfiguriert den Pin als Ein- oder Ausgang. Beispiele:
 	
`const int roteLED = 5;  //an Pin 5 ist eine rote LED angeschlossen`
`pinMode(roteLED, OUTPUT);  //Pin 5 als Ausgang setzen`
`const int taster = 9;  //an Pin 9 ist ein Taster angeschlossen`
`pinMode(taster, INPUT);  //Pin 9 als Eingang setzen`

digitalRead()

Liest den Zustand des Pins entweder HIGH (1, true) oder LOW (0, false) ein und gibt diesen als Wert 0 oder 1 zurück. Beispiel:

```
const int taster = 7;
int wert = digitalRead(taster); // liest von Pin 7 digital ein
```

digitalWrite()

Den Pin auf logisch HIGH oder LOW setzen.

Beispiel: `digitalWrite(4, HIGH); // setzt den digitalen Pin 4 auf HIGH`

analogRead()

Liest vom angegebenen Analogeingang ein.

Beispiel: `int wert = analogRead(A0);  // liest Pin A0 analog ein`

analogWrite()

Gibt ein PWM-Signal am definierten Pin aus. Beispiel:

```
analogWrite(3, 120); //PWM-Signal am Pin 3 mit Pulsweite 120
```

delay()

Stoppt die Ausführung des Programms für eine bestimmte Zeit.

Beispiel: `delay(500); // pausiert für 500 Millisekunden`

delayMicroseconds()

Stoppt die Ausführung des Programms für eine bestimmte Zeit.

Beispiel: `delayMicroseconds(100); // pausiert für 100 Mikrosekunden`

millis()

Gibt die Zeitdauer in Millisekunden zurück, welche seit dem Start des Mikrocontrollers vergangen ist. Beispiel:

```
long t1 = millis();
  // beliebiger Code
long t2 = millis();
```

Die Differenz `t2-t1` entspricht der Zeit in Millisekunden, welche zwischen den beiden Aufrufen der Funktion `millis()` vergangen ist.