Geekclock und Elektronik - Dokumentation

Andreas Müller

10. Juni 2007

Zusammenfassung

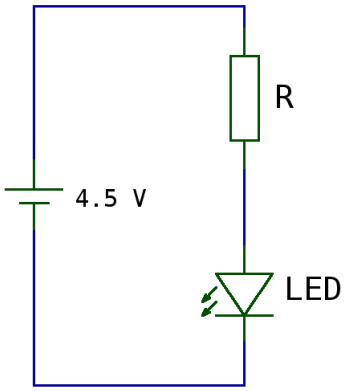
In diesem Artikel werden einige Grundlagen für das Bauen elektronischer Schaltungen erklärt und technische Details der Geekclock (original und simplified), sowie deren Aufbau und Programmierung beschrieben.

Inhaltsverzeichnis

1 Elektronik - Grundlagen

Es werden hier nur die grundlegendsten Dinge angesprochen. Für Details sei auf die Wikipedia (Kategorien Elektrotechnik [1], Bauteile [2]) verwiesen.

1.1 Stromkreis



- Strom fliesst nur in einem geschlossen Stromkreis
- im Beispiel sind die LED und der Widerstand in Serie geschaltet
  - im ganzen Stromkreis fliesst der gleich Strom
  - nicht an jedem Element fällt die gleiche Spannung ab
  - bei einer Parallelschaltung wäre es umgekehrt

1.2 Ohmsches Gesetz

Das Ohmsche Gesetz ist das Grundgesetz der Elektrotechnik. Es besagt, dass die Spannung über einem idealen Widerstand proportional ist, zum durch ihn fliessenden Strom:

$$U = R * I$$

mit Spannung U (gemessen in Volt $[V]$), Strom I (in Ampère $[A]$), Widerstand R (in Ohm $[\Omega]$).

1.2.1 Beispiel

Um im obigen Beispiel den benötigten Vorwiderstand für die LED zu berechnen, falls die Batterie 4.5 Volt liefert und an der LED 2 Volt abfallen und man einen Strom von 10mA möchte, würde man zuerst die Spannung über dem Widerstand ermitteln (Differenz der Batterie- und Ledspannung: $U_R = 4.5V - 2V = 2.5V$) und aus dem Ergebnis mit dem Ohmschen Gesetz den Widerstand berechnen:

$$R = \frac{U}{I} = \frac{2.5V}{10mA} = \frac{2.5V}{0.01A} = 250\Omega$$

Es würden also 250 Ohm benötigt.

1.3 Knotenpunkt- und Maschenregel

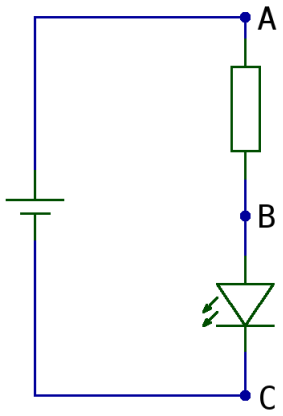
Zwei weitere wichtige Gesetze sind die Knotenpunkt- und die Maschenregel (Kirchhoff'sche Gesetze).

Knotenregel: Die Summe aller Ströme in einem Knoten ist Null ($\rightarrow$ es gehen keine Elektronen verloren).

Maschenregel: Die Summe aller Spannungen in einer Masche ist Null ($\rightarrow$ es fällt über einer idealen Leitung keine Spannung ab).

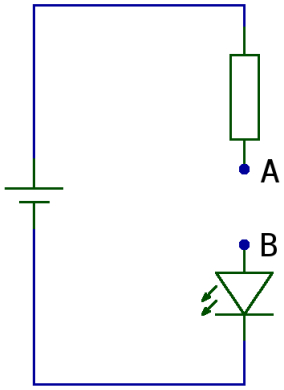
1.4 Ströme und Spannungen messen

1.4.1 Spannungsmessung



- Voltmeter auf "V=" (Gleichspannung)
- bei unbekannter Spannung beim grössten Spannungsbereich mit Messen beginnen
- eine Spannung wird immer über einem Bauteil gemessen (i.e. die Spannung, die am Bauteil abfällt)
  - im Beispiel würde eine Messung zwischen A und B die Spannung über dem Widerstand ergeben
  - eine Messung über B und C die Spannung über der LED
- ist das Multimeter auf Gleichspannung eingestellt, so kann beim Messen nicht viel schief gehen (grosser Innenwiderstand)

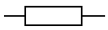
1.4.2 Strommessung



- Multimeter auf "I=" (Gleichstrom)
- beim grössten Strombereich beginnen
- der Stromkreis muss aufgetrennt werden
- Messung zwischen A und B misst Strom im ganzen Kreis
- ist das Multimeter auf Strommessung, so muss man aufpassen, keinen Kurzschluss zu produzieren, um nicht Bauteile oder das Multimeter zu zerstören
  - würde man z.B. so wie im Beispiel der Spannungsmessung zwischen A und B messen, so wäre der Widerstand kurzgeschlossen und die LED würde zerstört
  - würde man zwischen A und C messen, so wären beide Elemente kurzgeschlossen und das Multimeter (oder zumindest dessen Sicherung, falls vorhanden) würde zerstört

1.5 Bauteile

1.5.1 Widerstand

- Symbol: R
- Schaltzeichen: 
- Kenngrösse: Widerstand mit Einheit Ohm (Ω)
- Spannung über Widerstand ist proportional zu Strom
- grösserer Widerstand $\rightarrow$ kleinerer Strom fliesst
- Farbcodierung gibt Widerstandswert an (siehe Anhang A)
- wir verwenden Reihe E12 mit max. 0.25 Watt Belastbarkeit und $\pm 5\%$ Toleranz
- Einbaurichtung ist bei Widerständen egal
- Beispiele für Farbcodierung:
  - rot-rot-braun-gold: 220 Ohm

    * rot-violett-braun-gold: 270 Ohm
    * gelb-violett-braun-gold: 470 Ohm
    * braun-schwarz-orange-gold: 10 kOhm

1.5.2 Kondensator

- Symbol: C
- Schaltzeichen: 
- Kenngrösse: Kapazität mit Einheit Farad (F)
- Schaltzeichen für Elektrolytkondensatoren: 
- speichert Strom
- zeitliche Änderung der Spannung über dem Kondensator proportional zum Stromfluss:
$$i = C\frac{\mathrm{d}u}{\mathrm{d}t}$$
- Einbaurichtung nur bei Elektrolytkondensator relevant
- Werte sind meist direkt aufgedruckt
- Vorsilbe für Grössenordung wird als Komma wiederverwendet
    * $2n2 = 2.2nF = 2.2 * 10^{-9}F$
    * $\mu 33 = 0.33\mu F$
- bei kleineren Werten ist oft nur eine Zahl aufgedruckt; dann bezieht sich der Wert auf PicoFarad und die letzte Ziffer gibt die Anzahl Nullen an
    * $103 = 10nF$
    * $221 = 220pF$
    * $xyz = xy * 10^{z}pF$
- Toleranzen zu den aufgedruckten Werten ca -10% bis +50%

1.5.3 Diode und Leuchtdiode (LED)

- Symbol: D
- Schaltzeichen:  bzw.  (LED)
- Dioden lassen Strom nur in eine Richtung durch
- Einbaurichtung (Polarität) beachten
    * Dioden haben Ring bei Minus
    * LED's haben ein abgeflachtes Gehäuse bei Minus
- LED: Light Emitting Diode
- an LEDs verschiedener Farben fallen unterschiedliche Vorwärtsspannungen ab (erfordert passende Vorwiderstände)
    * rot: $1.2 - 2V$
    * gelb: $2.1V$
    * grün: $2.5 - 3.5V$
    * blau: $3.3 - 4V$

1.5.4 Quarz

- Symbol: Q
- Schaltzeichen: 
- liefert Referenzfrequenz
- Funktionsweise: Quarzplättchen mit angelegten Elektroden
  - beim Anlegen einer Spannung verbiegt sich der Quarz
  - nach Entfernen der Spannung verbiegt er sich zurück und produziert eine Spannung
  - positive Rückkoppelung nur bei der Resonanzfrequenz und bei Harmonischen davon

1.5.5 Der ATMega8 Mikrocontroller

- Ein Mikrocontroller (μC, MCU) ist ein CPU mit RAM, ROM und Peripherie auf einem Chip
- einige Features des ATMega8
  - RISC Architektur, bis 16 MHz Taktfrequenz
  - In-System programmierbar
  - integrierte AD-Wandler
  - integrierte Timer (wird für Geekclock verwendet)
  - max. 23 Pins für I/O verwendbar
  - Stromverbrauch 3mA
  - Sleep Mode
  - kann mit GCC programmiert werden
    - Tutorial unter [4]
  - avr-libc mit Standardfunktionen verfügbar [5]
- Datenblatt mit weiteren Infos unter [3]
- Pinbelegung siehe Anhang D

2 Löten

2.1 Allgemeine Tipps

- Nicht nur den Zinn mit dem Lötkolben erhitzen, sondern auch den Anschluss des Bauteils sowie die Leiterplatte. Die Metalle (Zinn und Kupfer) können sich nur verbinden, wenn sie die gleiche Temperatur haben (dies ist vermutlich der wichtigste Punkt überhaupt).
- Zinn nicht auf die Lötspitze, sondern immer direkt an die Leitstelle geben.
- Nicht zuviel Zinn verwenden (schafft Kurzschlüsse und erschwert allfälliges Entlöten).
- Empfindliche Bauteile möglichst schnell verlöten.
- Um Bauteile auf Leiterplatte zu befestigen, Beinchen umbiegen (erschwert allerdings ggf. das Entlöten).

2.2 Reihenfolge beim Auflöten

- unempfindliche Bauteile zuerst: Widerstände, Sockel, Kabel
- niedrige Bauteile vor hohen Bauteilen
- Bauteile, die durch Hitze beschädigt werden zuletzt: Transistoren, Quarze, Mikrocontroller

2.3 Gute/schlechte Lötstellen

gut

schlecht:
zuviel Zinn
→ kann abbrechen

schlecht:
Leiterbahn zu kalt
→ kein Kontakt

schlecht:
Draht zu kalt
→ kein Kontakt

3 Gehäuse bauen

Die meisten benötigten Materialien können in Handwerkshops gefunden werden.

3.1 Holz

- leicht zu bearbeiten
- elektrisch isolierend
- Beschriftung kann direkt mit dem Lötkolben eingebrannt werden
- Schrauben gegenüber Leim bevorzugt verwenden (gilt für alle Materialien)

3.2 Plexiglas

- schöne optische Effekte, z.B. bei abgeschrägten Kanten
- elektrisch isolierend
- kann gebrochen werden (einfacher als sägen!)
  - entlang Sollbruchstelle auf beiden Seiten kräftig einritzen
  - über einer harten Kante (z.B. eckige Tischkante) brechen

3.3 Metalle

- elektrisch leitend
  - schirmt elektromagnetische Felder ab
  - aufpassen, dass keine Kurzschlüsse entstehen
- Aluminium ist meist das bevorzugte Metall (einfach zu verarbeiten, geringes Gewicht, geringe Kosten)
- bitte keine Holzschrauben für Metalle verwenden!

4 Softwareumgebung für AVR MCUs

Zur Programmierung des ATMega8 können angepasste Versionen des gcc, der GNU libc und der GNU binutils verwendet werden. Zum Übertragen der Daten wird `uisp` verwendet. Unter Linux gibt es bei vielen Distributionen bereits die nötigen Packages (meist unter den Namen avr-gcc, avr-libc, avr-binutils und uisp). Die Installation ist von der Distribution abhängig:

- Gentoo

  ```
  # emerge uisp avr-libc crossdev
  # crossdev -t avr
  ```

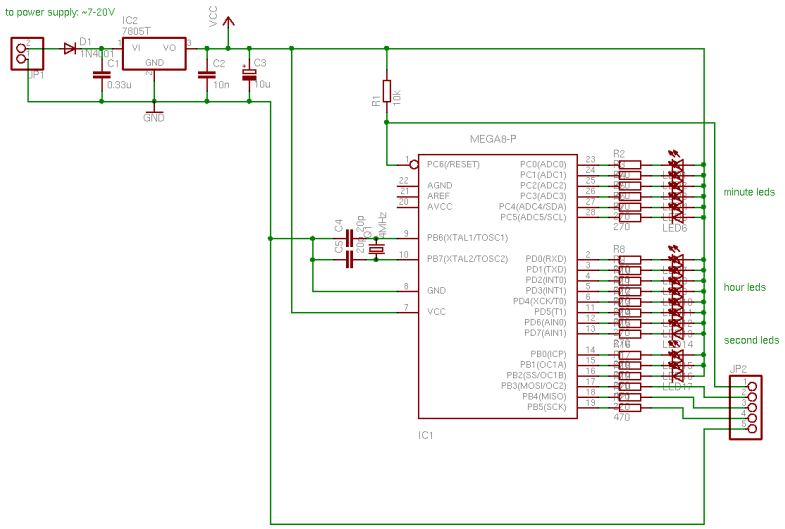
- Debian

  ```
  # apt-get install uisp avr-libc binutils-avr gcc-avr
  ```

- etc

5 Geekclock original

5.1 Hardware



- Kernstück ist der ATMega8 Mikrocontroller
- Quarz stabilisiert Takt auf exakt 4 MHz
- Oben links befindet sich ein Spannungsregulator (7805). Dieser sorgt dafür, dass am ATMega8 bei Eingangsspannung von 7-30Volt immer genau 5 Volt anliegen.
- Die vorgeschaltete Diode schützt vor verkehrt herum angelegter Versorgungsspannung (diese könnte den MCU zerstören).
- Rechts unten befindet sich die Schnittstelle für Programmierung über den Parallelport, mit vorgeschalteten Schutzwiderständen.

5.2 Software

- Zuerst werden die Pins initialisiert, damit sie als Output verwendet werden können
- Ausserdem wird der Hardwaretimer so initialisiert, dass ein Interrupt pro Sekunde generiert wird.
- Danach führt die Hauptroutine eine Endlosschleife aus.
- Die restlichen Aktionen finden in der Interruptbehandlungsroutine statt
- Bei jedem Interrupt werden die Zeitvariablen erhöht und die LEDs angepasst

5.3 Aufbau

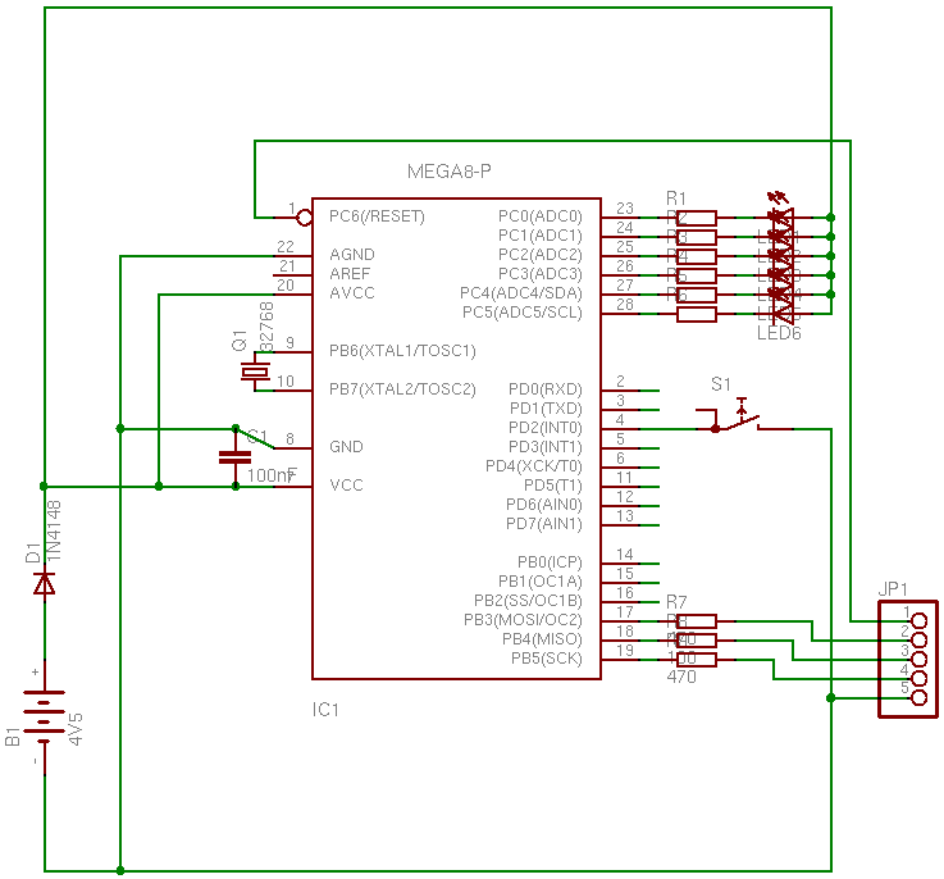
- zuerst alles zusammenlöten, MCU noch weglassen
  - der Quarz und die zugehörigen Kondensatoren sollten sich physisch möglichst nahe am MCU befinden
- Spannungen testen; nur bei korrekten Spannungen (5V von VCC zu GND, **nicht** -5V) MCU einstecken
- Programmierkabel gemäss Anhang C zusammenbauen
- Geekclock- und Ledtest-Code kompilieren (`make`)
- avrledtest laden (`make load_avrledtest`)
- Falls dabei ein Fehler auftritt, folgendes prüfen
  - sind die richtigen Pins vom LPT mit dem MCU verbunden?
  - hat der MCU genau 5V?
  - evtl. ist die parallele Schnittstelle zu schwach. Dann sollte der 220 Ohm Widerstand zwischen MISO und Busy überbrückt werden
  - ist der MCU defekt?
  - bei der Fehlersuche ist es hilfreich, verschiedene Kabel, PCs und MCUs auszuprobieren
  - falls es in der Umgebung viele elektromagnetische Felder gibt, so kann ein zusätzlicher 10nF Kondensator am MCU zwischen VCC und Masse helfen (dieser sollte nahe am MCU sein)
- fuses **korrekt** setzen (falsche Fuse Bits können den MCU unbrauchbar machen)
  - Befehle:
    - `uisp -drop=dapa -dlpt=0x378 --rd_fuses`
    - das High Bit ist vermutlich bereits d9, ansonsten: `uisp -drop=dapa -dlpt=0x378 --wr_fuse_h=d9`
    - `uisp -drop=dapa -dlpt=0x378 --wr_fuse_l`
  - die Schaltung funktioniert jetzt nur noch mit dem Quarz

    - falls `make load_avrledtest` vorher funktionierte, aber jetzt nicht mehr will, so ist vermutlich der Quarz nicht richtig mit dem MCU verbunden
    - mit einem Oszilloskop kann man die 4MHz am Quarz messen
- geekclock laden (`make load`)
- Programm beliebig abändern und neu laden
- Eichen: Zeitabweichung messen und an der entsprechenden Stelle in der Software (bei der Interruptbehandlung) korrigieren

6 Geekclock simplified

Die zweite Version der Geekclock ist gegenüber der ersten deutlich vereinfacht und hat den Vorteil, dass der Stromverbrauch klein genug ist für Batteriebetrieb.

6.1 Hardware



Hauptunterschied gegenüber der ersten Version ist der Quarz. Statt 4MHz wird nun ein Uhrenquarz mit 32768 Hz verwendet. Damit steht

zwar deutlich weniger Rechenleistung zur Verfügung, doch es wird sich zeigen, dass trotzdem einiges möglich ist. Als weitere Vereinfachungen wurden der Pull-Up Widerstand für Reset und die Kondensatoren für den Quarz weggelassen. Beides kann im ATMega8 intern aktiviert werden.

Der grosse Vorteil der sehr kleinen Taktfrequenz ist, dass der MCU nur noch sehr wenig Strom braucht ($20 - 30\mu A$ – damit wird eine Batterielebensdauer von mehreren Jahren erreicht). Dies liegt daran, dass bei CMOS Logik im statischen Zustand kein Strom fliesst. Der Stromverbrauch ist also davon abhängig, wie oft die einzelnen Gatter umgeschaltet werden (beim Umschalten wird jedes mal die Gatekapazität ge- oder entladen). Die LEDs werden bei dieser Version nur auf Knopfdruck aktiv, da sie für Dauerbetrieb zuviel Strom verbrauchen.

Bauteilliste

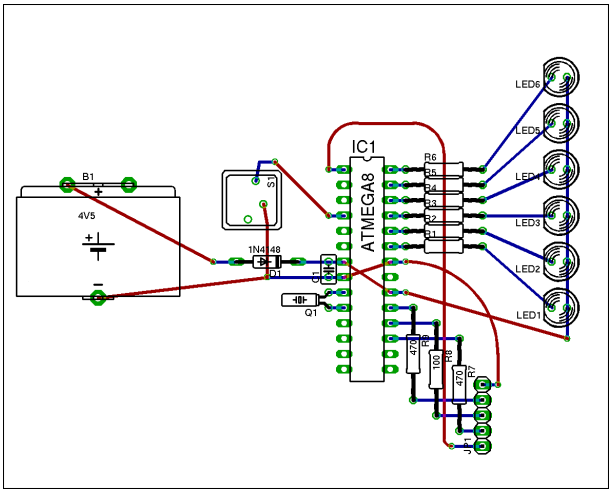
- B1 – 4.5V Flachbatterie
- IC1 – ATMega8 oder ATMega8L
- Q1 – Quarz, 32768 Hz
- C1 – Kondensator, 100nF
- S1 – Taster
- R1-R6 – Widerstände, $100 - 270\Omega$ (abhängig von LED-Farbe)
- R7,R9 – Widerstände, 470Ω
- R8 – Widerstand, 220Ω
- LED1-6 – Leuchtdioden
- D1 – Siliziumdiode, 1N4148
- JP1 – Buchsenleiste

6.2 Software

Die Software ist ähnlich wie beim Original: Beim Einschalten des Chips wird der Timer so initialisiert, dass sich jede Sekunde ein Interrupt ereignet; in der Interruptbehandlungsroutine wird entsprechend die Zeit angepasst. Der Sourcecode und ein Makefile befinden sich im Verzeichnis `software/simplified`:

- `geekclock.c` (Hauptcode): Enthält Interruptbehandlungen und main-Routine
- `lowlevel.c`: Enthält Initialisierung (Timer, Ports) und lowlevel Funktionen (`delay_ms()`, `button_down()`, `set_leds()`)
- `datetime.c`: Kalenderfunktionen (`is_leap_year()`)
- `led.c`: LED-Ansteuerung und Effekte

6.3 Aufbau



- IC-Sockel, Widerstände, Leuchtdioden, Drähte und Büroklammern für den Batterieanschluss anlöten
- zuletzt den Quarz auflöten (temperaturempfindlich!)
- Batterie anschliessen, Spannung im Sockel testen (+4.5V zwischen VCC und GND im Sockel?), Batterie wieder entfernen
- IC (ATMega8) einstecken (Beine müssen evtl. vorher zurechtgebogen werden)
- Programmierkabel anschliessen
- Software übersetzen (`make` im Verzeichnis `software/simplified`)
- avrledtest laden (`make load_avrledtest`)
- Schalter testen (sollt Blinken auf einzelne LEDs umstellen)
- Fuses setzen (`make write_fuses_32khz`); nun verwendet der MCU den externen Quarz zur Taktgenerierung und funktioniert nicht mehr ohne
- Software für Geekclock laden (`make load`)
- Stromverbrauch messen (muss $< 100\mu A$ sein)
- einige Tage laufen lassen und Drift messen; entsprechenden Ausgleichswert in der Software eintragen und `make;make load` ausführen

Die Uhr kann nun beliebig umprogrammiert oder erweitert werden. Es ist auch eine komplett neue Funktionalität möglich (z.B. Stoppuhr oder elektronischer Würfel).

A Widerstandscodierung

Die Widerstände der Reihe E12 haben 4 Ringe, von denen die ersten 3 den Wert (zwei Ziffern und Multiplikator) und der vierte die Toleranz angeben. Der Multiplikator kann auch direkt als Anzahl an die Ziffern angehängte Nullen abgelesen werden.

	1. Ring	2. Ring	3. Ring	4. Ring
Farbe	1. Ziffer	2. Ziffer	Multiplikator	Toleranz
schwarz	0	0	1	
braun	1	1	10	±1%
rot	2	2	100	±2%
orange	3	3	1000	
gelb	4	4	$10^4 = 10'000$	
grün	5	5	$10^5 = 100'000$	
blau	6	6	$10^6 = 1'000'000$	
violett	7	7	$10^7 = 10'000'000$	
grau	8	8	10^8	
weiss	9	9	10^9	
silber	-	-	0.01	±10%
gold	-	-	0.1	±5%

B Grössenordnungen bei Einheiten

Vorsatz	Zeichen	Faktor
Exa	E	10^{18}
Peta	P	10^{15}
Tera	T	10^{12}
Giga	G	10^9
Mega	M	10^6
Kilo	K	10^3
Milli	m	10^{-3}
Mikro	μ	10^{-6}
Nano	n	10^{-9}
Pico	p	10^{-12}
Femto	f	10^{-15}
Atto	a	10^{-18}

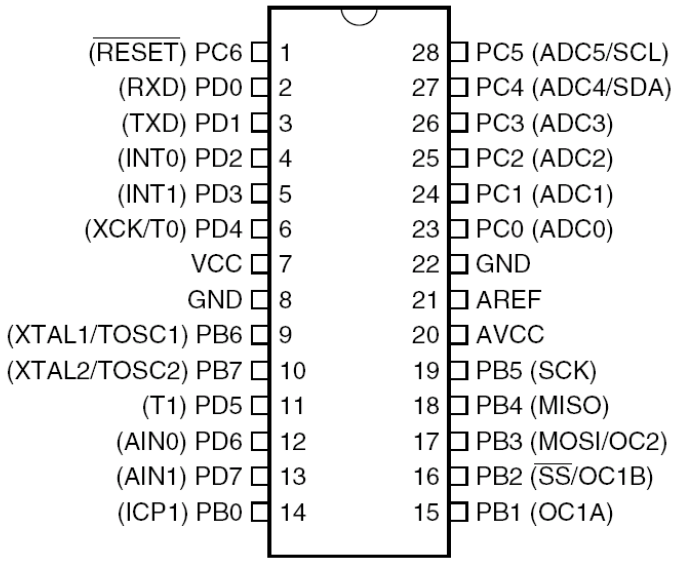
C Programmierkabel

Das Programmierkabel besteht nur aus einem LPT-Verbindungskabel. Es werden folgende Pins verbunden (Reihenfolge auf Jumper von oben nach unten).

Pin ATMega8	Pin auf Jumper	Pin LPT
1 (Reset)	1	16 (Init)
17 (MOSI)	2	2 (D0)
18 (MISO)	3	11 (Busy)
19 (SCK)	4	1 (Strobe)
8 (GND)	5	18 (GND)

Die Pins auf den LPT-Steckern sind übrigens oft beschriftet, was sehr praktisch ist.

D Layout des ATMega8



E Links

Literatur

[1] http://de.wikipedia.org/wiki/Kategorie:Elektrotechnik
Wikipedia Kategorie Elektrotechnik

[2] http://de.wikipedia.org/wiki/Kategorie:Elektrische_Bauelemente
Wikipedia Kategorie Elektrische Bauelemente

[3] http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf
Datenblatt des ATMega8

[4] http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial
AVR-GCC Tutorial auf mikrocontroller.net

[5] http://www.nongnu.org/avr-libc/user-manual/
avr-libc Dokumentation

[6] http://www.tldp.org/linuxfocus/Deutsch/November2004/article352.shtml
Einführung für den ATMega8 von Linuxfocus

[7] http://www.tuxgraphics.org/electronics/200506/article379.shtml
Artikel über eine digitale Laborspannungsversorgung mit dem ATMega8

[8] http://people.ee.ethz.ch/ andrmuel/
Meine Website – auch einige Mikrokontrollerprojekte :-)

[9] http://www.ulrichradig.de/
Website mit einigen interessanten ATMega Projekten, u.a. einem kompletten Webserver