Programmierung von ATMEL AVR Mikroprozessoren am Beispiel des ATtiny13

Eine Einführung in Aufbau, Funktionsweise, Programmierung und Nutzen von Mikroprozessoren

Teil II: Wat iss ene Bit, Byte un Word?

Warum Computer so gerne binär sprechen

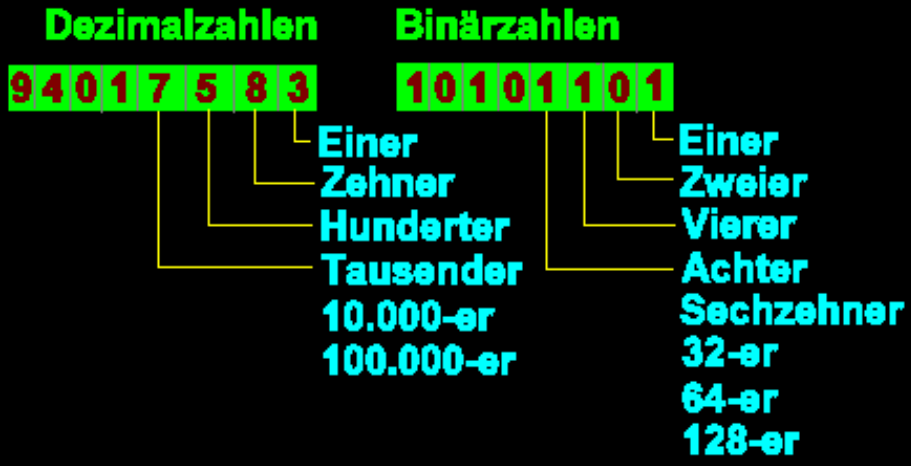
- Lampe: an und aus,
- Schalter: geschlossen und offen,
- Logik: wahr und falsch,
- Digitallogik: low und high,
- Ladung: Minus und Plus,
- MOSFET-Kondensator/SRAM: geladen und entladen,
- Serieller Loop: Strom und kein Strom,
- Lochstreifen/Lochkarte: Loch und kein Loch,
- Digitalelektronik/SRAM: Flipflop nach links und rechts gekippt,
- Kernspeicher/Tape: magnetischer Nord- und Südpol,
- CD/DVD: verkokelt und nicht verkokelt,
- u.v.a.m.
- Ist jetzt klar, warum Computer nur 0 und 1 sprechen oder muss ich noch weiter darüber philosophieren, wie man 10 verschiedene Abstufungen von „verkokelt“ oder „gelocht“ in Null-Komma-Nix erzeugt und erkennt?

Dezimal-, Binär- und Hexadezimalzahlen

- Dezimalzahlen:
  haben 10 Ziffern von 0..9,
  bei Zahlen die größer als 9 sind, werden Stellen davor angegeben
  Stellen davor sind 10, 100, 1000, ... , soviel wert wie ihre jeweilige Ziffer sagt
  Kennzeichnung in Assembler: Keine
- Binärzahlen:
  haben 2 Ziffern: 0 und 1
  bei Zahlen die größer als 1 sind, werden Stellen davor angegeben
  Stellen davor sind 2, 4, 8, 16, ... , soviel wert wie ihre Ziffer sagt
  Kennzeichnung in Assembler: 0b....
- Hexadezimalzahlen:
  haben 16 Ziffern: von 0..9 und A..F (10..15),
  bei Zahlen die größer sind als 15, werden Stellen davor angegeben
  Stellen davor sind 16, 256, 4.096, 65.536, ... , mal so viel wert wie die Ziffer sagt
  Kennzeichnung in Assembler: 0x... oder $..
- Alphabetimalsystem:
  hat 26 Ziffern: von A..Z
  bei Zahlen die größer sind als 25, werden Stellen davor angegeben
  Stellen davor sind 26, 676, 17.576, 456.976, ... , mal so viel wert wie die Ziffer sagt
  Kennzeichnung in Assembler: besser nicht, wenn dann 0a...
- Das Alphabetimalsystem wird im Folgenden nicht weiter behandelt, weil Computer einfach zu blöd sind, um das Alphabet zu kapieren.

Ähnlichkeiten zwischen Dezimal- und Binärzahlen

Zahlensysteme



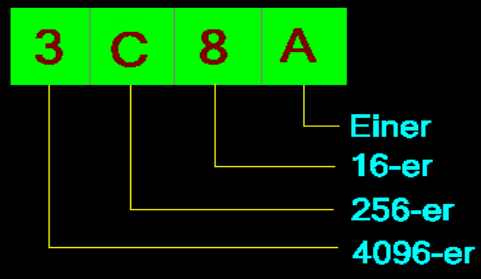
Umwandlung binär in dezimal:

10101101b = 1*128+0*64+1*32+0*16+1*8+1*4+0*2+1*1
= 128+32+8+4+1
= 173d

Verwandtschaft zwischen Dezimal-, Binär- und Hexadezimalzahlen

Zahlensysteme: Hexadezimal

Dezimal	Binär	Hexadezimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F



3C8Ah = 3*4096+12*256+8*16+10*1

= 12288+3072+128+10

= 15498d

Bytes: acht Binärziffern

- Eine Binärzahl mit acht binären Ziffern wird ein **Byte** genannt.
- Der darstellbare Wertebereich liegt zwischen Null (0b00000000) und (dezimal) 255 (0b11111111). Bezeichnung: vorzeichenlose Ganzzahl, engl. „unsigned integer byte“
- Die vier vordersten und die vier hintersten Bits können zu je einer Hexadezimalzahlziffer zusammengefasst werden (z.B. binär 0b10110111 ist hexadezimal 0xD7).
- Die vier vordersten Bits werden zusammen als „oberes Nibble“ und die vier hintersten zusammen als „unteres Nibble“ bezeichnet. In einem Nibble kann eine Dezimalziffer untergebracht werden (gepackte Binär Codierte Dezimal-Zahlen, packed BCD), die Zustände A bis F sind dann verboten. Unteres und oberes Nibble eines Bytes können z.B. mit der Instruktion SWAP vertauscht werden. Beispiel in Assemblersprache:

```
; Oberes und unteres Nibble in Register R16 vertauschen
      swap R16 ; tausche oberste und unterste vier Bits gegeneinander aus
```

- Wird das vorderste Bit nicht als Binärziffer, sondern als Vorzeichen-Bit interpretiert (bei negativen Zahlen ist das vorderste Bit gleich Eins), steht der Wertebereich von –128 (0b11111111) über 0 (0b00000000) bis +127 (0b01111111) zur Verfügung. Bezeichnung: vorzeichenbehaftete Ganzzahl, engl. „signed integer byte“.

Bytes in 8-Bit-Prozessoren

- Alle 8-Bit-Prozessoren verarbeiten 8 Bits gleichzeitig!
- Die Recheneinheit kann jeweils 8 Bit eines Registers (R0..R31) mit jeweils 8 Bit eines zweiten Registers (R0..R31) oder einer Konstanten (0..255) behandeln und speichert das 8 Bit breite Ergebnis in einem Register (R0..R31).
  Z.B. in Assemblersprache:

```
; Addiere R1 und R0, speichere Ergebnis in R1
      add R1,R0 ; Zeitbedarf: 1 Takt, @4MHz: 0,25 µs
```

- Überschreitet eine Operation (z.B. eine Addition oder Subtraktion) den zulässigen Wertebereich nach oben hin (Ergebnis ist größer als dezimal 255) oder nach unten hin (Ergebnis kleiner als Null), wird ein Status-Bit (engl. „flag“) gesetzt (Carry-Bit, C). Ist das Ergebnis Null, wird das Status-Bit Z gesetzt (Zero). Die Status-Bits können in Sprungbefehlen abgefragt und dazu benutzt werden, die Verarbeitung in verschiedenen Programmteilen fortzusetzen (bedingte Sprünge, engl. „conditional branch“).
  Z.B. in Assemblersprache:

```
; Subtrahiere R1 und R0
      sub R1,R0 ; ziehe R0 von R1 ab, Ergebnis in R1
      breq gleich ; Ergebnis war Null, springe nach gleich:
      brcs kleiner ; Ergebnis kleiner Null, springe nach kleiner:
      ... ; Ergebnis war größer Null, mache weiter
gleich:
      ... ; Ergebnis war gleich, mache was anderes
kleiner:
      ... ; Ergebnis war kleiner, mache ganz was anderes
```

Hast Du da noch Worte?

- 16 Bit breite Binärzahlen werden als Worte (eng. word) bezeichnet.
- Der Wertebereich reicht von 0 bis 65.535 (unsigned integer word).
- Jeweils vier Bits eines Wortes können als Hexadezimalziffer zusammengefasst werden (z.B. 0b1001.0101.1110.1010 ist gleich 0x95EA).
- Das obere Byte eines 16-Bit-Wortes wird als höherwertiges Byte oder „most significant byte“ (MSB), das untere Byte als niederwertiges Byte oder „least significant byte“ (LSB) bezeichnet.
- Wird das vorderste Bit als Vorzeichen interpretiert, reicht der Wertebereich von –32768 bis +32767 (signed integer word).
- 8-Bit-Prozessoren können auch 16-Bit-Zahlen verarbeiten. Dazu werden zwei 8-Bit-Register nacheinander verarbeitet. Im Prinzip können Paare aus beliebigen Registern gebildet werden, z.B. R5:R6 (R5 ist dann MSB, R6 ist LSB, „:“ signalisiert den Zusammenhang beider) und nacheinander behandelt werden.
- Einige Instruktionen erlauben die direkte Verarbeitung bestimmter Doppelregister. Bei den AVR sind dies die drei Registerpaare R27:26, R29:R28, R31:R30, sie werden als X (XH:XL), Y (YH:YL) und Z (ZH:ZL) bezeichnet. In sehr eingeschränktem Umfang ist auch R25:R24 für bestimmte Operationen zulässig, das Paar hat keinen eigenen Namen. Beispiele in Assembler:

```
adiw R24,1 ; erhöhe Wort R25:R24 um Eins
sbiw XL,5 ; erniedrige Wort X um fünf
movw ZL,XL ; kopiere Inhalt von XH:XL nach ZH:ZL
st Z+,R0 ; kopiere Inhalt von Register R0 zur Speicheradresse in Z und erhöhe die Adresse um Eins
ldd R16,Y+14 ; kopiere den Inhalt der Speicherzelle, auf die Y+14 zeigt, in das Register R16
```

Wo braucht man so was?

- Beispiel Frequenzzähler:
- Eingangsfrequenz für die Messung: max. 20 MHz (50 ns)
- Messdauer: 0,25 Sekunden
- Zählung mit eingebautem 8-Bit-Counter: Überlauf des Zählers alle 256*50 ns = 12,8 µs
- Zählung der Zählerüberläufe, maximale Anzahl = 20.000.000 / 4 / 256 = 19.531
- Anzahl Überläufe ist größer als 255 (Fassungsvermögen 1 Byte), nicht größer als 65.535 (Fassungsvermögen 2 Bytes), benötigt zwei Bytes für Überläufe
- Zusammen mit 1 Byte letzter Zählerstand: 3 Byte lange Zahl
  Speicherung z.B. in R3:R2:R1
- Multiplikation der Zahl mit 4 ergibt Frequenz in Hz, kann dabei ein Überlauf auftreten?
  20.000.000 ist hexadezimal 0x01.31.2D.00, benötigt also vier Bytes!
- Beispiel für Multiplikation mit vier in Assembler:

```
; R3:R2:R1 mit vier multiplizieren = Binärziffern zwei Mal links schieben
      clr R4 ; wird wegen Überlauf gebraucht, Null setzen
      lsl R1 ; niedrigstes Byte links schieben, Überlauf in Carry
      rol R2 ; zweites Byte links rotieren, Carry einschieben
      rol R3 ; drittes Byte links rotieren, Carry einschieben
      rol R4 ; allerhöchstes Byte links rotieren, Carry einschieben
      lsl R1 ; niedrigstes Byte noch mal links schieben
      rol R2 ; zweites Byte noch mal links rotieren
      rol R3 ; drittes Byte noch mal links rotieren
      rol R4 ; viertes Byte noch mal links rotieren, und fertig
; Zeitbedarf: 9 Takte, bei 4 MHz Taktfrequenz: 2,25 µs
```

Darf‘s ein bisschen mehr sein?

- Zahlen können fast beliebig groß sein. Dazu werden drei oder mehr Register zusammengefasst und als eine Zahl interpretiert und verarbeitet.
- Drei Bytes (z.B. R2:R1:R0) ergeben dann einen Wertebereich von 0 bis 16.777.215, vier Bytes (z.B. R15:R14:R13:R12) einen Wertebereich bis 4.294.967.295, usw.
- Die Verarbeitung solcher Zahlen (z.B. Erhöhen, Erniedrigen, Addition, Subtraktion) erfolgt dann Byte für Byte in einzelnen Schritten, wobei das Übertragsbit Carry C für Überträge aus niedrigeren Bytes und das Zerobit Z mit verwendet werden.
- Beispiel in Assemblersprache:

```
; Erhöhen von R32:R1:R0 um Eins
      inc R0 ; erhöhe niedrigstes Byte um Eins
      brne fertig ; wenn kein Überlauf zu Null springe nach fertig
      inc R1 ; Überlauf, erhöhe nächsthöheres Byte um Eins
      brne fertig ; wenn kein Überlauf zu Null springe nach fertig
      inc R2 ; Überlauf, erhöhe höchstes Byte um Eins
fertig: ; hier landen alle Fälle letztendlich
```

Schlussfolgerungen

- Es gibt beliebig viele Zahlensysteme, und man wundert sich, weshalb unsere Vorfahren ausgerechnet das Zehnersystem ausgewählt haben und nicht das Zweier-, Dreier-, Vierer-, Siebener-, Zwölfer-, Sechsundzwanziger- oder Sechziger-System. Wo wir doch zwei Füsse und zwei Hände haben (= vier), das Bier im Sixpack handhaben (=6), der Sieben eine gewisse Magie zugeschrieben wird (=7), ein Bierkasten 12 Flaschen fasst (=12), das Sechsundzwanziger-System mit dem Alphabet abzuwickeln wäre (Ziffern wären überflüssig) und Zahlen im 60-er System so schön kurz sind. Tradition ist manchmal arg ineffektiv.
- Computer sprechen binär, manchmal nibbelig und eher nebenbei etwas hexadezimal, keinesfalls aber dezimal!
- Die Zahlenumwandlung in binär/hexadezimal ist für‘s Denken etwas mühsam, aber aus Computersicht höchst effektiv!
- Auch im kleinsten Mikrokontroller lassen sich noch größte Zahlen mühelos verarbeiten!
- Für‘s Handling von 32- und 64-Bit-Zahlen braucht es keinen High-End-PC, es reicht ein 8-beiniges Mikrokontrollerchen!
- So eine Multiplikation mit vier geht auch bei nur 4 MHz Takt noch sauschnell.
- GHz-CPU-Takt ist dagegen total out, weil der PC mehr mit Fensterln und Platten beschäftigt ist als mit Rechnen!