1
Was ist ein Computer?
Und: Erste Schritte in C

Jörn Loviscach

Versionsstand: 25. September 2014, 18:37

Die nummerierten Felder sind absichtlich leer, zum Ausfüllen beim Ansehen der Videos:
http://www.j3L7h.de/videos.html

Bitte hier notieren, was beim Bearbeiten unklar geblieben ist

1 Was ist ein Computer?

Ein Computer = Rechner ist eine Maschine, die Daten verarbeitet. Typischerweise verlangt man auch noch, dass diese Maschine programmierbar ist, dass also der Job in weiten Grenzen nachträglich festgelegt werden kann. Ein Widerspruch in sich: Die in Waschmaschinen, Autos usw. versteckt eingebauten Computer („Embedded Controller“ oder Mikrocontroller) sind fest programmiert – machen also nur noch einen Job.

Die meisten der heutigen Computer treten weniger als Rechner in Erscheinung. Vielmehr messen, steuern und regeln sie oder verarbeiten Signale: MP3-Spieler, Handy, Fernseher, Navi, Bordelektronik eines Autos, Stromzähler.

Heute sind alle Computer elektronisch statt mechanisch und digital statt analog. Analog heißt stufenlos, wie ein Rechenschieber:

[1]


Digital ist abzählbar, abgestuft:

[2]


2 Binärsystem, Bit

Jeder Digitalrechner muss intern mit Zahlen hantieren. Die Zahlen werden im Rechner heute meist (aber nicht immer!) binär dargestellt, also im Zweiersystem = Binärsystem. Zur Erinnerung:

[3]


Jede einzelne Stelle einer Binärzahl heißt bit (von „binary digit“). Jedes Bit hat mögliche Zustände: 1 oder 0. Diese beiden Zustände kann man netterweise auch als „wahr“ und „falsch“ lesen. Ein Bit ist damit quasi ein Atom an Information. Informationsmengen werden in Bit gemessen.

Warum die binäre Darstellung? Sie vereinfacht die elektronischen Schaltungen: Jede Leitung führt entweder eine Spannung dicht bei Masse – das gilt dann als 0 – oder sie führt eine Spannung dicht an der Versorgungsspannung, was dann als 1 gilt. Das elektronische Rechenwerk muss dann pro Leitung nur noch mit diesen beiden Situationen umgehen können (vereinfacht gesprochen).

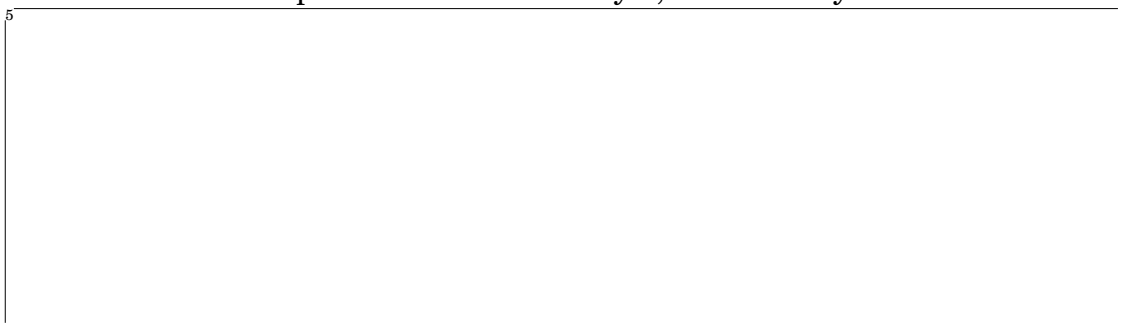
3 Byte, Kilobyte

Ein einzelnes Bit ist arg wenig. Mit acht Bits (eine schöne Zweierpotenz) hat man dagegen schon genug Kombinationen – nämlich [4] –, um die Zeichen der wesentlichen europäischen Alphabete zu codieren (siehe die Alt+...-Codes in der Zeichentabelle von Microsoft Windows). Die ersten 128 Positionen davon bilden den klassischen ASCII-Zeichencode aus den Zeiten der Fernschreiber. Dabei sind die ersten Codenummern und die letzte Codenummer nicht mit Buchstaben, sondern mit Steuerkommandos wie Wagenrücklauf (CR = Carriage Return) oder Zeilenvorschub (LF = Linefeed) belegt.

Ein Byte ist eine Binärzahl aus acht Bits. (Erst wurde das Byte mit i geschrieben: Bite = Biss; und es hatte auch nicht immer acht Bit.)

Früher war die Größe einer Textdatei in Byte etwa gleich der Zahl der Schriftzeichen inklusive Leerzeichen. Heute kann die Größe in Byte viel kleiner sein (dank Kompression) – oder auch doppelt so viel, um internationalen Zeichensätzen zu genügen.

Speicher wird auch binär addressiert: Speicherstellen im Rechner haben nicht „Hausnummern“ z. B. von 1 bis 1000, sondern sinnvollerweise von 0 bis 1023. Deshalb misst man Speicher nicht in 1000 Byte, Millionen Byte usw. sondern in:[5]



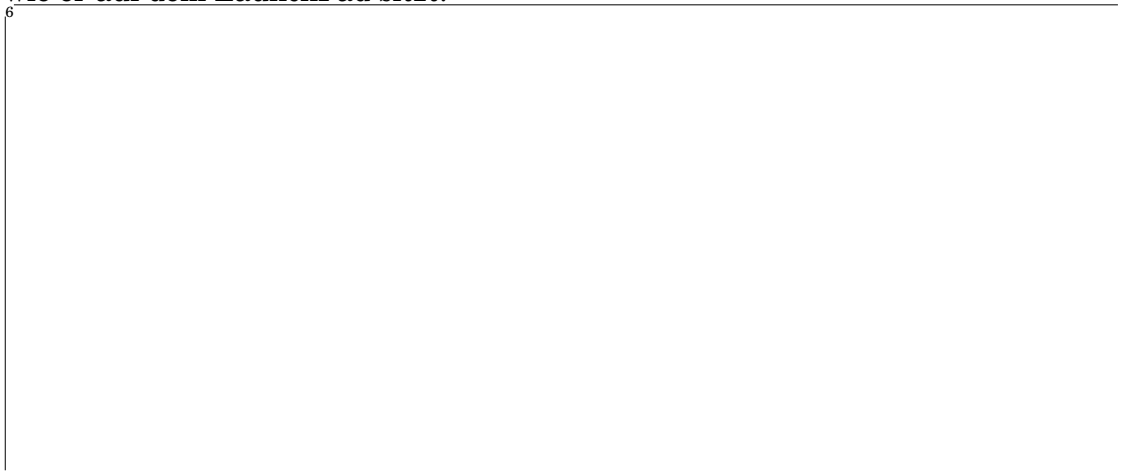
Diese Konvention gilt nicht bei Übertragungsraten: 1 MBit/s sind 1.000.000 kBit/s. Ebenfalls gilt sie nicht für die Herstellerangaben für Festplatten. (Warum?)

Demo: Größenangaben in den Dateieigenschaften von Microsoft Windows Explorer.

4 PC versus Embedded

In dieser Veranstaltung soll es sowohl um die Softwareentwicklung am herkömmlichen PC wie um die Softwareentwicklung für Mikrocontroller gehen, also die Steuerrechner, wie sie versteckt in Geräten landen. Als Beispiel für die Entwicklung mit Mikrocontrollern dient das Texas Instrument LauchPad.

Ein kleiner Vergleich zwischen einem herkömmlichen PC und dem MSP430G2231, wie er auf dem LaunchPad sitzt:[6]



Der zentrale Teil des LaunchPad ist die Fassung, in die man einen Mikrocontroller einsetzen kann, um ihn zu programmieren und zu testen. Die Anschlüsse dieses Chips sind nach außen geführt, so dass man auch gleich Experimente damit machen kann. (Diskussion der Platine)

Vorsichtsmaßnahmen: Die Platine nur am Rand anfassen. Vorher an einen großen metallischen Körper fassen, z.B. einen Heizkörper. Die Pins nicht kurzschließen. Die Platine im Betrieb nur auf eine isolierende Unterlage legen. Im Betrieb nicht auf die metallisierte Plastikverpackung legen.

5 Erste Schritte in C, Variablen, Typen

Um einen Computer zu programmieren – ihm mitzuteilen, was er tun soll –, muss man sich verständlich machen. Das geht bisher nicht durch gutes Zureden, sondern nur mit Hilfe streng formaler Programmiersprachen.

Im Seminar haben wir bereits die ersten Schritte auf dem MSP430... in der Programmiersprache C gemacht. Das ist die klassische Programmiersprache für Mikrocontroller.

„Variablen“ speichern Werte; man kann sie sich wie Schubladen vorstellen, die mit den Namen der Variablen beschriftet sind. Die Sprache C ist pingelig, was den Typ (also die Art) der Variablen angeht – was für eine Sorte von Gegenstand man in die Schublade packt.

Als erste solcher Typen betrachten wir `int` und `bool`:

```
7
```

Damit `bool` in C funktioniert, muss man am Anfang des Programms `#include <stdbool.h>` ergänzen. (Erklärung später)

Nun können wir so etwas rechnen lassen:

```
int anton = 42;
int berta = 4 + 3*anton;
int carla = berta/3;
bool doris = false;
bool edgar = (carla > 7);
bool friedhelm = (carla == 3);
```

Die Leerzeichen und Zeilenumbrüche sind in der Sprache C fast egal. In diesem Programmfragment könnte man alle Leerzeichen und Zeilenumbrüche weglassen, bis auf die Leerzeichen hinter `int` und `bool`.

Es gilt Punktrechnung vor Strichrechnung. Klammern werden beachtet.

Das == steht in C für den Test auf Gleichheit. Das einfache = sagt dagegen, dass der Ausdruck rechts ausgerechnet werden soll und sein Wert in die Variable links gespeichert werden soll. Diese Unterscheidung zwischen == und = ist eine beliebte Fehlerursache. Es gibt aber sogar Sprachen mit einem ===.

Anders als in der Mathematik (und einigen anderen Programmiersprachen) darf man nach den vorigen Zeilen schreiben:

```
carla = 98;
```

Die Variable `carla` erhält damit an dieser Stelle einen neuen Wert. Man darf dann aber nicht mehr `int` davor schreiben, denn die Schublade ist sozusagen schon eingerichtet.

6 Verzweigungen

Mit Hilfe einer `if`-Verzweigung kann etwas anderes passieren, je nachdem, ob die in Klammern angegebene Bedingung erfüllt ist oder nicht:

```
int anton = 42;
int berta = 4 + 3*anton;
int carla = berta/3;
int doris = 0;
if(carla > 7)
{
  doris = 3;
}
else
{
  doris = 4;
}
```

Der Teil mit dem `else{...}` darf auch fehlen.

Vorsicht mit weiteren Variablen, die innerhalb der Schweifklammern eingeführt werden: Die sind außerhalb der Schweifklammern nicht verfügbar.

7 Eine erste Schleife

Oft sollen mehrere Befehle mehrfach hintereinander ablaufen. Eines der Konstrukte dafür in C ist die `while`-Schleife. Sie läuft so lange, wie der Ausdruck in Klammern wahr ist. Er wird vor jedem Durchgang – auch dem ersten Durchgang! – überprüft:

```
int a = 0;
while((P1IN & 8) == 0)
{
  a = a+1;
}
```

Eine simple Lösung für Programme, die auf Dauer laufen sollen, ist die Endlosschleife:

```
P1DIR |= 1;
P1OUT &= ~1;
while(true)
{
  if((P1IN & 8) == 0)
  {
    P1OUT |= 1;
  }
  else
  {
    P1OUT &= ~1;
  }
}
```

Dieselbe Anmerkung wie schon beim `if`: Variablen, die innerhalb der Schweifklammern eingeführt werden, sind außerhalb der Schweifklammern nicht verfügbar.