embedded-projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Eine Open-Source Zeitschrift
zum Mitmachen!

Weihnachten

[PROJECTS]

- Softcore auf Spartan-3 FPGA
- Laser-Pointer-Fernseher
- AVR XMEGA Toolchain unter Linux
- Transistortester im Eigenbau
- Infrarotauslöser für Nikon Kamera
- Röhrenradio im Nachbau
- BGA-Löten mit dem Pizzaofen

Einleitung

Weihnachtsausgabe 2011

Embedded Projects Journal - Ausgabe No. 11

Schwup !

...und schon wieder ist ein Jahr vorbei. Wir freuen uns die 11. Ausgabe des embedded projects Journal präsentieren zu können. 2011 war für das Projekt Open-Source Zeitschrift ein sehr positives Jahr.

Denn seit diesem Jahr gibt es eine Internetseite. Nach und nach erhalten wir diverses Feedback von Lesern. Darüber freuen wir uns immer besonders. Schickt gerne Eure Kommentare - ob positiv oder kritisch an: journal@embedded-projects.net

Bis letztes Jahr konnte man gar nicht genau sagen, wie oft das Heft eigentlich auch online gelesen wurde.

In der Summe sind es auf unserer Internetseite in etwa 10.000 Online-Leser. Von den Papierlesern sind es knappe 120 Spenden-Abos. Vielen Dank dafür. Wir werden 2012 verstärkt auf Infoveranstaltungen das Open-Source Projekt präsentieren und die 2500 Stck. Papier-Journals weiterhin unter die Leute bringen :)

Gerne könnt Ihr uns natürlich anschreiben, wenn Ihr eine Hausmesse, Tagung oder einen Tag der offenen Tür habt. Entweder senden wir Euch Exemplare zum Verteilen oder es kann auch jemand von uns den „Stand“ betreuen.

Was uns ebenfalls besonders freut ist, dass wir 2012 hochkarätige Sponsoren haben, und so immer mehr das Journal auf eigene Beine stellen können.

Vielen Dank an unsere bis jetzt treuen Sponsoren und natürlich auch an alle neu dazu gekommenen.

Vielen Dank für die Unterstützung an alle Helfer! Vom Leser über den Autor bis zum Sponsor. Wir freuen uns auf das neue Jahr 2012 mit Euch!

Benedikt Sauter und

das embedded projects Team

wawision.embedded-projects.net

waWision -
die Steuerzentrale für Ihre Firma

Verwal-tung	Plug & Play	Waren-eingang	Marke-ting
FiBu	Produk-tion	automa-tisches Lager	Online-Shops

EIN SYSTEM AUS EINER HAND

- keine Installation
- Betriebssystem unabhängig
- Standardhardware Plug & Play
- mitwachsend

DEMOVERSION

weitere Infos finden Sie auf unserer Internetseite

Holzbachstraße 4, D-86152 Augsburg
Tel +49 (0) 821 279599-0, Fax +49 (0) 821 279599-20
info@embedded-projects.net

embedded projects GmbH
HARDWARE FOR PROJECTS

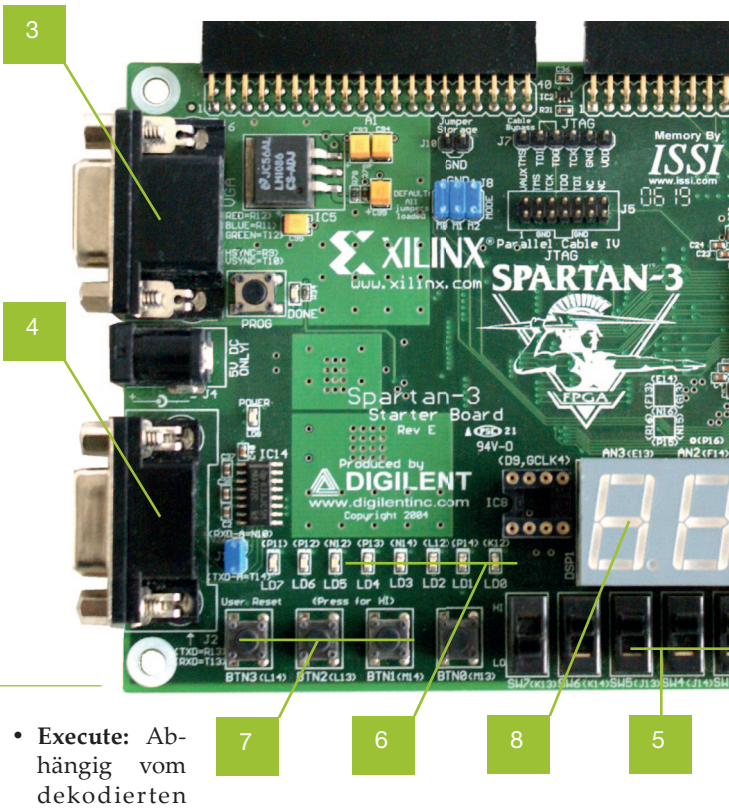
ZPU

Softcore Implementierung auf Spartan-3 FPGA

Johannes Steudle (Projekt an der DHBW Mannheim)

Einleitung

Im Rahmen des Projekts stand ein Experimentierboard Spartan-3 von Xilinx zur Verfügung. Das Spartan-3 ist ein Einsteigerboard (siehe Abb. re.) mit einem XC3S200 FPGA (1), das sich auf Grund der geringen Anzahl an BlockRAMs (12 Stk, entspricht 24 KB) nicht unbedingt für den Einsatz eines anspruchsvollen Softcore-Prozessors mit Peripherie eignet. Die wichtigsten Komponenten für das Projekt sind: eine PS/2- (2), eine VGA- (3), eine RS-232-Schnittstelle (4), 8 Schalter (5), 8 LEDs (6), 4 Buttons (7) und 4 Siebensegment-Anzeigen (8), des Weiteren der SRAM auf der Rückseite der Platine. Programmieren lässt sich das Board bzw. der FPGA mittels eines JTAG-Adapters. Eine passende Software - die Xilinx ISE Entwicklungsumgebung - lässt sich kostenlos von Xilinx herunterladen. Für das Projekt wurde die Version 10.1 verwendet. Die Auswahl fiel auf eine frei verfügbare Softcore-CPU, die ZPU von Øyvind Harboe. Sie ist eine kleine Stack-basierte 32 Bit CPU mit einem 8 Bit breiten Opcode. Sie hat einen einfachen und übersichtlichen Aufbau und ist deshalb optimal für die hier angestrebten Erweiterungen geeignet. Diese Opensource Softcore-CPU ist als VHDL-Projekt angelegt. Sehr nützlich ist die Möglichkeit, Programme, die in C oder C++ verfasst sind über eine spezielle GCC-Toolchain (ZPUGCC) in einen Maschinencode zu wandeln, den die ZPU direkt interpretieren kann. So kann man die ZPU mittels C-Programmen steuern. Verwendet wurde letztendlich deren „Small"-Variante.



Aufbau und Funktionsweise der ZPU

Die ZPU funktioniert im Prinzip wie eine Zustandsmaschine, die während der Abarbeitung im Wesentlichen die Phasen Fetch (Befehl holen), Decode (Befehl decodieren) und Execute (Befehl ausführen) durchläuft:

- **Fetch:** In diesem Zustand lädt die ZPU anhand des Program Counters einen kompletten 32 Bit Wert aus dem Programmspeicher. Hierbei wird der 32 Bit Wert über die oberen 30 Bit des Program Counters adressiert. Es ist zu beachten, dass der Program Counter zwar eine Breite von 32 Bit hat, die eigentliche Adressbreite des Programmspeichers jedoch abhängig von seiner Größe ist (s. u.). Dementsprechend kann eventuell nicht die volle Breite des Program Counters genutzt werden.

- **Decode:** Da die Opcodes nur 8 Bit breit sind, wird anhand der letzten zwei Bits des Program Counters entschieden, welche 8 Bit des 32 Bit breiten Speicherwertes dekodiert werden. So adressiert der Program Counter letztendlich immer 8 Bits. Dies ist ähnlich einer in CPUs genutzten Pipeline. Pipelines werden genutzt, um die Taktrate auf den Programmspeicher zu reduzieren. Hier würde durch den 32 Bit Speicherwert die CPU mit einem 8 Bit breiten Opcode 4x schneller takten können, als die maximale Taktrate des Speichers es erlaubt. Realisiert ist die ZPU jedoch so, dass sie in jedem Fetch-Zyklus auf den Speicher zugreift und einen 32 Bit Wert holt und dann nur 8 Bit nutzt. Insofern ist die Speicherzugriffstaktrate gleich der des Fetch-Zyklus.

- **Execute:** Abhängig vom dekodierten Opcode springt die ZPU in der Execution-Phase in verschiedene Zustände, um die Instruktionen abzuarbeiten. Des Weiteren wird der Program Counter um Eins erhöht.

Nach dem Fetch-State existiert zusätzlich noch ein Fetch-Next-State. Dies bedeutet, dass mindestens 4 Takte benötigt werden, um einen Befehl abzuarbeiten. Bei der Taktfrequenz der ZPU von 50 MHz ergibt dies eine maximale Frequenz von 12,5 MHz. Die ZPU besitzt direkt untergeordnet eine BlockRAM-Einheit, die sie als 32 Bit breiten Programmspeicher verwendet. Hierbei handelt es sich um einen Dualport-BlockRAM: Über beide Ports kann gleichzeitig auf die gleiche Adresse schreibend oder lesend zugegriffen werden. Jedoch nicht von beiden Ports gleichzeitig auf die gleiche Adresse schreibend. Außer dem Programm ist hier auch der Stack abgelegt, also auch der Datenspeicher. Der Speicher hat im Moment eine Größe von 16 KB. Im VHDL-Code entspricht die Struktur der ZPU einer Entity, die den ZPU Core integriert und nach außen hin eine Wishbone-Anbindung zur Verfügung stellt. Die Entity des ZPU Cores wiederum implementiert den Dualport-RAM.

Wishbone-Bus

Der Wishbone-Bus ist ein Opensource Hardware Computerbus, über den die Einheiten einer integrierten Schaltung miteinander kommunizieren können. Wishbone ist ein logischer Bus, er definiert also nicht elektrische Informationen. Stattdessen definiert die Spezifikation Signale, Taktzyklen und High- und Low-Pegel. Dies macht es so einfach, ihn in VHDL zu verwenden. Alle Signale sind dabei synchron zum Taktsignal. Für das Projekt wurde die 32 Bit Variante des Wishbone-Bus verwendet (32 Bit Adress- und 32 Bit Datenbreite). Als Topologie wird der Shared Bus verwendet, d.h. alle Teilnehmer sitzen am selben Adress- und Datenbus und es existiert nur ein Master, nämlich die ZPU. Sollten mehrere Slaves angeschlossen sein, wird anhand der auf dem Adressbus liegenden Adresse der entsprechende Slave aktiviert. Alle Peripherie-Geräte sind als Slaves angesetzt. Ein Beispiel für eine Verbindung vom Master zu einem Slave ist in Abb. 2 zu sehen. Hierbei enthält der Block SysCon den Taktgenerator des FPGAs und eine Verbindung zum Reset-Button. Eine Erklärung der Signale folgt in Tabelle 1. Beispiele für ein Wishbone-Output-Interface und Wishbone-Input-Interface sind in Abb. 3 und Abb. 4 zu sehen. Im Projekt wurden Schnittstellen für die zu Anfang des Artikels beschriebenen Komponenten realisiert.

Auf www.opencores.org gibt es viele frei verfügbare Peripherie-Geräte mit Wishbone-Schnittstelle zum downloaden. Die Schnittstelle muss trotzdem meist etwas angepasst werden. Sinnvoll ist es, einen Baustein zu verwenden, der bereits funktioniert, also in einem eigenen Projekt getestet wurde. So kann man diese Fehlerquelle beim Anbinden schon einmal ausschließen.

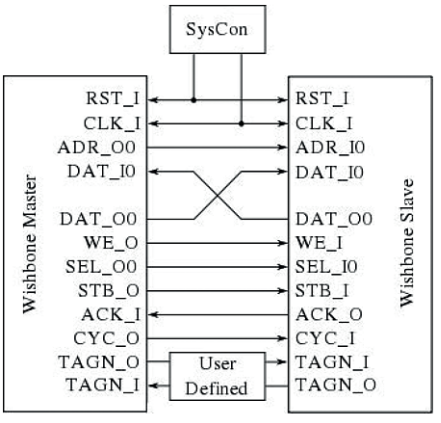


Abb. 2: Wishbone-Bus

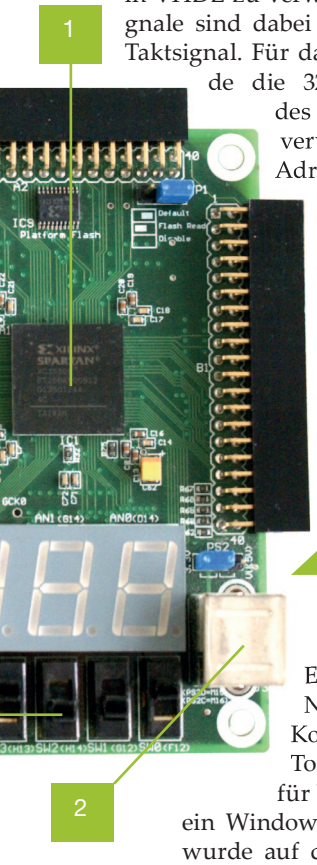


Abb. 1: Spartan-3 Board

Programmierung der ZPU

Die ZPU lässt sich in C bzw. C++ programmieren. Zum Erstellen der Quellcodes kann z.B. Notepad++ verwendet werden. Zum Kompilieren wird die spezielle ZPUGCC Toolchain verwendet. Diese ist jedoch für Linux gedacht. Zum Projekt stand nur ein Windows-Rechner zur Verfügung. Deswegen wurde auf diesem Cygwin installiert und unter Cygwin die ZPUGCC in Betrieb genommen.

Ansprechen von Hardware: Die absolute Hardware-Adresse kann über Zeiger des Typs volatile int gesetzt werden. Als Wert wird direkt die Adresse angegeben, auf die der Hardwaremultiplexer im FPGA überprüft. Über Dereferenzierung der Zeiger lässt sich der Wert auslesen oder setzen.

Interrupts: Für die Nutzung der Interrupts müssen diese zuerst aktiviert werden. Ab dem Zeitpunkt der Aktivierung reagiert die ZPU auf diese. Es ist ein Interrupt-Controller mit einem Wishbone-Interface vorhanden. Zusätzlich zu den Wishbone-Signalen hat er Eingänge für die Interrupts anderer Peripherie-Geräte und einen Ausgang, um den Interrupt der ZPU zu setzen. Wenn über letzteren der ZPU ein Interrupt mitgeteilt wird und diese ihn annimmt (bzw. Interrupts aktiviert hat), sendet der Controller ihr die Interruptnummer. Per Software kann der ZPU mitgeteilt werden, was bei Auftreten eines Interrupt mit diesem gemacht werden soll. Dies bedeutet auch, dass der Software-Programmierer wissen muss, welche Interruptnummer welchem Peripherie-Gerät zugeordnet ist. Mit 32 Bit Datenbreite sind so maximal 32 Interrupts möglich, da jedes Bit einen Interrupt präsentiert.

Kompilieren: Kompiliert wird der Quellcode mit ZPUGCC. Hierbei wird eine .elf-Datei erstellt, die direkt in den BlockRAM des FPGA geladen werden kann. Hierbei reicht es, dem Compiler die Haupt- C-Datei anzugeben, weitere Header, auch eigene etc. bindet ZPUGCC automatisch mit ein und kompiliert sie.

Programmspeicher der ZPU laden: Nach dem Erstellen der .elf-Datei durch die Kompilierung muss der Inhalt in den Programmspeicher (BlockRAM des FPGA) geladen werden. Abhängig von der Größe des Prozessorsystems und des verwendeten FPGAs kann die Synthese des Systems sehr langwierig sein. Damit nicht für jede Änderung am Programmspeicher der Cores das ganze System neu synthetisiert werden muss, wird von Xilinx das Kommandozeilen-Tool Data2MEM in der Entwicklungsumgebung des ISE zur Verfügung gestellt. Dieses ist standardmäßig bei der Installation von Xilinx ISE dabei. Außerdem spart man viel Arbeit, da man nicht jedes Byte einzeln hinein kopieren muss. Die Daten, die zur Konfigurierung eines FPGAs benötigt werden und damit auch die Programme der Softcores, befinden sich in der so genannten .bit-Datei. In dieser .bit-Datei kann Data2MEM ein altes Programm durch ein neues ersetzen. Um diese Transformation durchzuführen, benötigt Data2MEM Informationen darüber, wo Daten ersetzt werden sollen und wie diese aufgeteilt werden. Diese Informationen werden im BlockRAM Memory Map File angegeben. Aktuell stehen dem Core 16 KB (4096 x 32 Bit Datenbreite) Programmspeicher zur Verfügung, die in 8 BlockRAMs aufgeteilt werden. Neben dem Memory Map File muss Data2MEM noch die ursprüngliche .bit-Datei und das Programm als .elf-File übergeben werden. Das Ergebnis dieser Transformation ist die neue .bit-Datei mit geändertem Programmspeicher. Dieses wird dann auf den FPGA geladen.

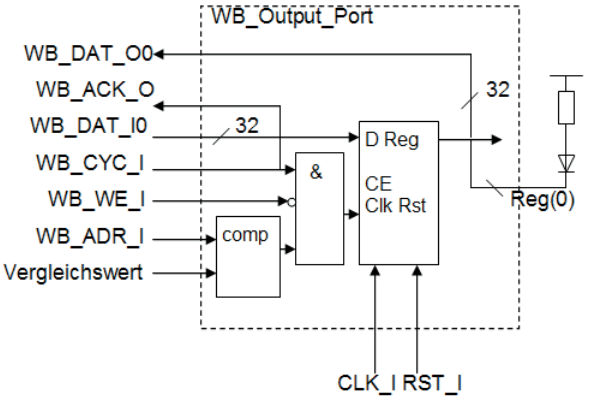


Abb. 3: Wishbone-Output-Interface

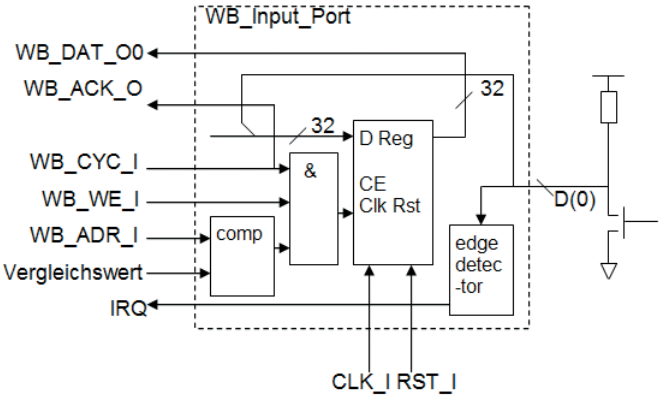


Abb. 4: Wishbone-Input-Interface

Links / Literatur

Das Projekt an sich und eine ausführliche Dokumentation kann von mir per Mail angefordert werden. Des Weiteren existiert ein Artikel mit der Projektbeschreibung, unter dem auch ein Downloadlink angegeben ist:

[1] http://www.mikrocontroller.net/articles/ZPU:_Softcore_Implementierung_auf_Spartan-3_FPGA

[2] Xilinx Spartan-3 Starter Kit: http://www.xilinx.com/products/devkits/HW-SPAR3-SK-UNIG.htm

[3] User Guide: http://www.xilinx.com/support/documentation/boards_and_kits/ug130.pdf

[4] JTAG-Programmier-Adapter: http://shop.trenzelectronic.de/catalog/product_info.php?cPath=30&products_id=591

[5] Xilinx Entwicklungumgebung: http://www.xilinx.com/support/download/index.htm

[6] ZPU Download: http://opensource.zylin.com/zpudownload.html

[7] Projekt Lava (Multicore-Systeme auf FPGAs): http://ess.cs.tudortmund.de/EN/Research/Projects/LavA/

[8] Diplomarbeit Matthias Meier: http://ess.cs.tudortmund.de/Teaching/Theses/2009/DA_Meier_2009.pdf

[9] Wishbone-Bus: http://en.wikipedia.org/wiki/Wishbone_%28computer_bus%29

[10] http://kujo.cs.pitt.edu/index.php/Wishbone_Interconnect

[11] http://www.armadeus.com/wiki/index.php?title=A_simple_design_with_Wishbone_bus

[12] Weitere Quellen s. Dokumentation

RST _ I	Synchrones Resetsignal
CLK _ I	Synchrones Taktsignal
ADRO0, ADR _ I0	32 Bit Adressbus
DAT _ O0, DAT _ I0	32 Bit Datenbus für Daten vom Master zum Slave
DAT _ I0, DAT _ O0	32 Bit Datenbus für Daten vom Slave zum Master
WE _ O, WE _ I	WE kennzeichnet, ob der aktuelle Buszyklus ein Schreib- oder Lesezyklus ist. WE ist negiert während des Lesezyklus
SEL _ O, SEL _ I	4 Bit breiter Selectbus. Jedes Bit aktiviert ein Byte aus dem übermittelten 32 Bit Datenwort. Beispielsweise für SEL _ O = 1111 sind alle 32 Bit aktiviert, für SEL _ I = 0011 nur die unteren 16 Bit.
STB _ O, STB _ I	STB zeigt einen gültigen Datentransferzyklus an. Wird vom Master gesetzt, um anzuzeigen, dass eine gültige Adresse auf dem Adressbus liegt.
ACK _ I, ACK _ O	ACK markiert das Ende eines Busuyklus durch einen Slave, also Write complete oder Read-Daten vorhanden.
CYC _ O, CYC _ I	Definiert den Beginn und das Ende eines Buszyklus. Der Master setzt das Signal um einen neuen Bustransfer zu initiieren
TAGN _ O, TAGN _ I	Der Tag-Bus wird in diesem Projekt nicht genutzt

Tabelle 1: Signale des Wishbone-Bus

Keine Lust mehr auf kleine Jobs bei großen Firmen?

Sie wollten nach Ihrem Abschluss unbedingt bei einem Großunternehmen arbeiten? Nun sind Sie als einer unter vielen nur für wenige Details zuständig. Starre Verantwortlichkeiten lassen Ihnen nicht die Freiräume, die Sie von Ihrer Position erwarten. Es wird Zeit über einen Wechsel zu Bürkert nachzudenken!

Bürkert ist mit 2.100 Mitarbeitern in 35 Ländern eines der weltweit führenden Unternehmen für Mess-, Steuer-, und Regeltechnik. Gleichzeitig sind wir ein Familienunternehmen mit starken Wurzeln und Werten. Eine seltene Kombination aus Globalität und starken Traditionen. Sie macht die Faszination Bürkert aus.

Gehen Sie bei uns Ihre eigenen Wege und definieren Sie Ihren Aufgabenbereich selbst. Wir bieten Ihnen echte, vielfältige Fach- und Führungspositionen, z. B. als

- **Entwicklungsingenieur (m/w)**
  Software oder Elektronik-Hardware

  Standort: Karlsruhe

- **Projektleiter (m/w) Elektronikentwicklung**
  Embedded Systems / Displayentwicklung

  Standort: Ingelfingen

Wachsen Sie mit uns an verantwortungsvollen Aufgaben – in einem zukunftssicheren Familienbetrieb, in einer Region, in der auch Ihre Familie wachsen kann.

Haben wir Sie überzeugt? Dann überzeugen Sie uns mit Ihrer Bewerbung.

Bürkert Fluid Control Systems
Human Resources | Simone Kuner
Christian-Bürkert-Straße 13-17
74653 Ingelfingen
Telefon 07940/10-91212
simone.kuner@buerkert.com
www.buerkert.com

Zertifikat seit 2010
audit berufundfamilie

bürkert
FLUID CONTROL SYSTEMS

Experimenteller Laser-Pointer-Fernseher

Wenn man den Raum abdunkelt, kann man fernsehen

Georg Zimmermann < Zimmermann@Biologie.Uni-Osnabrueck.de >

Meine Motivation

Einen „Prinzip-Fernseher“ wollte ich schon immer einmal konstruieren. Reizvoll daran ist, dass unter anderem Aspekte der Optik, Elektronik und Mechanik zusammenfließen. Nicht das perfekte, sondern ein rein zum Experimentieren gedachtes Gerät war hier das Ziel - auch volle Auflösung kein Muss. Der Fernseher sollte mit Transistoren aufgebaut sein, keine teuren Komponenten benötigen und ohne Vakuum, Hochspannung, Pixel-Matrix auskommen. LASER-Pointer sind mittlerweile recht billig geworden. Sollte da nicht etwas mit bewegten Spiegeln zu machen sein? Eine Internet-Suche bringt Beispiele realisierter Bauformen zutage - z.B. unter dem Begriff „mechanisches Fernsehen“: Spiegelrad Weiller (1889), Spiegelschraube Okolicsanyi (1927), Spiegelkranz Mihaly (1933), Schneider TV (1993).

Verwendete Teile aus dem Computerbereich:

1. Diffusionsfolie aus einem Notebook-Display
2. Motor mit Hexagonspiegel aus einer Drucker-Laser-Einheit
3. Schrittmotor aus einem Drucker
4. Festplattenmotor als Lager und Sensor
5. Antriebslager aus Floppylaufwerk
6. LASER-Pointer

Das Ergebnis

Das eigentliche Display besteht aus einer Diffusionsfolie aus einem Notebook-Display. Der Lichtstrahl eines LASER-Pointers erzeugt das Bild, indem dieser mit der Helligkeitsinformation aus dem Videosignal moduliert und durch bewegte Spiegel vertikal und horizontal abgelenkt wird (Abb. 2). Dies ist vergleichbar mit der Bilderzeugung beim Röhrenfernseher durch Modulation und Ablenkung des Elektronenstrahls.

Abb. 1: Das Ergebnis: Breite x Höhe = ca. (9cm x 7cm)

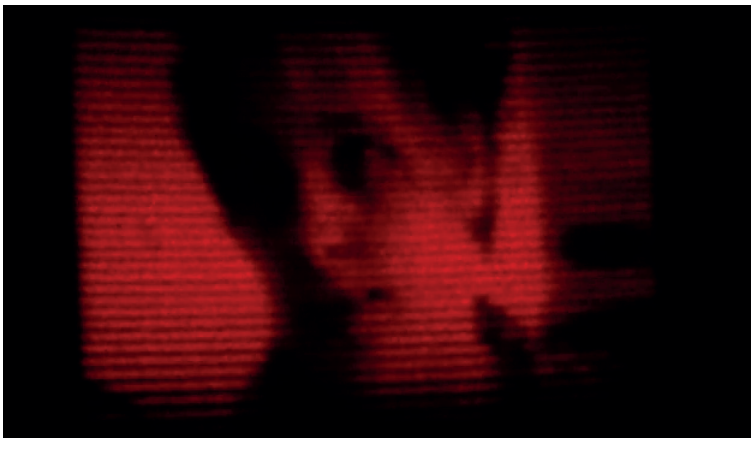
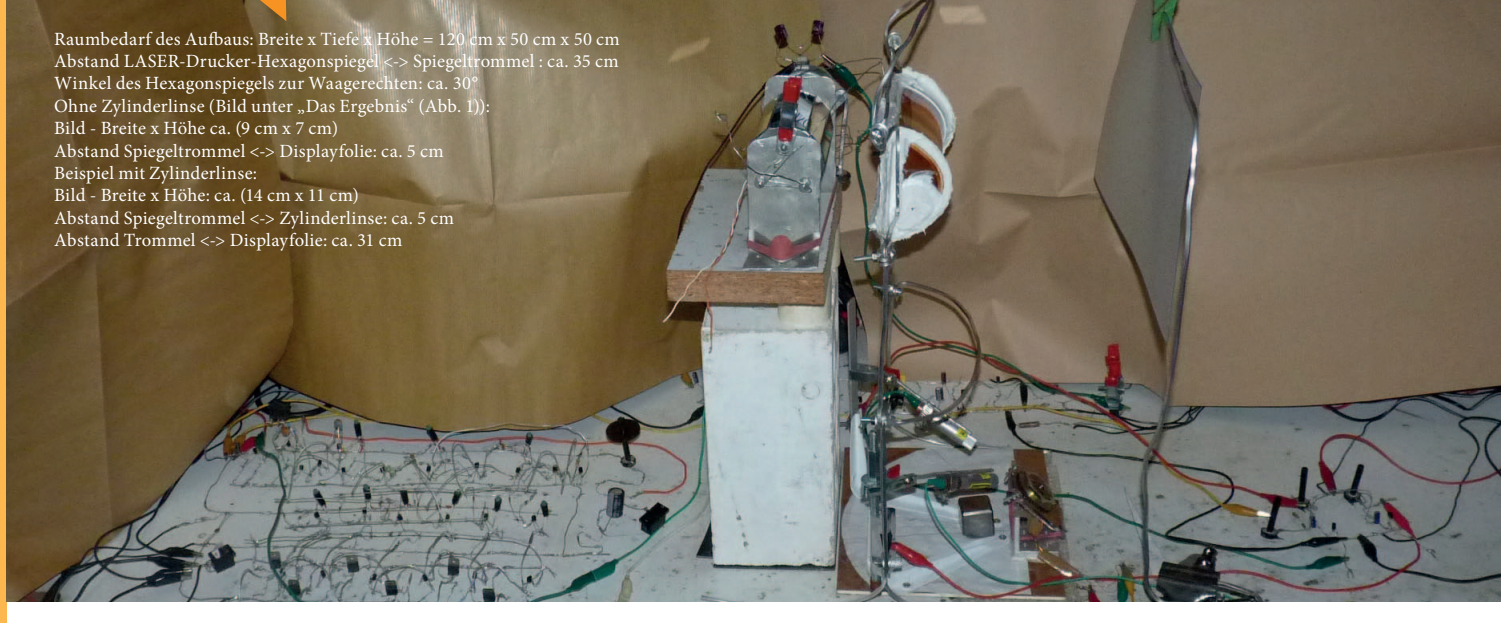


Abb. 2: Der Gesamtaufbau



To the Roots - Anschalten, Hochfahren und Zuschalten des Synchronsignals

Nach dem Anschalten läuft der Motor für die vertikale Ablenkung an und geht selbstständig in den Synchronlauf (250 RPM). Der Motor für die horizontale Ablenkung läuft kurz nach dem Einschalten mit einer konstanten, geringen Drehzahl - wenn nicht, hilft gefühlvolles Anstoßen. Dann ist er zum Hochfahren bereit. Durch das Schließen eines weiteren Schalters wird der Motor für die horizontale Ablenkung in die Nähe des Fangbereichs hochgefahren. Die Geschwindigkeitszunahme (Laden eines Kondensators) ist so bemessen, dass der Motor dem immer schneller werdenden Drehfeld noch folgen kann. Wenn der Motor warmgelaufen ist, dreht man ihn mit einem Potentiometer in den Fangbereich. Dann kann das Horizontal-Synchronsignal zugeschaltet werden. Der Motor schwingt sich ein (Drehzahl = 19531.25 RPM), und das Bild kommt zum Stehen (Wie ruhig das Bild ist, hängt unter anderem davon ab, ob der Motor die Betriebstemperatur erreicht hat, wie gut die Controllerfrequenz im Synchron-Bereich liegt, u.a. ...).

Let's do Retronik (Retro-Elektronik)

Alle Schaltungen sind diskret realisiert (nur Transistoren, Dioden, Widerstände, Kondensatoren und Spulen). Auch sind diese nicht mit Platinen aufgebaut, sondern die einzelnen Bauteile sind mit verzinktem Draht - dieser ist gut lötbar - verbunden, und zwar so, dass jede Schaltung einem gezeichneten Schaltbild sehr nahe kommt. Die Absicht: Stromverläufe bzw. Spannungszusammenhänge können so sehr gut überschaut und auch einzelne Bauteile leicht ab- und angelötet werden. Es geht eben nicht um Perfektion, sondern ausdrücklich um die größtmögliche Bastelfreiheit. Eine weitere Herausforderung war, für jede der hier enthaltenen Aufgabenstellungen die einfachste noch funktionierende Lösung zu finden - mit so wenig Bauteilen wie irgend möglich. Dadurch kommt jede Schaltung jeweils auch einer Prinzipschaltung so nahe, wie es nur geht, was dem Verständnis sehr und der Experimentierfreudigkeit sowieso entgegenkommt. Dementsprechend wurden auch nur zwei Transistortypen verwendet, die Komplementärpaare BC547B/BC557B und TIP120/TIP125.

Die Komponenten

Billig-LASER-Pointer (rot), Befestigung dreh- sowie höhenverstellbar; 1 LASER-Pointer -> 1/8 Auflösung vertikal (Abb. 3).

Optional: Erhöhung der Auflösung und Intensität durch Verwendung mehrerer LASER-Pointer, im 120° -Kreisbogen je um 15° versetzt:

- 2 LASER-Pointer -> 1/4 Auflösung vertikal
- 8 LASER-Pointer -> volle Auflösung vertikal

Die einzelnen LASER-Pointer so zu justieren, dass alle Zeilen richtig liegen, erfordert allerdings viel „Fingerspitzengefühl". Eine entsprechende feinmechanische Unterstützung zu konstruieren ist eine interessante Aufgabe schon für sich. Für den Anfang empfiehlt sich jedenfalls erstmal das Handling mit einem LASER-Pointer…

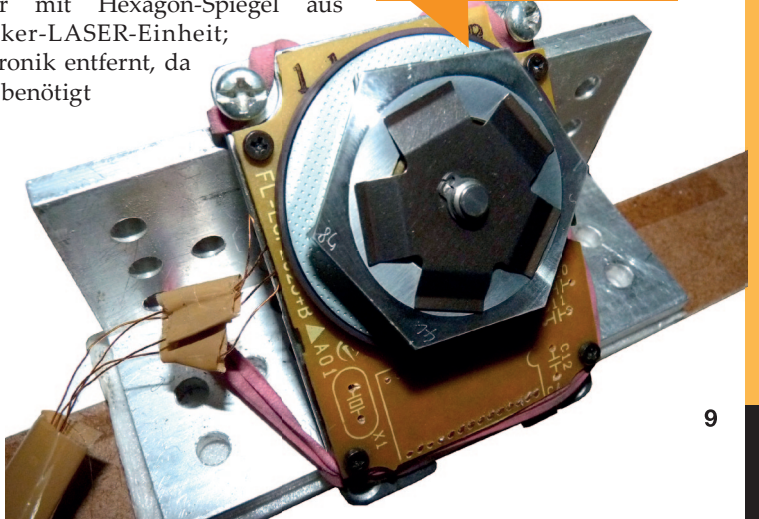
Abb. 3: Laserpointer

- Video-Signal: gelbe Leitung links
- Kontrasteinstellung: Poti Mitte links
- Helligkeitssteller für zwei LASERPointer

Abb. 4: Videosignal-Verstärker

- Motor mit Hexagon-Spiegel aus Drucker-LASER-Einheit; Elektronik entfernt, da nicht benötigt

Abb. 5: Horizontalablenkung - Motor und Spiegel



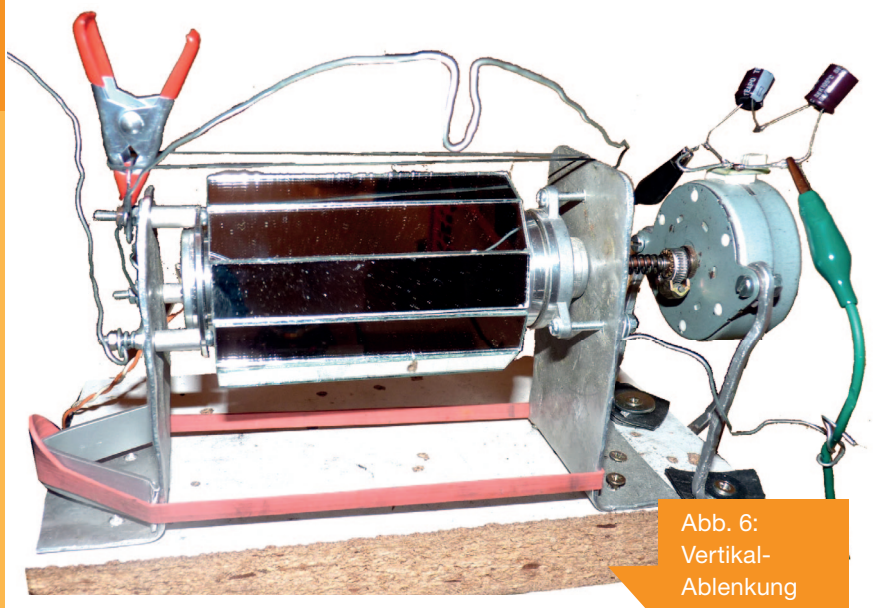


Abb. 6: Vertikal-Ablenkung - Spiegel

- Trommel mit 12 Spiegeln (Eigenbau, Abb. 6)
- Länge 9 cm, Durchmesser 6 cm
- Lagerung rechts: Antriebslager aus Floppylaufwerk
- Lagerung links: Festplattenmotor und gleichzeitig Sensor. Die Anschlüsse dieses Motors liefern, da er hier Generator spielt, Informationen über z.B. Drehzahl, Laufruhe oder die Synchronität mit dem Videosignal - am Oszilloskop schön anzuschauen.

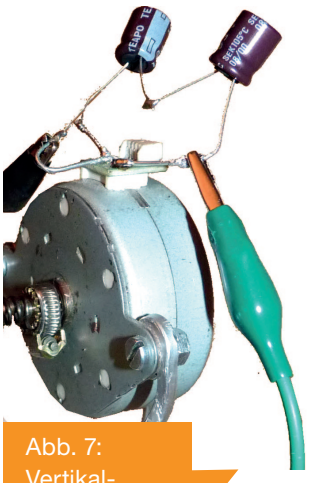


Abb. 7: Vertikal-Ablenkung - Antrieb

- Schrittmotor aus Drucker (Abb. 7)
- Mikroschrittbetrieb (Sin./Cos.) mit Phasenschieber
- Indirekte mechanische Kopplung mit Schraubenfeder

- (1) Horiz.-Sync.-Separator, Sync.-Zuschalter, Rechteckoszillator, Video-Signal: graue Leitung oben links
- (2) Dekoder
- (3) H-Brücke
- (4) Zuschaltung-Hochfahr-Kondensator und Drehzahl-Feintuning

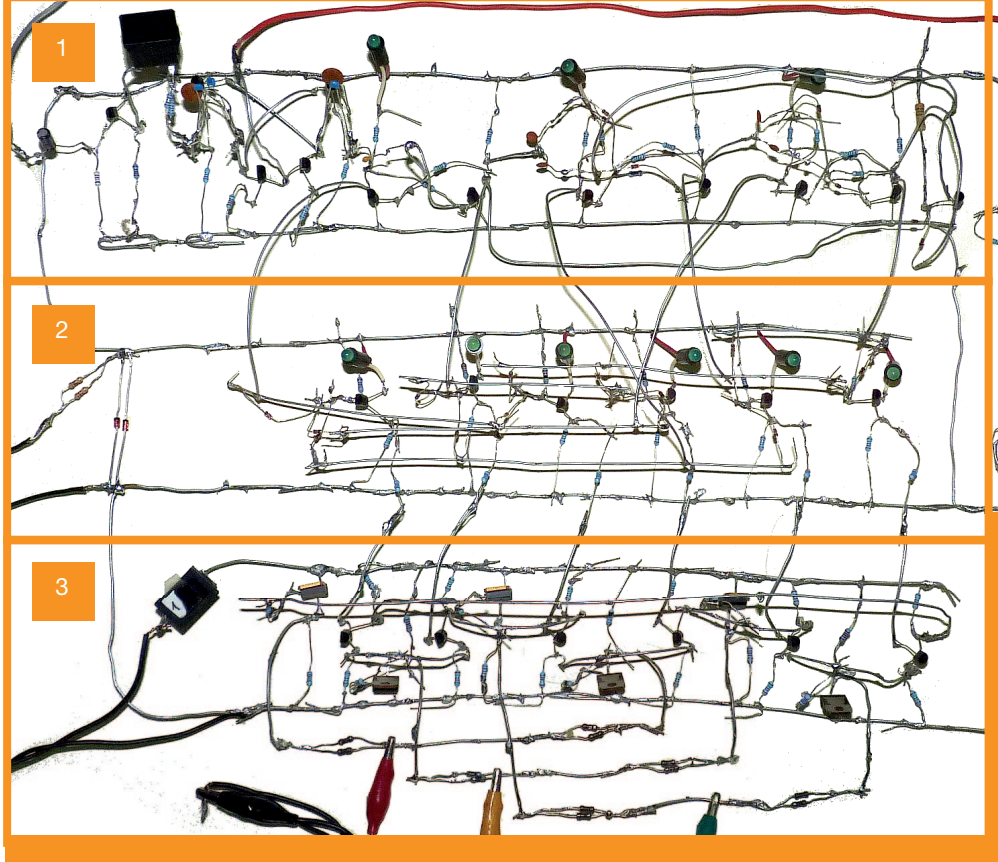


Abb. 8: Horizontal-Ablenkung - Controller

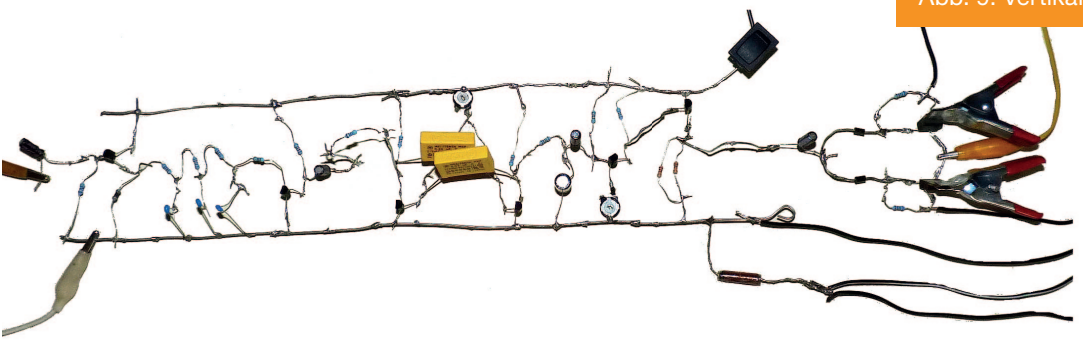


Abb. 9: Vertikalablenkung - Controller

- Video-Signal -> rote Klemme ganz links
- Vert.-Sync.-Separator, Rechteckoszillator, Impulsformer, Gegentaktverstärker

Literatur / Links

[1] Weiterführende Informationen direkt vom Autor: zimmermann@biologie.uni-osnabrueck.de

[2] Webseite im Aufbau: www.mightbemoving.de

[3] Umfassender Überblick über die Fernsehgeschichte: http://fernsehmuseum.info/geschichte01.html

Abb. 10: Wassergefüllte Zylinderlinse

Optional – Zylinderlinse für größeres Bild (Abb. 10)

Eine Abstandsvergrößerung zwischen Displayfolie und Trommel ergibt ein größeres Bild. Hierbei muss allerdings das Höhen- Breitenverhältnis durch eine Zylinderlinse angepasst werden. Da der LASER-Strahl durch die Zylinderlinse eine Konvergenz erfährt, entsteht für ihn hinter der Linse ein Focus. Der Maximalabstand (=maximale Bildgröße) liegt damit im Focus des LASER-Pointer-Strahls, in größerem Abstand wird das Bild unscharf. Beeinflussen lässt sich die Focuslage durch Verstellen des Kollimators im LASER-Pointer. Ein Kompromiss zwischen Bildschärfe und Bildgröße ergibt etwa eine 1.5-fache bis maximal doppelte Größe des Bildes unter „Das Ergebnis“.

- Wassergefüllte Zylinderlinse, Dicke ca. 4 cm, Höhe ca. 8 cm
- Plane Seite aus Plexiglas
- Gekrümmte Seite aus Radierfolie, Form per Näherungsrechnung ermittelt (Ellipsenform) -> Ausdruck als Vorlage,
- Aluminiumbänder handgebogen, eins innen, eins außen,
- Radierfolie zwischen den beiden Aluminiumbändern in Form gehalten, verschraubt
- Dichtungsmittel: Silikon

AVR Xmega Toolchain

Erfahrungsbericht und Zusammenfassung

Christian Rother

Dieser Artikel ist aus einer Zusammenfassung entstanden und stellt meine gesammelten Erfahrungen dar. Diese können dem einen oder anderen vielleicht als Hilfestellung dienen, sollte er sich dafür entscheiden, sich selbst eine Toolchain zu erstellen. Der Artikel stellt keinerlei Ansprüche auf Vollständigkeit.

AVR Xmega Toolchain

Um eigene Programme für den XMEGA unter Linux zu kompilieren sind ‚binutils', ‚avr-gcc' und ‚avr-libc' unerlässlich. Auch wird mit hoher Wahrscheinlichkeit die Software ‚avrdude' benötigt, um damit, in Kombination mit einem Programmiergerät, den XMEGA zu Programmieren. Diese Sammlung von Tools wird hier im Folgenden als Toolchain bezeichnet und dieser Artikel beschreibt, wie diese kompiliert werden können. Aber genug der einführenden Worte...

Die Vorbereitungsschritte

Wir legen im Benutzerverzeichnis einen Ordner Namens ‚toolchain' an und betreten diesen.

```
$ mkdir ~/toolchain

$ cd ~/toolchain
```

Die aktuellen Patches für die GNU-Toolchain holen wir uns nun von Atmel mit nachfolgendem Befehl.

```
$ wget http://distribute.atmel.no/tools/opensource/as5-beta/avr8-gnu-toolchain-3.2.0.255-source.zip
```

Wir können diese dann auch gleich mit ‚unzip' entpacken.

```
$ unzip avr8-gnu-toolchain-3.2.0.255-source.zip
```

Grundsätzlich waren das die Vorbereitungsschritte, die wir treffen mussten. Jetzt beginnen wir mit den einzelnen Tools unserer Bau-Kette.

```
$sudo apt-get install patch build-essential texinfo bison flex libusb-dev
```

GNU-Binutils

Den Anfang machen wir mit mit den GNU-Binutils. Die GNU-Binutils stellen eine Sammlung von Programmierwerkzeugen für die Manipulation von Objektcode in Objektdateien bereit. Wir Holen uns die Binutils 2.20.1, da für die höhere Version noch keine Patches bereitstehen.

```
[ $ cd ~/toolchain ]

$ wget ftp://ftp.gnu.org/gnu/binutils/binutils-2.20.1.tar.bz2

$ tar xvf binutils-2.20.1.tar.bz2

$ cd binutils-2.20.1
```

Die Atmel Patches können in richtiger Reihenfolge mit nachfolgender Kommandozeile angewendet werden:

```
$ find ../ -name ‚*binutils*.patch' | sort | while read X; do patch -p0 < $X; done
```

Dies sollten - von kleinen Offsets mal abgesehen - keine Probleme geben. So dass wir mit der Erstellung eines Objektverzeichnisses weitermachen können. Diese von den Binutils Programmierern empfohlene Bauweise soll unsere Source Dateien vor Verunreinigung schützen:

```
$ mkdir obj-avr

$ cd obj-avr
```

Nun Können wir ‚configure' anschmeißen, um anschließend mit einem ‚make' den Bauprozess zu starten.

```
$ ../configure --target=avr --prefix=$PREFIX --disable-nls

$ make
```

Wenn dies alles Funktioniert hat können wir nun die Dateien, im über ‚PREFIX' mitgegebenen Verzeichnis, installieren. Für ‚make install' benötigen wir Root Rechte, deshalb lautet der Befehl:

```
$ sudo make install
```

Der GNU-GCC Kompiler

Der nächste Schritt ist der GNU-GCC. Die Vorgehensweise ist nahezu Identisch mit der Vorgehensweise bei den Binutils. Hier sollte angemerkt werden, dass man sich einiges an Donwload- und Bauzeit sparen kann, indem man sich nicht das Gesamtpaket GCC sondern nur den GCC-Core herunterlädt. Dieser ist für die meisten Anwendungen vollkommen ausreichend. Hier wird nun aber im Folgenden vom Gesamtpaket ausgegangen. Die Schritte dürften sich nur marginal unterscheiden.

```
$ cd ~/toolchain

$ wget ftp://ftp.gnu.org/gnu/gcc/gcc-4.5.1/gcc-4.5.1.tar.
```

```
bz2

$ tar xvf gcc-4.5.1.tar.bz2

$ cd gcc-4.5.1
```

Um den Bau des Gesamtpakets zu ermöglichen benötigen wir noch ein Paar Bibliotheken, welche wir in dem GCC Unterverzeichnis des Toolchainordners platzieren können. Sie werden dann automatisch mit gebaut. Es auch möglich, diese von der Standard-Repository ihrer Distribution herunter zu laden. Ich ziehe es jedoch vor, die aktuellen Versionen selbst zu bauen. Zumal wir sie dazu lediglich innerhalb des GCC Verzeichnis platzieren müssen. Den Rest erledigt dann ‚configure' für uns.

GNU Multiple Precision Library (GMP)

Die GMP ist unsere erste Bibliothek, sie ermöglicht arithmetische Funktionen für beliebig große Zahlen und wird wie folgt heruntergeladen und entpackt:

```
$ wget ftp://ftp.gmplib.org/pub/gmp-4.3.2/gmp-4.3.2.tar.bz2
```

```
$ tar xvf gmp-4.3.2.tar.bz2

$ mv gmp-4.3.2 gmp
```

MPFR Library

Ist eine Bibliothek zum Rechnen mit Gleitkommazahlen beliebiger Genauigkeit. Die aktuelle Version wurde von http://www.mpfr.org/ heruntergeladen.

```
$ wget http://www.mpfr.org/mpfr-current/mpfr-3.0.1.tar.bz2
```

```
$ tar xvf mpfr-3.0.1.tar.bz2

$ mv mpfr-3.0.1 mpfr
```

MPC Library

Ist eine Bibliothek zum Rechnen mit Komplexen Zahlen. Nachstehende Kommandos sollten bekannt vorkommen ;) :

```
$ wget http://www.multiprecision.org/mpc/download/mpc-0.9.tar.gz

$ tar xvf mpc-0.9.tar.gz

$ mv mpc-0.9 mpc
```

Nun zurück zum GCC, die Atmel Patches werden mit nachstehender Kommandozeile angewandt:

```
$ find ../ -name „*gcc*.patch" | sort | while read X;do patch -p0 < $X;done
```

Dann benötigen wir noch einen Patch, da sich im gcc 4.5.1 ein Fehler befindet. Dieser Bug trägt die Nummer: 45261. Der Patch kann unter http://gcc.gnu.org/bugzilla/show_bug.cgi?id=45261 heruntergeladen werden:

```
[ $ cp ~/Downloads/gcc_45261_fix_2.patch . ]
```

Je nach Speicherort, hier im GCC Verzeichnis, wird er dann wie beschrieben angewendet:

```
$ patch -p1 < gcc_45261_fix_2.patch
```

Dies sollte abgesehen von einigen Offsets funktioniert haben, so dass wir dann mit dem Kompilieren beginnen können.

Auch beim GCC erstellen wir vorerst ein Obj-Verzeichnis:

```
$ mkdir obj-avr

$ cd obj-avr
```

Auch im nächsten Punkt scheint ein Fehler enthalten zu sein, denn anders als bei anderen Versionen, hat dieser GCC während des kompilieren Probleme die GMP Bibliothek zu finden. Wir helfen also über die Parameter „--with-gmp-include=$(pwd)/gmp" und „--with-gmp-lib=$(pwd)/gmp/.libs" nach.

Wieder übergeben wir ‚configure' „/usr/local/avr" durch eine ‚PREFIX' Variable, oder eben händisch. Dann bestimmen wir mit „--target=avr" unser Zielsystem.

Anschließend deaktivieren wir dann noch den ‚Native Language Support (NLS)' und die ‚Laufzeitbibliotheken für ‚Stack-Smashing-Protection (SSP)', da diese nicht benötigt werden.

```
$ ../configure --prefix=$PREFIX --target=avr --enable-languages="c,c++" --disable-nls --disable-libssp --with-gmp-include=$(pwd)/gmp --with-gmp-lib=$(pwd)/gmp/.libs
```

Nun können wir den Bauprozess starten:

```
$ make
```

Für das Installieren des ‚GCC' benötigen wir wieder Root Rechte, weshalb wir uns nachfolgender Kommandozeile bedienen:

```
$ sudo make install
```

AVR LIBC

Nun ich denke, die Vorgehensweise ist bekannt. Wir holen uns zuerst die neueste Version, entpacken diese, betreten das Verzeichnis:

```
$ cd ~/toolchain
```

```
$ wget http://download.savannah.gnu.org/releases/avr-libc/avr-libc-1.7.1.tar.bz2

$ tar xvf avr-libc-1.7.1.tar.bz2
```

```
$ cd avr-libc-1.7.1
```

Als nächster Schritt würde nun das Anwenden der von Atmel zur Verfügung gestellten ‚avr-libc' Patches an der Reihe sein. Seltsamerweise führt dies jedoch zu

Kompilerfehlern. Bei Recherche auf Themen bezogenen Mailinglisten habe ich herausgefunden, dass die Patches nicht benötigt werden und deshalb getrost weggelassen werden können. Bei dieser Recherche wurde ich auch auf einen dringlicheren Patch hingewiesen. Nachzulesen auf https://savannah.nongnu.org/bugs/?32698. Angewandt direkt auf die Zieldatei, deshalb mit der Kommandozeile:

```
$ cp ~/Downloads/power.patch
.

$ patch include/avr/power.h <
power.patch
```

Um auf die richtigen Bibliotheken zuzugreifen erneuern wir die Links und den Cache zu den Libs mit:

```
$ sudo ldconfig
```

Nun zurück zur Routine. Objektverzeichnis kreieren, betreten und ‚configure' starten:

```
$ mkdir obj-avr

$ cd obj-avr

$ ../configure --prefix=$PREFIX
--build=`./config.guess`
--host=avr
```

Anschließend den Bauprozess an stupsen mit ‚make'.

```
$ make
```

Beim Anschließenden ‚make install' gab es nun erneut einen Fehler. Anscheinend unterschied sich die PATH Variable bei Ausführung mit sudo von der ohne Root Rechten. Nachfolgend ist der Lösungsweg beschrieben, wie ich das Problem umgehen konnte.

```
$ sudo -s -H

# export PATH=$PATH:/usr/lo-
cal/avr/bin

# make install

# exit
```

AVRDUDE

Erheblich Ereignisloser gestaltete sich der Bauvorgang des ‚avrdude'. Die unten genannten Befehle sollten keiner weiteren Erklärung bedürfen:

```
$ cd ~/toolchain
```

```
$ wget http://download.savan-
nah.gnu.org/releases/avrdude/
avrdude-5.10.tar.gz

$ tar xvf avrdude-5.10.tar.gz

$ cd avrdude-5.10
```

```
$ ./configure --prefix=$PREFIX

$ make

$ sudo make install
```

C auf XMEGA

Noch ein paar Worte zu C und XMEGA. Das Ansprechen der Register des XMEGA unterscheidet sich von dem früherer Controller der Atmega Serie. Guter Rat allerdings, ist noch relativ schwer zu finden. Natürlich dient die ‚Documents' Spalte der Atmel Hompage als erste und unerlässliche Anlaufstelle.

Zu finden ist diese unter der Rubrik „Products ◎ Microcontroller ◎ Atmel AVR 8- and 32-bit ◎ AVR XMEGA ◎ Documents". Sie bieten einige Anwendungsbeschreibungen und Codebeispiele bestimmter XMEGA Funktionen. Der Code liest sich aber relativ zäh, wenn man wie ich lediglich beispielbezogene Codestücke sucht, die einem die Änderungen der Registeransprache näher bringen.

Als besonders erfreulich fand ich demzufolge die Zusammenfassung unter dem Link: http://www.stromflo.de/dokuwiki/doku.php?id=xmega-c-tutorial. Sie stellt eine kompakte Zusammenstellung dar, die gut als Nachschlagewerk für die ersten Gehversuche dienen kann. Wenn auch das Kapitel USART bei Entstehung dieses Artikels fehlte.

Die Dokumente „Xmega A1 Device Datasheet" mit dem schlichten Namen „doc8067.pdf" (http://www.atmel.com/dyn/resources/prod_documents/doc8067.pdf)

Und vor allem das Dokument „Xmega A Manual" mit dem Namen „doc8077.pdf" (http://www.atmel.com/dyn/resources/prod_documents/doc8067.pdf) dürften wohl als Nachschlagewerk jedes Programmierers dienen.

Das gute alte Blinklicht ...

Besonders schlicht und deshalb idealer Einstiegspunkt. Die ‚includes' beließ ich bei der ersten Demo ganz klassisch.

```
#include <avr/io.h>

#define F _ CPU 32000000UL

#include <util/delay.h>
```

Die Konfiguration des Oszillator stellte dann die erste Herausvorderung dar, da sich doch einiges geändert hat.

```
int main(void)

{

      //Oszillator auf 32Mhz stellen
```

```
	OSC.CTRL |= OSC_RC32MEN_bm;

	// Warten bis der Oszillator bereit ist

	while(!(OSC.STATUS & OSC_RC32MRDY_bm));
```

Mit Hilfe des ‚Configuration Change Protection Registers' schützt Atmel den XMEGA vor unerwünschter Fehlkonfiguration. Ein Schreiben von ‚0xD8' entfernt für 4 Taktzyklen den ‚IO Protection' Schreibschutz, dann übernehmen wir unsere Änderungen mit Hilfe des ‚Clock Control Registers'.

```
	//Configuration Change Protection Register

	CCP = 0xD8;

	//Select 32Mhz OSC

	CLK.CTRL = (CLK.CTRL & ~CLK_SCLKSEL_gm) | CLK_SCLKSEL_RC32M_gc;
```

Letztendlich stellen wir das Register, an dem unser Blinklicht angeschlossen ist, auf Ausgang und lassen die LED in einer Endlosschleife Blinken.

```
	PORTA.DIR = 0xFF;

	while (1){

		PORTA.OUTTGL = 0xFF;

		_delay_ms(100);

	}

}
```

Literatur / Links

[1] Atmel Patch für Toolchain: http://distribute.atmel.no/tools/opensource/as5-beta/avr8-gnu-toolchain-3.2.0.255-source.zip

[2] GCC: ftp://ftp.gnu.org/gnu/gcc/gcc-4.5.1/ gcc-4.5.1.tar.bz2

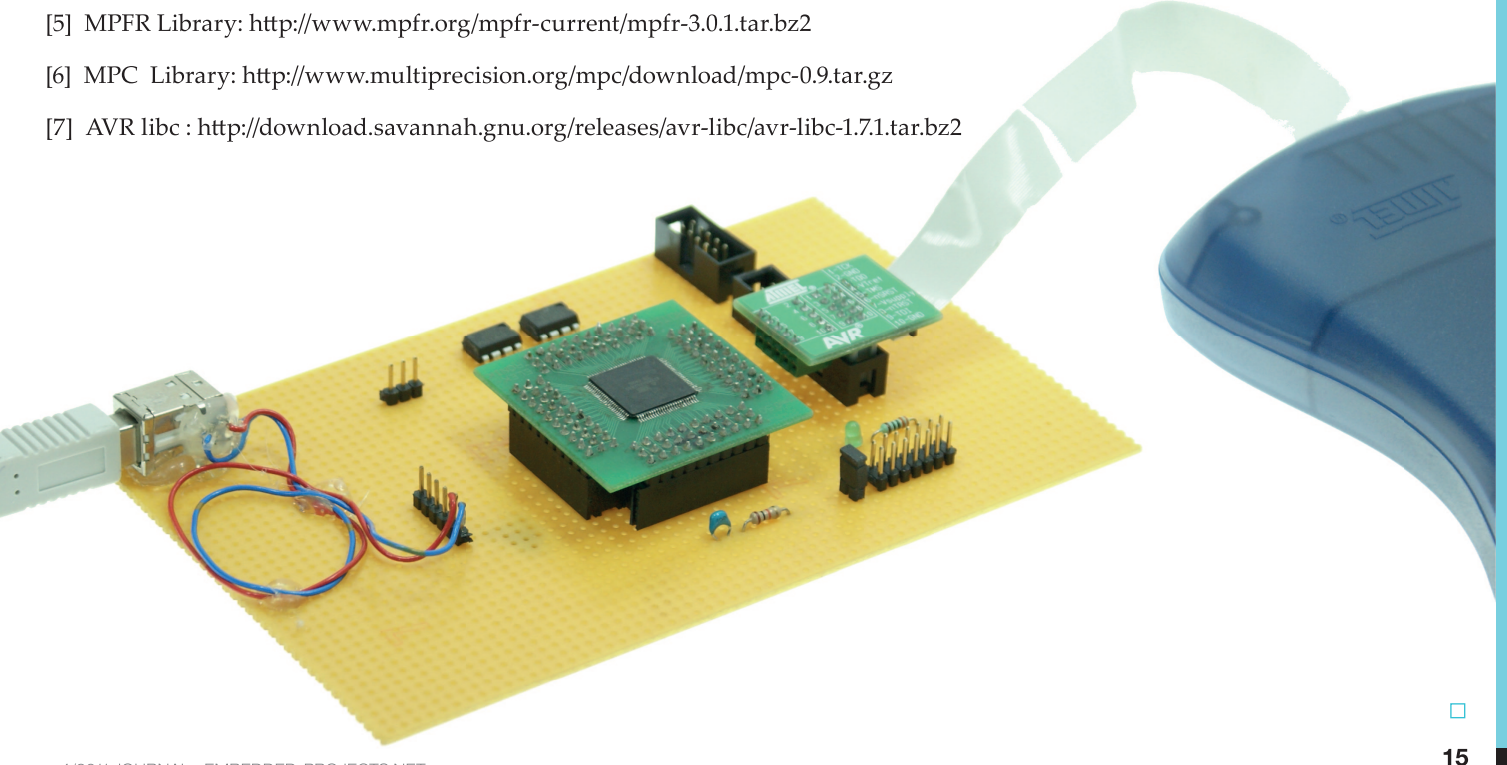
[3] Binutils: ftp://ftp.gnu.org/gnu/binutils/binutils-2.20.1.tar.bz2

[4] GMP Library: ftp://ftp.gmplib.org/pub/gmp-4.3.2/gmp-4.3.2.tar.bz2

[5] MPFR Library: http://www.mpfr.org/mpfr-current/mpfr-3.0.1.tar.bz2

[6] MPC Library: http://www.multiprecision.org/mpc/download/mpc-0.9.tar.gz

[7] AVR libc : http://download.savannah.gnu.org/releases/avr-libc/avr-libc-1.7.1.tar.bz2



AVR-Transistortester

Markus Frejek

Grundsätzliches

Jeder Bastler kennt wohl dieses Problem: Man baut einen Transistor aus einem Gerät aus oder kramt einen aus der Bastelkiste. Wenn die Typenbezeichnung zu erkennen ist (und sie einem bekannt ist), ist ja alles in Ordnung. Oft ist dem aber nicht so, und dann geht das Übliche mal wieder los: Typ und Pinbelegung des Transistors aus dem Internet oder einem Buch heraussuchen. Das nervt auf Dauer natürlich ziemlich. Auch Transistoren im gleichen Gehäuse haben nicht immer die gleiche Pin-Belegung. Zum Beispiel hat ein 2N5551 eine andere Belegung als ein BC547, obwohl beide ein TO92 Gehäuse haben. Wenn bei einem Transistor die Bezeichnung nicht mehr zu erkennen ist (oder nicht mal Google was dazu weiß), wird es mit konventionellen Methoden richtig umständlich, den Transistortypen herauszufinden: Es könnte sich um NPN, PNP, N-oder P-Kanal-Mosfet, etc. handeln. Eventuell hat das Bauteil auch noch eine Schutzdiode intern. Das macht die Identifizierung noch schwieriger. Durchprobieren per Hand ist also ein recht zeitraubendes Unterfangen.

Warum soll man das nicht per Mikrocontroller erledigen lassen? So ist dieser automatische Transistortester entstanden.

Features

1) Automatische Erkennung von NPN und PNP-Transistoren, N-und P-Kanal-MOSFETs, Dioden (auch Doppeldioden), Thyristoren, Triacs und auch Widerständen und Kondensatoren.
2) Automatische Ermittlung und Anzeige der Pins des zu testenden Bauteils.
3) Erkennung und Anzeige von Schutzdioden bei Transistoren und MOSFETs.
4) Ermittlung des Verstärkungsfaktors und der Basis-Emitter-Durchlassspannung bei Transistoren.
5) Messung der Gate-Schwellspannung und Gatekapazität von Mosfets.
6) Anzeige der Werte auf einem Text-LCD (2x16 Zeichen).
7) Dauer eines Bauteil-Tests: Unter 2 Sekunden (Ausnahme: größere Kondensatoren).
8) Ein-Knopf-Bedienung; automatische Abschaltung.
9) Stromverbrauch im ausgeschalteten Zustand: < 20 nA .

Selbstleitende FETs (z. B. JFETs) werden mittlerweile auch unterstützt. Bei Leistungs-Thyristoren und -Traics kann es auch zu Problemen kommen, wenn der Teststrom (ca. 7 mA) unter dem Haltestrom liegt. MOSFETs und Transistoren wurden aber in meinen Tests immer korrekt erkannt.

Der Messbereich für Widerstände liegt bei etwa 2 Ohm bis 20M Ohm, deckt also die meisten Widerstandswerte ab. Die Genauigkeit ist aber nicht sehr hoch.

Kondensatoren können von ca. 0,2nF bis gut 7000µF gemessen werden. Oberhalb von rund 4000µF wird die Genauigkeit aber zunehmend schlechter. Prinzipbedingt dauert die Messung großer Kondensatoren auch recht lange, eine Messdauer bis zu einer Minute ist normal.

Hardware

Es empfiehlt sich, den Schaltplan in einem weiteren Browserfenster zu öffnen, um die folgenden Beschreibungen nachvollziehen zu können.

Als Mikrocontroller wurde der ATMega8 gewählt. Er verfügt über mehr als genug Flash und RAM. Außerdem hat er genug Portpins und ist sehr preisgünstig. Der Transistortester wird mit einer 9V-Block-Batterie betrieben. Die 5V-Betriebsspannung für den AVR wird ganz konventionell mit einem 78L05 erzeugt. Am Port B des ATMega8 sind verschiedene Widerstände angeschlossen: Je Transistor-Pin ein großer (470 k) und ein kleiner (680 Ohm). Hiermit können zwei verschiedene Stromstärken auf den zu testenden Pin gegeben werden. Die Widerstände sind mit ADC0 , ADC1 und ADC2 verbunden. An diese Pins wird auch der zu testende Transistor angeschlossen. Der linke Teil der Schaltung (mit den 3 Transistoren) ist für die automatische Abschaltung zuständig. Dazu später mehr. An den ersten Pins von Port D ist das LCD angeschlossen. Es handelt sich dabei um ein 2x16 Zeichen Text-LCD mit HD44780-kompatiblem Controller.

Es ist zu beachten, dass die Test-Eingänge keine Schutzbeschaltung haben. Eine Schutzbeschaltung würde vermutlich auch die Messergebnisse verfälschen. Es sollten also keinesfalls Bauteile, die noch in eine Schaltung eingebaut sind, getestet werden. Sonst könnte der ATMega8 beschädigt werden.

- Bezugsquelle für Platinen: IT-WNS (Doppelseitige Platine)
- Dateien zum Selbstätzen: (Artikel-Link, einseitige Platine)

Testablauf

Der Test des Transistors funktioniert nach einem einfachen aber effektiven Prinzip: Es werden zunächst mal alle sechs Kombinationsmöglichkeiten für die Pins durchprobiert. Hierfür werden die Pins entweder über den 680 Ohm Widerstand auf + oder fest auf Masse gelegt oder ganz freigelassen. Es ergeben sich folgende sechs Möglichkeiten:

	Zustand Pin 1	Zustand Pin 2	Zustand Pin 3
1. Möglichkeit	+	-	frei
2. Möglichkeit	+	frei	-
3. Möglichkeit	frei	-	+
4. Möglichkeit	frei	+	-
5. Möglichkeit	-	frei	+
6. Möglichkeit	-	+	frei

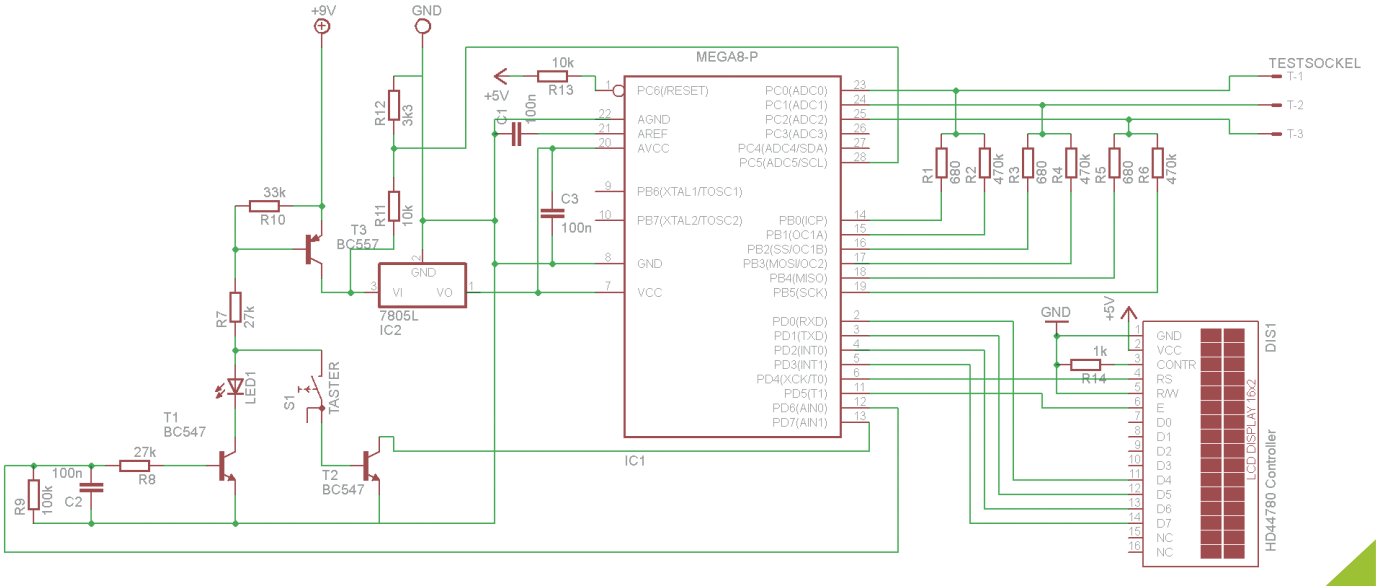


Abb. 1: Schaltplan mit Abschaltung

... zu den Messungen

Für jede Kombinationsmöglichkeit wird überprüft, ob zwischen dem + Pin und dem -Pin Durchgang herrscht. Hierfür wird mit den ADCs die Spannung am + Pin ermittelt. Falls Durchgang besteht und die abfallende Spannung im Bereich zwischen 0,2 V und 4 V liegt, wird von einer Diode zwischen den betreffenden Pins ausgegangen. Dieses Ergebnis wird gespeichert. Der Test wird dadurch natürlich nicht abgebrochen, da dieses Ergebnis auch bei einem Transistor eintreten kann und wird (ein Transistor hat zwei pn-Übergänge, also 2 Dioden).

Falls kein Durchgang herrscht, wird der bisher frei gelassene Pin über 680 Ohm auf Masse gelegt. Falls jetzt Durchgang besteht, muss es sich um einen pnp-Transistor oder p-Kanal-MOSFET handeln. In diesem Fall wird der Pin (es handelt sich dann um die Basis) über den 470 k Widerstand auf Masse gelegt. Jetzt wird die Spannung zwischen dem +Pin und -Pin (Kollektor und Emitter) gemessen und zwischengespeichert. Daraus kann nachher der Verstärkungsfaktor und die „richtige" Anschlussbelegung errechnet werden. Ein Transistor funktioniert nämlich auch „falsch herum" (also bei einem pnp-Transistor Kollektor auf Plus), allerdings ist der Verstärkungsfaktor deutlich geringer. Wenn hingegen immer noch kein Durchgang zwischen dem positiven und negativen Pin herrscht, wird der anfangs frei gelassene Pin über 680 Ohm auf Plus gelegt. Besteht nun Durchgang, handelt es sich um einen npn-Transistor, einen n-Kanal-MOSFET oder einen Thyristor/Triac. Der weitere Ablauf entspricht dem bei pnp. Der anfangs frei gelassene Pin wird dann allerdings nochmal hochohmig geschaltet. Wenn das Bauteil dann immer noch leitet, ist es ein Thyristor oder Triac.

Durch die 680 Ohm-Widerstände ist der mögliche Strom allerdings recht gering (max. ca. 7,4 mA). Wenn der Haltestrom des Thyristors oder Triacs über diesem Wert liegt, wird das Bauteil als npn-Transistor erkannt. Das dürfte bei vielen Leistungsthyristoren oder -triacs der Fall sein. Eine Verkleinerung der Testwiderstände (für mehr Strom) wäre zwar möglich, erhöht aber auch das Risiko, sehr empfindliche Transistoren zu zerstören.

Die Unterscheidung zwischen Transistor und MOSFET ist nicht schwer: Bei einem Transistor fließt Basisstrom. Somit wird der Basis-Anschluss „zum Emitter hin" gezogen. Bei einem MOSFET passiert das natürlich nicht. Das lässt sich dann leicht über den angeschlossenen ADC ermitteln.

Zur Messung der Gate-Schwellspannung von N-Kanal-Anreicherungs-Mosfets wird Source fest auf Masse gelegt. Drain wird über einen 680-Ohm-Widerstand gegen Plus gelegt und Gate wird über einen 470kOhm Widerstand gegen Plus gelegt. Dann wartet der Tester, bis der Mosfet schaltet, also der Drain-Anschluss auf logisch 0 geht. Nun wird die am Gate anliegende Spannung gemessen. Das ist etwa die Gate-Schwellspannung.

Zur dann folgenden Messung der Gatekapazität werden Gate, Drain und Source fest gegen Masse gelegt. Das entlädt das Gate komplett. Dann wird das Gate über 470kOhm gegen Plus gelegt, Drain und Source bleiben auf Masse. Nun wird in einer Schleife die Zeit gemessen, die vergeht bis der Gate-Anschluss auf logisch 1 geht. Das passiert bei recht genau 3,6V, wenn der AVR mit stabilen 5V versorgt wird. Das sind 72% der Betriebsspannung. Somit ergibt sich rechnerisch eine Zeitkonstante von 1,28 Tau (1,28 R*C). Da R mit 470kOhm bekannt ist, lässt sich daraus C, also die Gatekapazität, berechnen. Der reale Faktor für die Berechnung weicht etwas davon ab und kann per Define im Sourcecode angepasst werden.

Die Messung der Gate-Kapazität ist, wie auch die Messung der Gate-Schwellspannung, ohnehin nicht besonders genau. Für eine Abschätzung und den groben Vergleich verschiedener Mosfets sind diese Funktionen aber brauchbar. Der Ablauf der Gate-Schwellspannungs und -Kapazitätsmessung für P-Kanal-Anreicherungs-Mosfets ist identisch mit dem oben beschriebenen Ablauf für N-Kanal-Mosfets. Nur kehren sich alle Polatitäten um und die Zeitkonstante für die Gatekapazität ändert sich.

Für die Messung von Widerständen wird aus einem der internen Widerstände (680 Ohm oder 470kOhm) und dem Prüfling ein Spannungsteiler aufgebaut. Die Spannung am Mittelpunkt des Teilers wird gemessen, einmal mit dem 680Ohm und einmal mit dem 470kOhm-Widerstand. Daraus lässt sich der Widerstandswert des Prüflings berechnen.

Dieses Messverfahren hat das Problem, dass die Genauigkeit stark vom Widerstandswert des Prüflings abhängt.

Am genauesten ist die Messung, wenn der Prüfling etwa den gleichen Widerstandswert wie einer der internen Widerstände hat, also 680 Ohm oder 470kOhm. Dann hat der Spannungsteiler ein Verhältnis von etwa 1:1. Eine kleine Änderung des Widerstandswerts des Prüflings ändert hier die Spannung in der Mitte des Teilers recht stark.

Mit dem internen 680Ohm-Widerstand ergibt sich mit einem 600-Ohm-Prüfling eine Teilerspannung von 2,34V; mit einem 800-Ohm-Prüfling sind es 2,7V. Eine Erhöhung des Prüflings um 33% erhöht also die Spannung hier um immerhin 0,36V, erhöht also den ADC-Wert um 74.

Bei einem 6-Ohm-Prüfling dagegen sind es 0,044V in der Mitte des Teilers, bei 8 Ohm sind es 0,058V. Hier steigt der Widerstandswert ebenfalls um 33%. Die Spannung in der Teilermitte erhöht sich aber nur um 0,014V, was den ADC-Wert um gerade mal 2 bis 3 erhöht.

Übrigens wird jeder Prüflings-Widerstand sowohl mit dem 470kOhm-Widerstand als auch mit dem 680Ohm-Widerstand vermessen. In der Auswertung wird dann das genauere Ergebnis verwendet, also das Ergebnis, bei dem die Teilerspannung näher am Optimalwert von 2,5V liegt.

Der gesamte Messbereich liegt bei etwa 5 Ohm bis 910kOhm. Im Bereich unter 20 Ohm ist die Messung sehr ungenau, Abweichungen von 10-20% können schon vorkommen.

Nicht besonders genau ist auch der Bereich von ca. 20 bis 100 Ohm. Zudem ist auch der Bereich von ca. 10-50kOhm nicht richtig genau, weil dafür keiner der zur Verfügung stehenden Messwiderstände (680 Ohm oder 470kOhm) gut geeignet ist.

Wenn in dem Tester präzise Widerstände (Metallschicht mit <=1% Toleranz) verbaut werden, dürften +/-3% Abweichung über den gesamten Messbereich aber machbar sein. Damit ist die Widerstands-Messfunktion natürlich keine Konkurrenz zu einem guten Multimeter, aber durchaus passabel.

Die Messung von Kondensatoren ist von der Messung der restlichen Bauteile getrennt. Das bedeutet, dass sie eine eigenständige Messfunktion besitzt, die in allen 6 Pin-Kombinationen auf das Vorhandensein eines Kondensators prüft. Das ist nötig, weil sich die Messungen sonst gegenseitig stören würden.

Das Vorhandensein eines Kondensators wird folgendermaßen geprüft: Einer der beiden Pins wird fest auf Masse gelegt, der andere über 470kOhm auf Masse.

Nach einer kurzen Zeit wird die Spannung an dem Pin, der über 470kOhm auf Masse liegt gemessen und der Wert wird gespeichert. Nun wird dieser Pin für 10ms über 680 Ohm auf Plus gelegt. Nachher wird wieder die Spannung gemessen. Wenn sie um mindestens ca. 10mV gestiegen ist, wird davon ausgegangen, dass ein Kondensator vorhanden ist. Das legt auch die maximal messbare Kapazität von ca. 7350µF fest (680 Ohm lassen an 5V 7,35mA fließen, was einen Kondensator von 7350µF in 10ms um 10mV lädt).

Abb. 2: Transistortester im Gehäuse

Falls die Spannung nicht hinreichend angestiegen ist, wird der Test mit dem Ergebnis „Kein Kondensator“ abgebrochen. Ist die Spannung hingegen angestiegen, wird der Kondensator wieder vollständig entladen. Dann wird die Kapazität gemessen, genau nach dem gleichen Prinzip wie bei der Messung der Gatekapazität beschrieben.

Zunächst erfolgt die Messung mit dem 680 Ohm Widerstand. Falls die Zeit, bis der Pin umschaltet sehr gering ist (geringe Kapazität), wird mit dem 470kOhm Widerstand erneut gemessen. Falls auch da die Zeit, bis der Pin umschaltet sehr gering war, wird die Messung abgebrochen. Damit ist die untere Grenze des Messbereiches von ca. 0,2nF festgelegt. Das dürfte ein sinnvoller Wert sein, sonst können Kapazitäten des Prüfkabels o.ä. schon fälschlicherweise zur Erkennung eines (nicht vorhandenen) Kondensators führen.

War die Messzeit im vorgegebenen Bereich, wird der Kondensator als gefundenes Bauteil gespeichert. Die gemesene Zeit wird mit den (per Define festlegbaren) Faktoren in eine Kapazität umgerechnet, die dann in nF bzw. µF angezeigt wird. Auch hier ist die Genauigkeit nicht überragend hoch, bis zu 10% Abweichung sind möglich. Für die ungefähre Bestimmung der Kapazität ist es aber ausreichend.

Nachdem die genannten Messungen für alle sechs Kombinationsmöglichkeiten der Pins gemacht wurden, geht es an die Auswertung:

1) Bei Bipolartransistoren muss - wie gesagt, die richtige Belegung von Kollektor und Emitter anhand des Verstärkungsfaktors bestimmt werden

2) Wurden bei einem Bipolartransistor drei Dioden oder bei einem MOSFET eine Diode gefunden, besitzt das Bauteil eine Schutzdiode. Das wird durch ein kleines Dioden-Symbol rechts unten im LCD dargestellt

3) Wenn mehrere Dioden, aber kein Transistor erkannt wurden, wird die Stellung der Dioden zueinander überprüft, um den Bauteiltyp und die Pin-Belegung zu ermitteln (Doppeldiode mit Common Anode/Cathode, zwei Dioden in Serie, zwei Dioden antiparallel)

Das Ergebnis wird dann auf dem LCD dargestellt. Dann schaltet sich der Tester nach ca. 10 Sekunden automatisch ab.

Abb. 3: Ausführung als Einschub für einen 19“-Baugruppenträger, versorgt über einen Netztrafo, ohne Abschaltautomatik

Darstellung auf dem LCD

Das ermittelte Testergebnis wird auf dem LCD dargestellt. In der ersten Zeile links wird der erkannte Bauteiltyp angezeigt. Folgende Bauteile werden bis jetzt erkannt:

Je nach Bauteil werden noch weitere Daten angezeigt:

- Pin-Namen und Pin-Belegung
- Verstärkungsfaktor hFE und Basis-Emitter-Durchlassspannung (bei Bipolartransistoren)
- Gate-Schwellspannung und Gate-Kapazität(bei Anreicherungs-MOSFETs)
- Durchlassspannung (bisher nur bei einfachen Dioden, nicht bei Doppeldioden u.ä.)
- Bei Transistoren: Anzeige, ob eine Schutzdiode vorhanden ist (durch ein kleines Dioden-Symbol)
- Widerstandswert bei Widerständen und Kapazität bei Kondensatoren

Da bei den meisten JFETs Drain und Source gleichwertig sind, können diese Anschlüsse nicht erkannt werden. Es kann also vorkommen, dass Drain und Source bei JFETs vertauscht angezeigt werden.

Bauteil	Display-Anzeige
NPN-Transistor	NPN
PNP-Transistor	PNP
N-Kanal-Anreicherungs-MOSFET	N-E-MOS
P-Kanal-Anreicherungs-MOSFET	P-E-MOS
N-Kanal-Verarmungs-MOSFET	N-D-MOS
P-Kanal-Verarmungs-MOSFET	P-D-MOS
N-Kanal-JFET	N-JFET
P-Kanal-JFET	P-JFET
Thyristor	Thyristor
Triac	Triac
Doppeldiode, gemeinsame Anode	Doppeldiode CA
Doppeldiode, gemeinsame Kathode	Doppeldiode CC
2 antiparallele Dioden	2 Dioden
2 Dioden in Serie	2 Dioden
einfache Diode	Diode
Widerstand	Widerstand
Kondensator	Kondensator

Eine Unterscheidung zwischen bipolaren Kondensatoren und gepolten Elkos war ursprünglich geplant. Elkos entladen sich nämlich schneller, wenn sie in die falsche Richtung aufgeladen werden. Allerdings ist dieser Effekt bei Spannungen von 5V und Messzeiten von ein paar 100ms fast unmessbar gering, daher ist diese Erkennung nicht möglich.

Automatische Abschaltung

Am einfachsten wäre es natürlich, den AVR nach Abschluss des Tests einfach in den Power-Down-Mode zu versetzen und einfach per Taster beim nächsten Test wieder aufzuwecken. Der AVR braucht in diesem Modus unter 0,3 µA. Da würde die Batterie fast ewig halten. Da der Tester aber stabile 5 V braucht, kommt man um einen Spannungsregler nicht herum. Hier haben wir auch schon das Problem: Der Spannungsregler läuft munter weiter, auch wenn der AVR schläft, und verbraucht dabei gar nicht so wenig Strom: Ein 78L05 braucht ca. 3 mA. Das würde die Batterie in gerade mal einer Woche leeren. Selbst der sparsame LP2950 braucht noch 75 µA. Da würde die Batterie auch nur neun Monate halten. Schon besser, aber alles andere als ideal.

Es gibt aber noch eine bessere Möglichkeit: Den Strom mit einem pnp-Transistor in der Plus-Leitung (T3) schalten. Hiermit liegt der Standby-Strom bei gerade mal ~10 nA (0,01 µA oder 0,00001 mA). Eine 9 V Batterie mit 500 mAh wäre damit theoretisch erst nach über 5000 Jahren leer. Das sollte wohl reichen...

Beschreibung dieser Schaltung:

Im Ruhezustand wird die Basis von T3 über den Widerstand R10 auf Plus gelegt. Der Transistor ist damit gesperrt und die Schaltung nicht in Betrieb. Wird jetzt aber der Taster S1 gedrückt, wird die Basis von T3 über R7 und die BE-Strecke von T2 auf Masse gelegt. Damit leitet T3 und die Schaltung bekommt Betriebsspannung. Somit wird auch der AVR aktiv. Seine erste Aktion ist es, den Portpin PD6 (Pin 12) auf Plus zu legen. Damit bekommt der Transistor T1 Basisstrom und beginnt zu leiten. Dieser lässt nun über R7 und LED1 den Basisstrom von T3 fließen. Der Taster kann jetzt wieder losgelassen werden. 10 Sekunden nach Abschluss des Tests legt der AVR die Basis von T1 wieder auf Masse. Damit sperrt T1, T3 erhält keinen Basisstrom mehr, sperrt auch und schaltet die Betriebsspannung wieder ab. Der Transistortester ist damit wieder ausgeschaltet. Transistor T2 leitet, wenn der Taster gedrückt ist. Er ist an PD7 (Pin 13) an dem AVR angeschlossen. Damit merkt der Controller, wenn die Taste im Betrieb erneut gedrückt wird, und startet den Testzyklus neu. Das ist sinnvoll wenn man mehrere Transistoren hintereinander testen möchte. Sonst müsste man jedesmal warten, bis sich der Tester wieder automatisch abgeschaltet hat (ca. 10s). R9 und C2 sollen eventuelle Störungen unterdrücken. Sonst kann sich der Transistortester auch unerwünschterweise durch elektromagnetische Störungen einschalten. LED1 dient dazu, die Spannung an S1 auch im Betrieb hoch genug zu halten um T2 aufzusteuern: Wenn T1 geschaltet ist, liegt dessen Kollektor quasi auf Masse. Die Spannung am Kollektor reicht dann nicht mehr aus, um beim erneuten Drücken der Taste den Transistor T2 zu schalten. Anstelle der LED können auch zwei normale Dioden (1N4148 ö.ä.) in Reihe benutzt werden.

Die Widerstände R11 und R12 dienen zum Messen der Batteriespannung. Bei zu geringer Batteriespannung wird dann eine entsprechende Meldung („Batterie leer“) auf dem LCD angezeigt. (Abb. 5)

In dem Thread zum Artikel wurde mehrfach der Wunsch geäußert, den Transistortester ohne die automatische Abschaltung aufzubauen. Stattdessen wird dann ein einfacher Schalter verwendet. Das spart eine ganze Menge Bauteile ein. Die Software muss dafür nicht geändert werden. Taster S1 ist optional, wenn die automatische Abschaltung nicht eingebaut wird.. Damit kann man einen neuen Testzyklus starten, ohne den Tester dafür aus- und wieder einschalten zu müssen. Wer das nicht benötigt, kann ihn einfach weglassen. Achtung: Der Taster ist natürlich nur optional, wenn der Tester ohne die automatische Abschaltung aufgebaut wird!

Infos zur Software

In dem zum Download bereitstehenden Archiv ist eine fertig kompilierte Firmware-Version für den ATMega8 und eine kleinere für den ATMega48, sowie der komplette Quellcode enthalten. Die Version für den ATMega48 hat folgende Features nicht:

- Erkennung von Widerständen und Kondensatoren
- Messung der Gatekapazität von Anreicherungs-Mosfets
- Messung der Basis-Emitter-Spannung von Bipolartransistoren

Für diese Features bietet der ATMega48 einfach nicht genug Platz.

Für „Selbstkompilierer“: Um zwischen der Version für den ATMega8 und ATMega48 zu wechseln, einfach in den Projektoptionen oder im Makefile den AVR-Typen entsprechend ändern. Über #ifdef-Blocks werden dann automatisch die (vom Platz her) nur auf dem ATMega8 möglichen Features aktiviert bzw. deaktiviert. Es wird aber dringend davon abgeraten, den Tester mit dem ATmega48 aufzubauen: Dieser Controller ist kaum billiger als der ATMega8, und die Firmware für diesen wird kaum noch weiter gepflegt, weil sie (verständlicherweise) ohnehin kaum verwendet wird; außerdem bietet der Controller auch für Programm-Verbesserungen gar keinen Platz mehr.

Ich würde es schön finden, wenn ihr gefundene Fehler in der Software oder Verbesserungsvorschläge in den Forums-Thread zum Artikel schreibt.

Fehlersuche

Falls überhaupt nichts auf dem Display anzeigt wird, sollten folgende Dinge überprüft werden:

- Sind alle Verbindungen zum LCD richtig angeschlossen?
- Sind die Fusebits des ATMega8/ATMega48 richtig gesetzt (interner Oszillator mit 1MHz)?
- Wurde die .eep-Datei ins EEPROM

des Controllers geflasht?

- Hat das LCD einen HD44780-kompatiblem Controller?
- Handelt es sich möglicherweise um ein Hochtemperatur-LCD? Diese brauchen nämlich eine negative Kontrastspannung.
- Testweise auch mal R14 überbrücken. Dieser Widerstand muss ohnehin an das verwendete LCD angepasst werden, um einen guten Kontrast zu erhalten (ggf. ein Poti verwenden).

Sollte auf einem 2x16 Display nur eine Zeile als Klötzchen dargestellt werden und in der zweiten Zeile gar nichts, dann ist bei Anschluss mittels Adapterkabel die Verbindung komplett verdreht (Pin 1 der Platine also auf Pin 16 des LC-Displays). Es kann aber auch ein Fehler bei der Display-Initialisierung vorliegen (abweichend von HD44780). Das sind aber nur 2 von mehreren weiteren Möglichkeiten. Falls Bauteile nur mit einer bestimmten Reihenfolge der Anschlüsse an den Test-Pins richtig erkannt werden, ein Bauteil erkannt wird obwohl gar keins angeschlossen ist oder Daten wie der Verstärkungsfaktor bei verschiedenen Anschlussreihenfolgen stark voneinander abweichen, sollte die Platine auf Lötbrücken, schlechte Lötstellen o.ä. überprüft werden. Zwischen den Test-Pins sollten auch keine Flussmittelreste verbleiben. Flussmittel ist meist auch geringfügig leitfähig. Da die verwendeten Teileströme sehr gering sind, kann auch der über das Flussmittel fließende Strom zu einer Verfälschung des Ergebnisses führen.

Links / Downloads

[1] http://www.reichelt.de/?ARTICLE=81766 Dieses Gerät hat mich auf die Idee gebracht, sowas selbst zu bauen

[2] http://www.mikrocontroller.net/topic/131804 Thread zum Artikel

Hinweis: Die Entwicklung findet jetzt in einem SVN-Archiv statt. Dort finden sich auch Extras (z. B. Platinenlayouts). Wer sicher gehen will, eine aktuelle Version zu erhalten, sollte also entweder das Archiv auschecken oder den Snapshot herunterladen, der maximal einen Tag alt ist.

Firmware inkl. Schaltplan und Sourcecode: Es sind fertig kompilierte Programme für den ATMega8 und ATMega48 enthalten. Die Mega48-Version hat nicht alle Features, wie unter „Infos zur Software“ beschrieben (Achtung: Möglicherweise veraltet.)

Download-Site, wo die gewünschte Display-Sprache und weitere Optionen ausgewählt werden können (empfohlene Download-Quelle)

- aktueller Archiv-Snapshot
- SVN Archiv (online browsen)
- SVN Archiv

[3] Von „http://www.mikrocontroller.net/articles/AVR-Transistortester“ (Kategorie: AVR-Projekte)

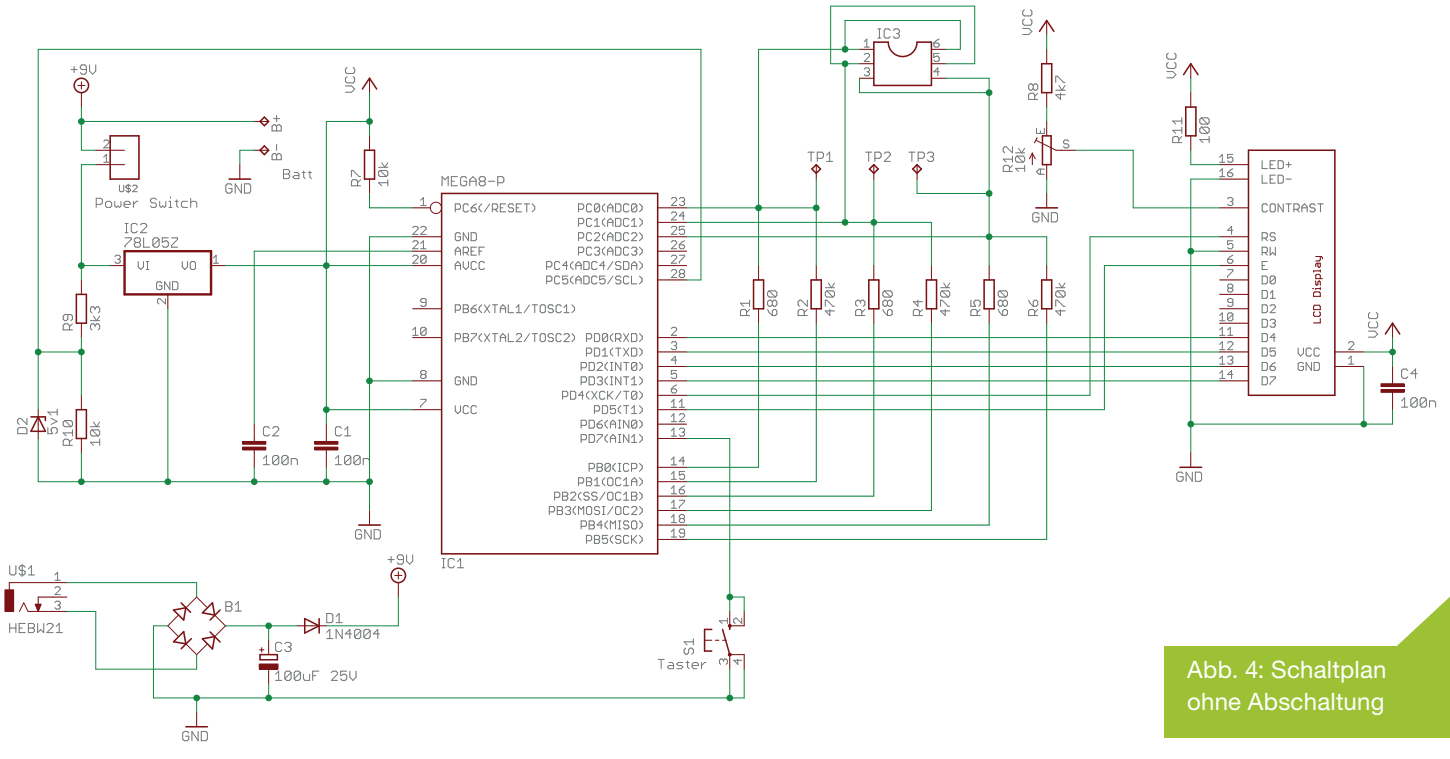


Abb. 4: Schaltplan ohne Abschaltung

Infrarot Auslöser für Nikon Kameras

Martin Matysiak <kaktus621@googlemail.com>

Einleitung

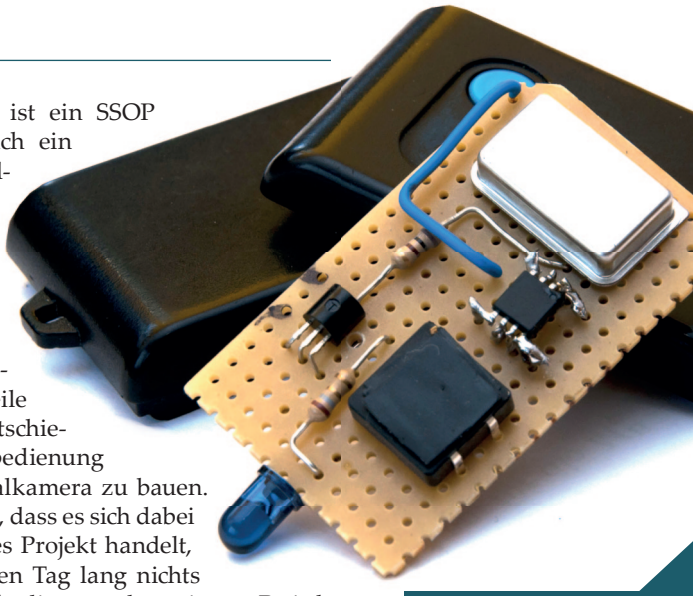


Abb. 1: Platine Vorderseite

Innenansicht - Ja, das ist ein SSOP Controller, wo eigentlich ein DIP Controller sein sollte - ich hatte keine mehr übrig und der letzte ist durch einen kleinen Fehler „gestorben“ ;-)

Während eines Moments der Langeweile habe ich mich dazu entschieden, eine Infrarot-Fernbedienung für meine Nikon Digitalkamera zu bauen. Dabei stellte sich heraus, dass es sich dabei um ein extrem einfaches Projekt handelt, perfekt, wenn man einen Tag lang nichts zu tun hat. Die Fernbedienung hat eine Reichweite von mindestens 15 - 20 Meter, mehr konnte ich bislang nicht testen. Die Fernbedienung besteht hauptsächlich aus einem Mikrocontroller, der per Knopfdruck mit Strom versorgt wird. Daraufhin wird eine Infrarot-LED mit einer Frequenz von ca. 38kHz angesteuert, die ein bestimmtes Leuchtmuster an die Kamera sendet. Die Fernbedienung ist funktionsgleich zu Nikon‘s ML-L3 und funktioniert somit mit einer vielzahl von Nikon Kameras.

Hardware

Für die Fernbedienung wird nur eine geringe Anzahl an Bauteilen benötigt. Wer öfters mit Elektronischen Schaltungen bastelt, wird vermutlich alle Bauteile lagernd haben. Nachfolgend eine genaue Auflistung:

- Plastikgehäuse für die Schaltung (am besten ein Fernbedienungsgehäuse wie in Abb. 1 gezeigt)
- Ein Stück Loch- oder Streifenrasterplatine
- 1x Atmel ATTiny13 Mikrocontroller
- 1x 8 MHz Quarzoszillator
- 1x 3V Knopfzelle (z.B. CR2032, vorzugsweise mit Lötfahnen)
- 1x Taster
- 1x Infrarot-LED
- 1x ca. 1 bis 2 Ohm Widerstand
- 1x 100 Ohm Widerstand
- 1x 10 kOhm Widerstand
- 1x BC547 NPN Transistor (oder äquivalentes Modell)
- 2x 100 nF Kondensator

Theoretisch sollte die Fernbedienung auch mit dem internen Oszillator funktionieren. In der Praxis erwies sich der Einsatz eines externen Quarzoszillators allerdings deutlich zuverlässiger. Der Schaltplan der Fernbedienung ist in Abb. 3 zu sehen.

Schaltplan

Ich werde hier kein Platinenlayout zur Verfügung stellen, da das Layout stark von der Wahl der Bauteile und des Gehäuses abhängt. Außerdem habe ich nicht erwähnt, ob ich DIP oder SMD Bauteile verwendet habe - such Dir einfach die Form aus, die Dir am besten passt! Ich persönlich habe eine Mischung aus beidem verwendet, da ich nur Bauteile verwenden wollte, die ich bereits lagernd hatte.

Software

Bevor ich den eigentlichen Quellcode vorstelle, möchte ich ein paar Worte über das Protokoll verlieren, welches die Fernbedienung einsetzt:

Das Infrarot-Signal wird mit einer Frequenz von etwa 38,4 kHz moduliert. Das bedeutet, wenn die LED „an“ ist, leuchtet sie nicht durchgehend, sondern blinkt 38400 mal in der Sekunde. Weiterhin muss die LED zwei mal in einem ganz bestimmten Muster leuchten (mit einer Pause von 63ms), damit sie von der Kamera erkannt wird. Im Quellcode nenne ich dieses Muster „burst“. Glücklicherweise haben sich bereits einige Leute mit dieser Thematik befasst und das Signal der Original-Fernbedienung genaustens analysiert. Ich habe die Daten von BigMike's Webseite [1] genommen.

Der Quellcode für den Mikrocontroller ist relativ simpel. Da der Schalter einfach die Stromversorgung aktiviert und nicht etwa als digitaler Eingang agiert, ist das einzige, was der Controller tun muss, direkt nach dem Start das richtige Muster auszugeben. Diese Tatsache spiegelt sich in der extrem kurzen Hauptmethode wieder:

```
int main(void) {

  // Initialize port

  DDRB |= (1 << IO_IR);

   burst();

  _delay_ms(63);

  burst();

     while(1) {}

}
```

Das war einfach, nicht wahr? Viel interessanter ist die burst Methode:

```
#define F_CPU 8000000UL

// DO NOT USE PB3! PIN IS USED AS CLOCK INPUT!

#define IO_IR PB4


#define LED_ON() PORTB |= (1 << IO_IR)

#define LED_OFF() PORTB &=~(1 << IO_IR)


// Length of a clock in microseconds (ca. 38,4kHz)

#define CLOCK_DURATION 9


// Number of clocks per data burst

#define BURST_LENGTH 3088


#include<avr/io.h>

#include<util/delay.h>

#include<avr/interrupt.h>


// Clock-counts at which the LED state has to be toggled

// The values are determined by experimenting, they work

// with a 8MHz crystal oscillator
```

```
const uint16_t thresholds[8] = {2,164, 2564, 2597, 2727, 2760, 3054, 3087};


// Sends a single data burst (two of them are needed)
void burst() {
  uint16_t clock = 0;
  uint8_t current_threshold = 0;
  uint8_t status = 0;

  while(clock++ < BURST_LENGTH) {
   if (clock == thresholds[current_threshold])
{
      status ^= 1;
      current_threshold++;
    }

    if (clock & status) {
      LED_ON();
    } else {
      LED_OFF();
    }

    _delay_us(CLOCK_DURATION);
  LED_OFF();
  return;
}
```

Nun, was passiert hier?

Ein Durchlauf der while-Schleife benötigt ungefähr 13 Mikrosekunden, wodurch die Modulationsfrequenz von 38.4 kHz erreicht wird. Während jeder Iteration wird ein Zähler inkrementiert. Dieser Zähler entspricht der Anzahl der bisher erfolgten Takte. Pro Durchlauf wird er mit einem Array von Schwellwerten verglichen. Wird ein Schwellwert erreicht, schaltet der LED Zustand von „an“ auf „aus“ oder umgekehrt (wobei „an“ wie vorhin erwähnt Blinken mit 38.4 kHz bedeutet).

Damit die LED also wirklich leuchtet, müssen zwei Bedingungen gelten:

- Der aktuelle Takt ist ungerade (dadurch wird das Blinken mit Modulationsfrequenz erreicht)
- Der Zustand der LED ist „an“

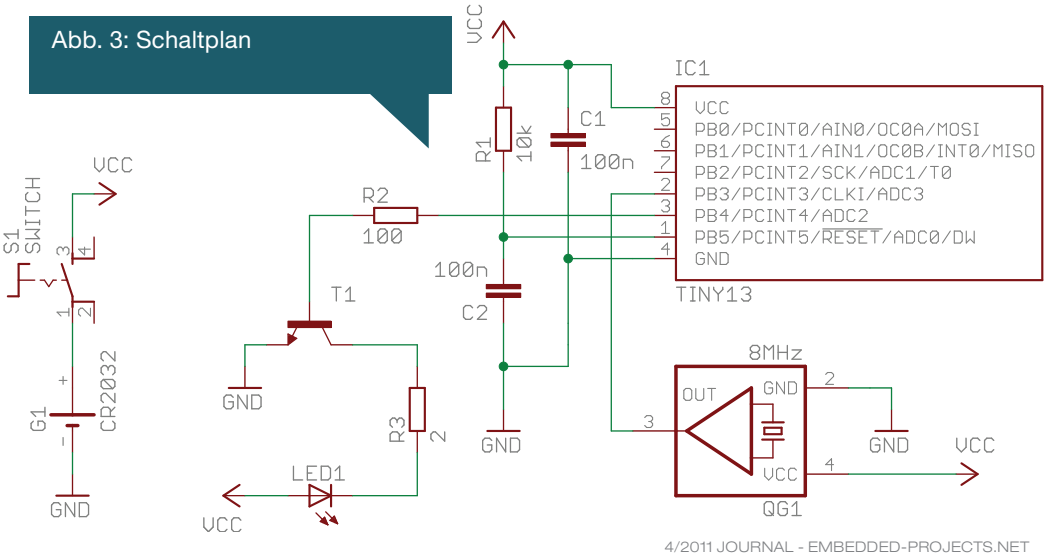
Auch wenn die status Variable ein 8-bit Integer ist, verwende ich nur das niederwertigste Bit zur Indikation des Zustandes. Der Zähler wird bei jedem Schleifendurchlauf inkrementiert, das heißt, dass das niederwerigste Bit des Zählers bei jedem Durchlauf zwischen Null und Eins springt. Nun kann man einfach die beiden Variablen UND-verknüpfen - wenn das Ergebnis der Verknüpfung eine Eins ist, heißt es, dass beide Bedingungen erfüllt sind und die LED leuchten kann. □

Abb. 2: Platine Rückseite im Gehäuse

Links / Literatur

[1] www.bigmike.it/ircontrol

Abb. 3: Schaltplan



Röhrenradio im Nachbau

Markus Musielak mars@augusta.de

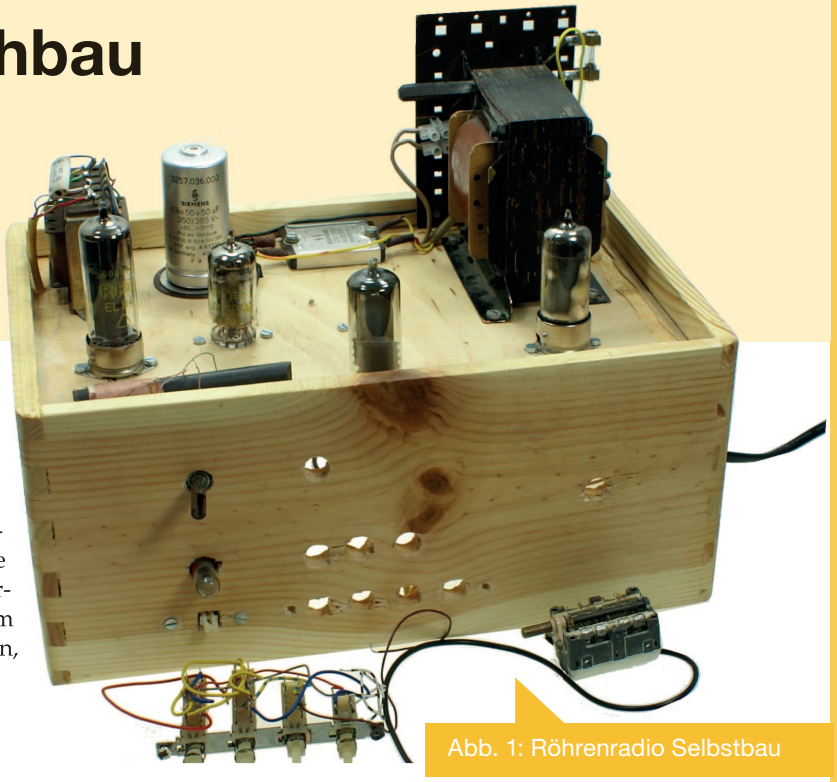


Abb. 1: Röhrenradio Selbstbau

Eine Kiste voll Material ...

Vor Jahren baute ich einen Stereo-Röhrenverstärker und einen Synthesizer, den ich vor etwa einem Jahr in einem 19“-Gehäuse umbaute. Weil ich einem Ex-Elektronikbastler eine alte Gebrauchsanleitung kopierte, bekam ich eine ganze Kiste verschiedener Kleingeräte und Material. Darunter einige Teile aus einem SABA-Radio . Daraufhin kam mir die Idee ein Röhrenradio (siehe Abb. 1) nachzubauen, trotz Synthesizertuner, Internet- und Satellitenradio.

Die Woodware

Als Gehäuse dient eine Holzkiste aus dem Baumarkt. Um Störstrahlungen zu vermeiden muss man im Inneren mit Alufolie o.ä. auskleiden. Man kann auch das Gehäuse komplett aus Blech bauen.

Hier eine Skizze (Abb. 2), wie man es aufbauen kann: Die Bedieneinheit besteht aus einem Senderknopf auf einem Luftdrehko (Abb. 3 rechts), einem Lautstärkeknopf, einem Vorreglerknopf für die Gitterspannung des Mischsystems, Einschaltknopf und einem Umschalter für die Frequenzbereiche LW, MW und evtl. KW (nicht in der Schaltung vorhanden). Zur Betriebsanzeige dienen 7V Skalenlämpchen (Abb. 3 li.). Ein Umschalter für die Frequenzbereiche LW, MW und evtl. KW (nicht in der Schaltung vorhanden). Zur Betriebsanzeige dienen 7V Skalenlämpchen (Abb. 3 li.). Die Röhren sind typische Rundfunkröhren (Abb. 4), die noch leicht zu beschaffen sein sollten. Als Aussteuerungsanzeige habe ich noch einen Magischen Fächer EM 80 gefunden (Abb. 4 re. vorne).

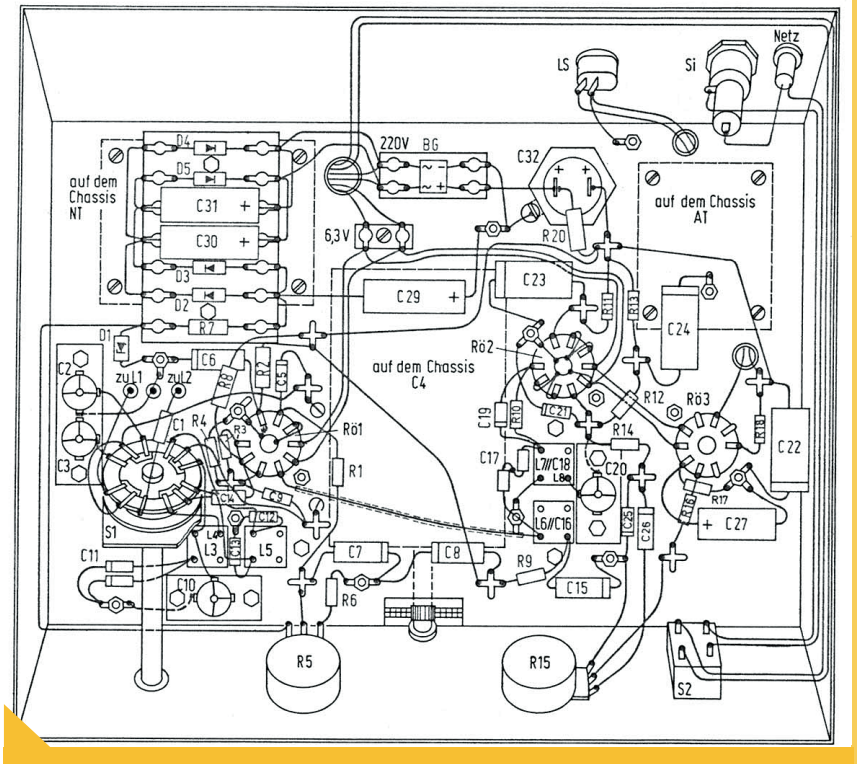


Abb. 2: Audion Kleinsuperhet Lötaufbau (Quelle Link 1)

Die Schaltung

In Abb. 8 ist der Schaltplan dargestellt. Das Netzteil besteht aus einem umschaltbaren Netztransformator, Gleichrichter und Siebelko C32 A/B für die Betriebsanodenspannung + 235 V. Die Röhrenfäden haben die übliche Heizspannung von 6,3 Volt. Für die Vorregelung der Misch- und Oszillatorröhre ECH81 Rö wird noch -0,8V bis -28V benötigt, wo auch das HF der Ferritantenne ankommt, um eine Übersteuerung bei Ortssendern zu vermeiden. S1 ist der Mehrfachumschalter zwischen Mittel- und Langwelle mitsamt der zugehörigen Filterkreise. Der Gitterkreis der NF-Pentode EF86 Rö2 arbeitet als Diodendemodulator und erscheint an der Anode die HF- und NF-Spannung in Abhängigkeit ihrer unterschiedlichen Außenwiderstände spannungsverstärkt. L6/L7 dient als ZF-Bandfilter. Für eine Rückkopplung ist C20, C21 und L8 zuständig. Das Nf-Signal läuft dann über R14, C26 zum Lautstärkeregler R15. Die

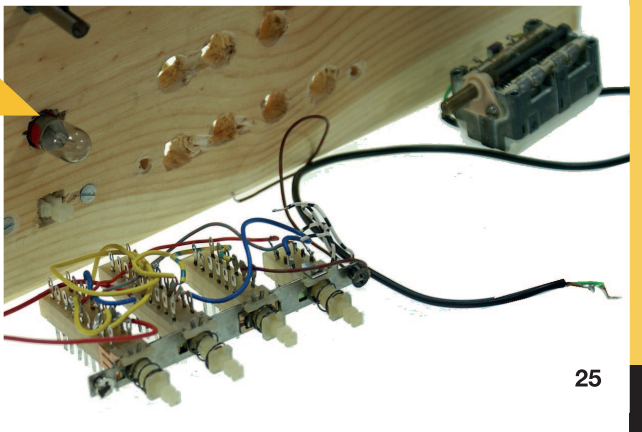


Abb. 3: Luftdrehko (re.), Umschalter (li.), Skalenlämpchen (li.)

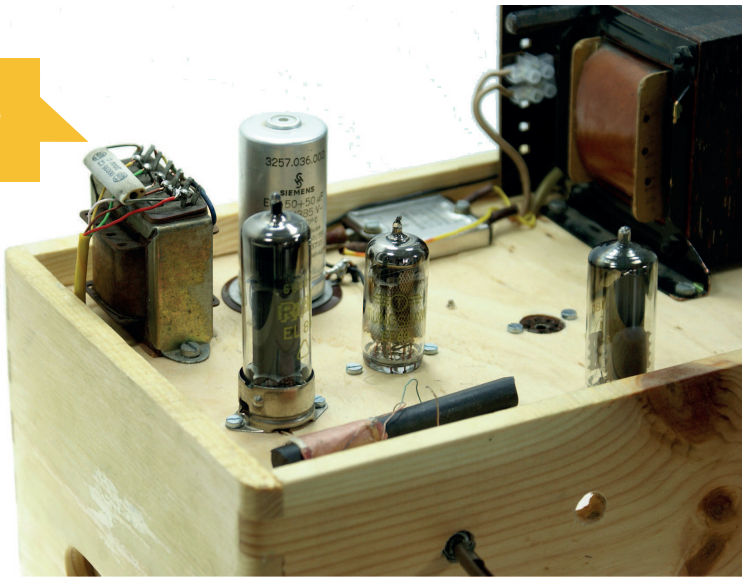


Abb. 4: Röhren + Anzeige EM 80

Leistungsendpentode EL84 Rö3 macht 4 - 5W (reicht für Zimmerlautstärke) und geht auf den Ausgangsübertrager zum Lautsprecher. Weil Röhren hochohmige Bauelemente sind und Lautsprecher niederohmig (üblich 4 – 16 Ohm), benötigt man einen Niederfrequenz-Transformator (also Ausgangsüberträger) mit guter Bandbreite und idealerweise Luftspalt, um Verzerrungen zu vermeiden.

Abb. 5: Audion Kleinsuperhet Stückliste (Quelle Link 1)

Stückliste

Röhren:
1× ECH81 (Rö1)
1× EF86 (Rö2)
1× EL84 (Rö3)

Halbleiter:
4× 1N4007 (D2, D3, D4, D5)
1× ZPY 30 (D1)
1× B380 C800 (BG)

Widerstände (5 %)
¼ W Belastbarkeit
1× 162 Ω (R17)
1× 220 Ω (R3)
1× 470 Ω (R18)
1× 1 kΩ (R16)
1× 1,5 kΩ (R6)
1× 8,2 kΩ (R9)
1× 18 kΩ (R13)
1× 47 kΩ (R4)
1× 56 kΩ (R12)
1× 100 kΩ (R14)
1× 360 kΩ (R11)
1× 750 kΩ (R1)
1× 1 MΩ (R10)

1 W Belastbarkeit
1× 24 kΩ (R2)
1× 36 kΩ (R8)
1× 120 kΩ (R7)

2 W Belastbarkeit
1× 270 Ω (R20)

Potentiometer:
1× 47 kΩ, lin. (R5)
1× 470 kΩ, log. (R15)

Kondensatoren:
1× 1,9 pF (= 2×3,9 pF, 5%, 150 V; in Serie) (C17)
1× 68 pF, 10%, 250 V (C25)
3× 100 pF, 10%, 100 V (C5, C9, C19)
1× 100 pF, 20%, 250 V (C21)
1× 120 pF, 2%, 63 V (C16)
1× 120 pF, 10%, 160 V (C12)
1× 136 pF, (= 2×68 pF, 2%, 63 V; parallel) (C18)
1× 150 pF, 2%, 160 V (C13)
1× 470 pF, 10%, 400 V (C14)
1× 490 pF (= 220 pF, 270 pF, 5%, 250 V; parallel) (C11)
1× 1,5 nF, 5%, 63 V (C1)
1× 33 nF, 20%, 400 V (C26)
4× 100 nF, 20%, 400 V (C6, C7, C8, C15)
2× 470 nF, 20%, 400 V (C22, C23)
1× 1 µF, 20%, 400 V (C24)
2× 4,7 nF, 350 V, Elko, axial (C30, C31)
1× 10 µF, 350 V, Elko, axial (C29)
1× 100+100 µF, 350 V, Elko mit Schraubsockel (C32)
1× 220 µF, 16 V, Elko (C27)

Trimmkondensatoren:
4× 4–40 pF (C2, C3, C10, C20)
1× Luftdrehkondensator 2×520 pF (C4)*), z. B. Hopt G15-00

Spulen:
4 Einzelspulen-Bausätze (siehe Spulen-Tabelle)
1 Ferritstab 200×10 mm (FA)

Sonstiges:
1 Drehwellenschalter, 1 Ebene, 4×UM (S1)
1 Netzschalter 2polig (S2)
1 Netztransformator MD-55-Kern (NT) Bv 20 047 (Schumacher, München)
1 Ausgangsübertrager EI-60-Kern (AT) Bv 20 186 (Schumacher, München)
3 Novalfassungen für Chassismontage
1 Drehkoantrieb mit Skala
1 Lautsprecherbuchse
1 Sicherungshalter mit Feinsicherung, 0,315 mA mtr.
div. Lötstützpunkte
1 Alu-Chassis, Breite ca. 220 mm, Tiefe ca. 150 mm, Höhe ca. 50 mm
5 Drehknöpfe
1 Breitband-Lautsprecher, 4 Ω/4 W

*) Evtl. Ausbau aus einem ausgedienten Rundfunkempfänger der 50er und 60er Jahre.

Aktueller Status

Aktuell funktioniert das Netzteil und der Verstärkerteil (EL84, AÜ und LS). In Abb. 7 sieht man die momentane interne Verkabelung. In Abb. 5 ist die Stückliste abgedruckt und in Abb. 6 die technischen Daten der Spulen. □

Literatur / Links

[1] Artikel: http://www.jogis-roehrenbude.de/Bastelschule/Funkschau-Nostalgie-Radio/Funkschau-Audion.htm

[2] Röhrenladen „Frag Jan Zuerst“: http://www.die-wuestens.de/dindex.htm

[3] Röhren bei Pollin: http://www.pollin.de/shop/p/MDk4OTk4/Bauelemente_Bauteile/Aktive_Bauelemente/Roehren_Roehrenfassungen.html

[4] Funktionsweise von Röhren: http://www.elektronikinfo.de/strom/roehren.htm

Wer Lust hat jetzt mitmachen! Herbst und Winter ist Zeit zum basteln. Aber immer dran denken:

Vorsicht Hochspannung !

Und

Vorsicht Verbrennungsgefahr !

Abb. 6: Audion Kleinsuperhet Spulendaten (Quelle Link 1)

Position	Induktivität (µH)	Abgleichbereich	Windungszahl	Beispiel b. A_L-Wert*) (nH)	Draht	Bemerkungen
L1	165	−15 %, +10 %	47 (1 Lage)	78	6× 0,07 CuLS	auf Ferritstab, ±10 mm verschiebbar
L2	2520	−15 %, +10 %	178 (3 Lagen)	80	3× 0,07 CuLS	auf Ferritstab, ±10 mm verschiebbar
L3	99	±15 %	70	20	3× 0,05 CuLS	
L4	–	–	8	–	0,2 CuLS	gegensinnig zu L3 gewickelt
L5	539	±15 %	163	20	3× 0,05 CuLS	
L6	820	±15 %	202	20	3× 0,05 CuLS	
L 7	770	±15 %	196	20	3× 0,05 CuLS	
L8	–	–	8	–	0,1 CuLS	gegensinnig zu L7 gewickelt

*) A_L-Wert ist Induktivität **einer** Windung!

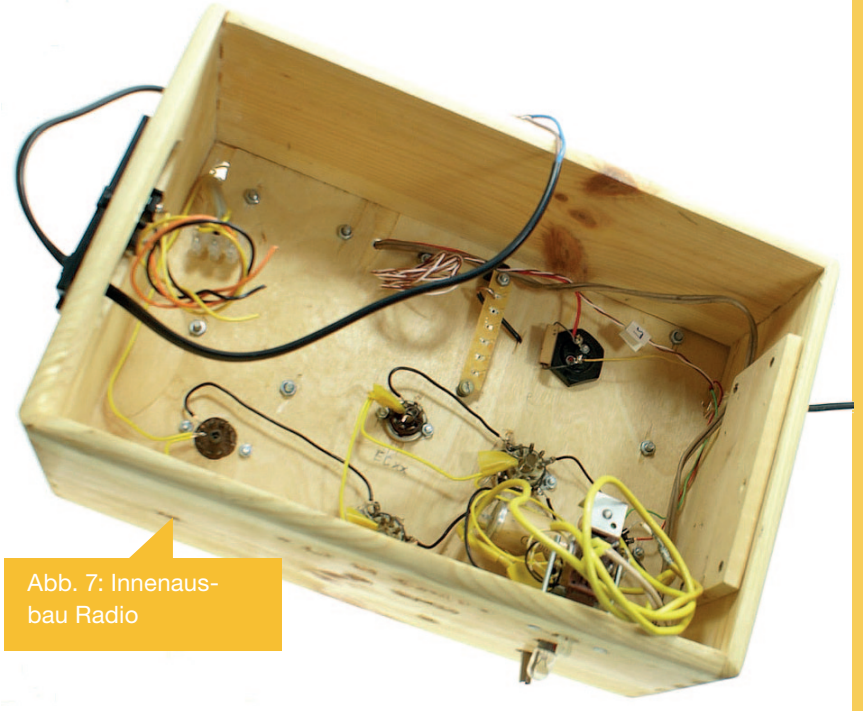
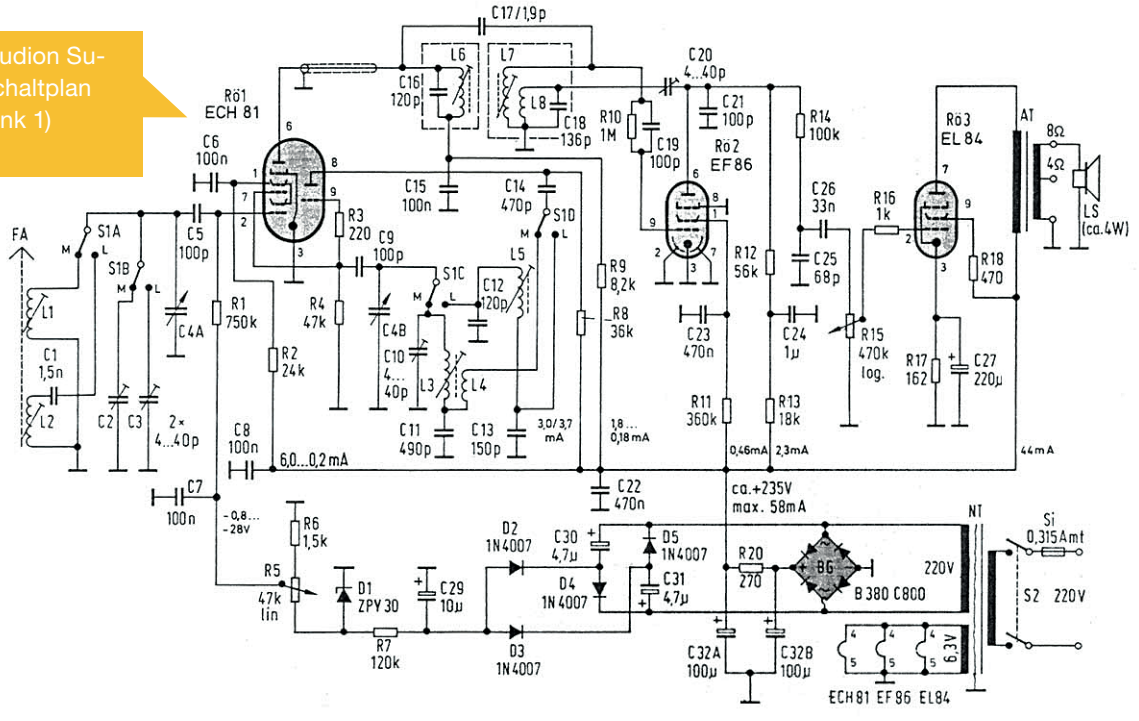


Abb. 7: Innenausbau Radio

Abb. 8: Audion Superhet Schaltplan (Quelle Link 1)



BGA Löten mit dem Pizzaofen

Ein Projekt mit Vorgeschichte

Benedikt Sauter < sauter@embedded-projects.net >

Die Vorgeschichte

Seit nun ca. 8 Monaten - muss ich gestehen - nehme ich den Lötkolben immer seltener in die Hand. Früher habe ich komplette Prototypen bis zur Baugröße 0402 - wenn es denn sein musste - von Hand bestückt. Den meisten sage ich immer: „man muss davor keine Angst haben“. Mit Lötlitze und oft einer extra Ladung Lötzinn und dem ein oder anderen kräftigen Schlag auf den Tisch um das Zuviel auch wieder loszuwerden, verhilft man sich zu sehr guten Lötstellen. Anfang dieses Jahres hatte ich dann zum ersten Mal einen fast 100 poligen Stecker mit einem Raster von 0,5 mm zu löten. Da hört dann doch der Spaß auf, vor allem musste diese Schaltung auf Grund eines Auftrags laufen und konnte Notfalls nicht in der Schublade verschwinden.

Alternative zum Lötkolben ?

Zurück zu den Anfängen... als ich mit der Elektronik so langsam begonnen hatte, habe ich Sonntags einmal eine Adapterplatine für einen TQFP geätzt. Natürlich wollte ich diesen direkt löten, hatte jedoch nur einen katastrophalen Lötkolben zur Hand. Was ich jedoch komischerweise da hatte, war eine SMD-Pasten Spritze aus Kunststoff. Kurzerhand habe ich eine Wurst SMD Paste auf jeder Seite „hingebazelt“ und das ganze anschließend in den Küchenofen bei 230 Grad gelegt. Man muss von außen sehr gut auf die Lötstellen schauen. Wenn die Paste zu schmelzen beginnt und schön um die Beinchen verfließt sieht man das. Sobald also die Paste silbern glänzt und gut verflossen ist schnell die Ofentüre zum Abkühlen öffnen. Das mit dem Ofen musste ich leider schnell wieder aufhören, als herausgefunden wurde was ich dort mache, denn die Platinen sollte man ja nicht im selben Ofen backen wie seine Plätzchen... Aber diese Erfahrung, dass das SMD-Paste Löten einfacher geht als gedacht hatte ich jedoch gemacht.

Abb. 1: Ofen mit Steuerung

Auf der Suche nach einer sinnvollen Alternative für den Lötkolben war mir nun klar was ich brauchte. Relativ schnell bin ich dann auf das Reflow-Kit von PCB-Pool gestoßen. Das ist im Wesentlichen ein einfacher Pizza-Ofen, wie es Ihn regelmäßig in jedem Supermarkt für z.T. ab 20 bis 30 EUR gibt. Zusätzlich zum Ofen gibt es auch noch eine Steuerung. Man steckt die Steuerung in eine 230V Steckdose, steckt in die Steuerung wiederum den Stecker des Ofens und schaltet am Ofen die Zeitschaltuhr auf mindestens 30 Minuten (am besten aber gleich auf Anschlag), dann Ober- und Unterhitze (gleichzeitig) einstellen und den Temperaturregler auf das Maximum. In den Ofen legt man noch den Temperatursensor der Ofensteuerung, der idealerweise mit Draht auf Platinenmaterial befestigt ist. Auf diese Weise misst man die Temperatur sozusagen genau auf der gleichen Höhe über der Platinenoberfläche wo später auch die Paste an die Beinchen und Kontakte der bestückten Platine fließen soll.

Das Wesentliche ist aber beim Löten mit dem Ofen und dieser Steuerung nicht der Ofen. Auch nicht die Steuerung und ebenfalls nicht der Temperatursensor. Darauf könnte ich jetzt wieder verzichten. Sondern das Verfahren: Ofen brutal auf 230 Grad aufheizen lassen, Platine einlegen und schauen bis das Lötzinn schmilzt und dann schnell die Ofenklappe öffnen. So spart man sich z.B. zunächst den Temperatursensor und die Steuerung. Das kritischste in meinen Augen beim Löten mit dem Ofen sind die SMD-Pastenschablonen, die SMD-Paste und das Auftragen der Lötpaste mit einem Rakel bzw. etwas geradem. Wenn man das gut heraus hat, klappt jede Platine für wenige Stückzahlen tadellos.

Löpaste auftragen

Die erste Frage stellt sich natürlich: Woher kriegt man eine Lötpasten Schablone und was ist das überhaupt? Ganz einfach, eine Lötpasten Schablone oder Pastenmaske ist ein dünnes Blech ca. 150 um auf dem überall Löcher bzw. kleine Rechtecke sind wo Pads von Bauteilen später mit der Platine verlötet werden sollen (siehe Abb. 2). In Eagle ist das Layer 31 tcream und 32 bcream. Diese Layer sollte man ab jetzt immer sehr gut kontrollieren. Dazu aber nachher noch mehr. Mit dieser Vorlage (der Pastenmaske) kann man die Lötpaste auf die Platine „drucken“ im Prinzip ist das wie beim Siebdruck. Man richtet die Maske schön auf den Pads der Platine aus. Dies muss man von oben mit den Augen oder ggf. einer Lupe kontrollieren. Wenn alle Pads schön sichtbar sind kann man die Schablone mit Klebestreifen am oberen Rand auf einer Unter-

lage festkleben (z.B. Platinenmaterial, Hartfaserplatte von einem alten Bilderrahmen, etc. mindestens Din A4). Zuvor sollte man die Platine auf der Unterlage mit kleinen Stücken aus gleich hohem Material wie die Platine selbst (z.B. alten Platinen, am besten winkelförmig), so dass diese nicht mehr verrutscht.

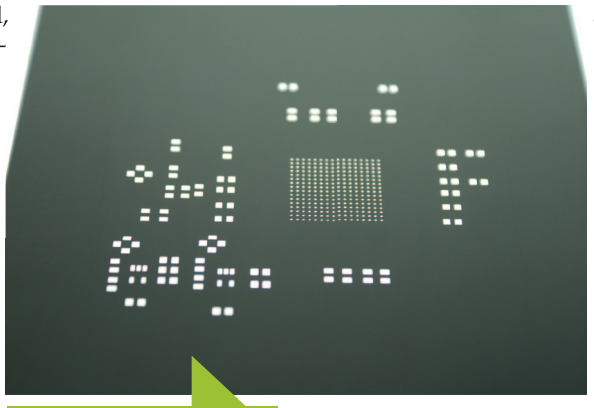


Abb. 2: Schablone für die Lötpaste

Anschließend nimmt man sich die Lötpaste. Idealerweise lagert man diese kühl und nimmt sie direkt vor dem Löten heraus und rührt dann knappe 15-30 Sekunden mit einem Spatel, bis sie schön „geschmeidig" ist. Verteilt wird die Paste auf einer Seite der Maske, indem man eine dicke Wurst - ca. 5-8 mm Durchmesser (siehe Abb. 5) - über eine ganze Schablonenseite unterhalb des Malerkrepps aber über den Pads zieht. Welche Seite man verwendet findet man nach und nach heraus. Oft beginne ich gerne mit der Seite mit den „feineren" Pads und ende bei den „größeren". Manchmal bietet es sich auch an, eher von der schmaleren als von der längeren Seite zu beginnen, weil dann die Pasten-Wurst nicht so lang sein muss und man außerdem mit dem Rakel (einfache Metallschaber aus dem Baumarkt ab ca. 1 EUR - die gehen prima) besser Druck auf einer kleinen Stelle als über eine lange Seite erhält. Ist die Pastenwurst also bereit und man ist sich klar, von wo aus man mit dem Rakel ziehen möchte (an dieser Kante muss die Schablone auch festgeklebt sein), sollte man die Platine schön vor sich ausrichten und bequem sitzen, denn jetzt kommt der aller kritischste Moment. Man muss mit genügend Druck die Paste über die Schablone gleichmäßig ziehen, so dass die Paste nicht obenauf der Schablone in Schmierresten zurück bleibt, aber auch jedes Pad-Loch schön mit der Paste gefüllt ist. Man darf nicht zu oft herumdrücken, sonst verschiebt man jedes mal die Schablone etwas und die Paste verschmiert sich an den Rändern und läuft weiter zwischen Schablone und Platine. Im günstigsten Fall zieht man die Paste mit einem Mal sauber über die gesamte Seite.

Am Anfang braucht man ein paar Versuche. Das macht aber gar nichts. Mit dem Rakel kann man ganz vorsichtig die Paste von der Platine wieder nehmen, die Schablone gut reinigen. Ich nehme hierfür Platinenreiniger aus der Sprühdose mit einer Bürste vorne dran und einem Schwung Papiertücher. Die Platine kann man auch nochmal schnell mit den Tüchern und dem Reiniger von der verschmierten Paste befreien. Ist die Platine einmal erfolgreich bedruckt, legt man dieser am besten ein paar Meter ganz weit weg, wo man nicht aus versehen dran kommt. Auch das wird jedem zwischendurch passieren :-).

Zur Frage: Woher bekommt man diese Maske? Es gibt viele Anbieter im Internet, die solche Pastenmasken ab ca. 25 EUR und teurer anbieten. Man muss aufpassen, viele bieten sehr gute Industriequalität für einen Druck von einigen 1000 Platinen und mehr an, aber das braucht man ja nicht. Letztens habe ich bei http://www.bilex-lp.com welche bestellt. Dort kosten Sie 25 EUR + 9 EUR Versand. Die Qualität war super. Was ich meistens jedoch mache, ich bestelle die Platine bei PCB-Pool, dort kann man immer die Maske kostenlos mit bestellen. Man darf nur nicht vergessen in der Bestellung den Haken „FreeStencil" an zu klicken. Bei den meisten Platinenherstellern kann man die Schablonen immer kostenpflichtig zur Platine gleich mit bestellen.

Checkliste: Schablonen-Daten

Bevor man die Schablone bestellt, sollte man aber zuvor einen Blick auf die entsprechenden Layer werfen. Aber auch hier bekommt man nach und nach Erfahrungen. Ich habe einmal meine wichtigsten notiert:

- Sind die Löcher der Pastenmaske etwas kleiner als das Kupferpad? Das ist wichtig, denn wenn die tcream und bcream Löcher gleich groß oder zu eng aneinander sind, besteht die Gefahr, dass beim Drucken zu viel Paste auf dem Pad stehen bleibt und auch darüber hinausläuft. Nach dem Löten im Ofen hat man dann häufig Lötbrücken zu anderen Pads (z.B. bei SMD-Buchsen, im Vergleich zu Chips sind diese aber noch relativ unkritisch und leicht zu beheben).
- Hat ein Chip ein Exposed-Pad (siehe Abb. 3) (unter dem Chip eine Metallfläche), sollte man an dieser Stelle auf der Platine lieber eine kleine Matrix von Rechtecken anordnen, sonst hat man schnell viel zu viel Lötzinn unter dem Chip. Manche machen auch in die Mitte eine Durchkontaktierung, so dass dort überflüssiges Lötzinn „abfließen", bzw. man später auch einfacher den Baustein wieder entfernen kann.

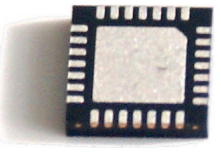


Abb. 3: QFN

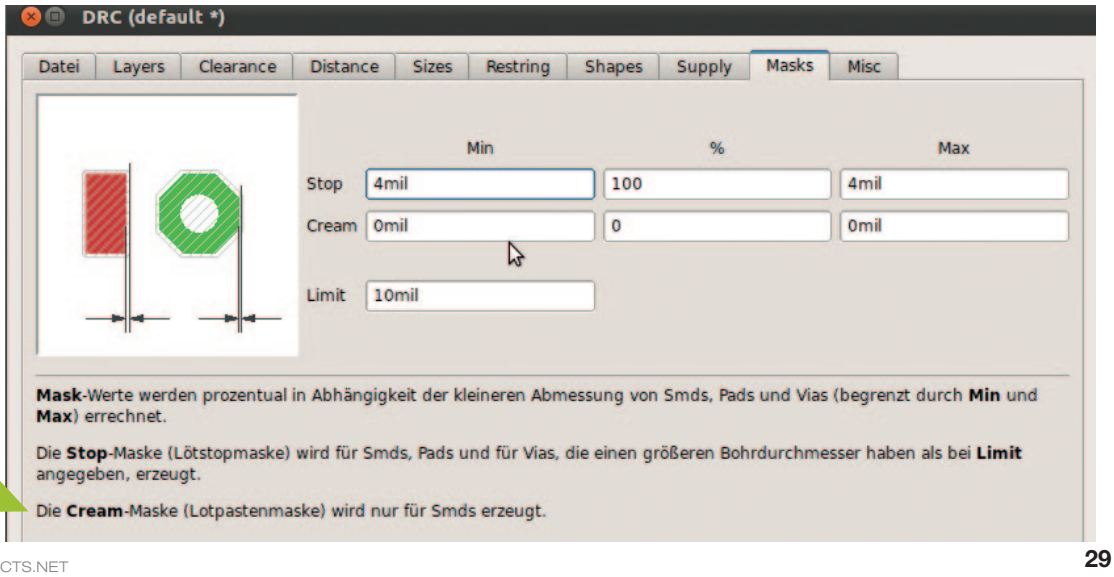


Abb. 4: Stoppmaske

- Nicht zu vergessen ist auch, dass man bei normalerweise ca. 150um dickem Blech (mit etwas kleineren Löchern als die Pads) an manchen Stellen nicht plötzlich sogar zu wenig Lötpaste pro Pad hat. Hat man dickeres Blech muss man aber unbedingt die Löcher nochmals verkleinern, sonst ist das zu viel Lötpaste. Hat man hingegen dünneres Blech, muss man sie entsprechend wieder vergrößern, wobei hier dann beim Drucken die Gefahr vom „verschmieren" wieder steigt. Klingt schwieriger als es ist. Bei etwas größeren Bauteilen wie Buchsen, Quarzen, Spulen ist dies nicht ganz so kritisch und funktioniert meist auch ohne Probleme. Im Notfall kann man diese Teile ja noch mit Litze nach bearbeiten. Man sollte deshalb zu Beginn ersteinmal langsam den Schwierigkeitsgrad steigern und nicht gleich zu Anfang alle kritischeren Chip- und Bauteil-Sorten auf einer Platine gleichzeitig vereinen.

- Meine Platinen lasse ich immer mit Lötstopplack lackieren. Ohne Lack besteht schneller die Gefahr eine nicht gewollte Lötstelle/Lötbrücke zu erhalten. Wenn man auf den Lötstopplack verzichten möchte, sollten die Leitungen aber so weit wie möglich von Pads entfernt sein - bzw. idealerweise ganz grade von den Pads weglaufen und außerdem nicht unbedingt parallel an andere Pads angrenzen. Immer genügend Abstand! Dann kann nichts rüberlaufen und man spart sich das lästige Nacharbeiten am Prototyp.

Zum Kernpunkt des Artikels

Nach einiger Erfahrung mit dem Löten mit Rakel und Ofen wollte ich mich nun an BGA Bausteine (siehe Abb. 5) wagen.

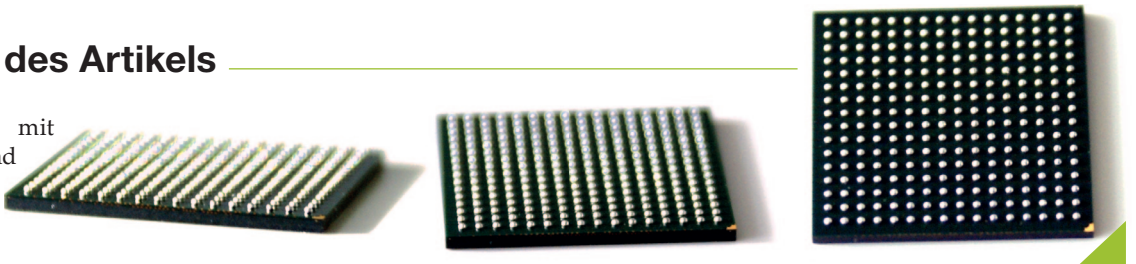


Abb. 5: BGA von unten

Bei BGA Bausteinen ist ein wesentlicher Faktor der „Pitch". Das ist der Abstand von Ball zu Ball. Hier gibt es 1.00 mm, 0.8mm oder 0.5mm. Sicherlich gibt es noch feinere, aber dafür ist der Ofen wohl nicht geeignet. Die Platine sollte immer mit Lötstopplack erstellt werden - und (ganz wichtig) Durchkontaktierungen im BGA dringend über lackiert sein. In Eagle stellt man das bei DRC (siehe Abb. 4) unter dem Punkt „Masks" ein. Das Limit muss größer als die Durchkontaktierungen gestellt werden. Dann werden diese mit Lötstopplack bedeckt und die Gefahr für ungewollte Kontakte wird drastisch minimiert. Passend für den BGA ist in der Pastenmaske dann ein entsprechendes Löcherraster für die Pads. Das Drucken geht auf die gleiche Art und Weise wie oben bereits beschrieben. Man benötigt ein wenig mehr Zeit, bis man die Schablone auf der Platine ausgerichtet hat. Das wars aber eigentlich. Die Schwierigkeit beim BGA Baustein liegt darin, dass man den Baustein von oben platzieren muss und jeder Ball auf seinen „Lötpastehaufen" treffen muss (Abb. 7 und Abb. 8). Dafür kann man sich aber eine kleine Hilfe in die Platine machen. An den Ecken des BGA's füge ich kleine Markierungen ein (siehe Abb. 9 und Abb. 10). Wichtig ist, dass diese korrekt an gegenüberliegenden oder mindestens drei geraden Seiten sind. So dass man den Baustein parallel zu den Linien auf die Platine legen kann. Nun zum Platzieren des BGA: Wenn man den BGA vorsichtig von oben mit der Pinzette ganz locker auf die Pastenmatrix auflegt (entsprechend der Markierung auf der Platine, und Achtung: Pin 1 nicht vergessen), kann man ihn durchaus noch mit der Pinzette auf den „Pastenhäufchen" herum schieben, ohne diese zu verschmieren. Am besten sieht man dies von der Seite. Keine Panik, wenn der BGA mit den Balls direkt zwischen den Häufchen gelandet ist. Entweder man nimmt ihn vorsichtig nochmals nach oben senkrecht weg, oder man kann ihn sogar noch auf die Häufchen „drauf schieben". Sitzt der BGA von allen Seiten betrachtet perfekt auf den Löthäufchen (Kontrolle von allen Seiten von schräg unten), so kann man ihn nun vorsichtig mit der Pinzette erst mittig und dann an den Kanten in die Paste eindrücken. Voilà.

Abb. 6: Schablone und Platine beim Rakeln auf einer Metallunterlage. Gut zu sehen ist hier die „Pastenwurst"

Achso – ich bestücke immer als erstes den BGA und später die Bauteile außen herum, so hat man beim BGA platzieren genug Freiheit in alle Richtungen. Nur Vorsicht: Nicht aus Versehen die Nachbarpads verschmieren. Am besten Ärmel hochkrempeln :)

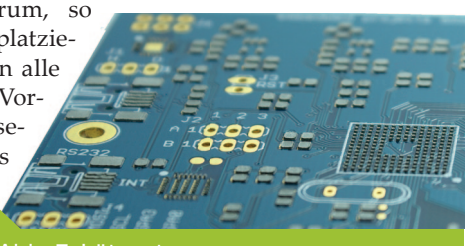
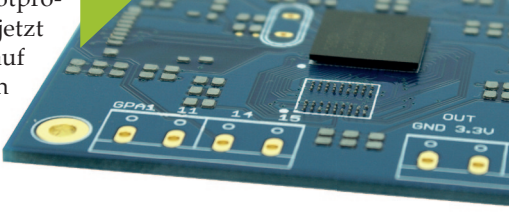


Abb. 7: Lötpaste

So - das war schon das Löten eines BGAs. Jetzt geht's genauso weiter wie sonst. Alle restlichen Bauteile platzieren. Die Platine vielleicht etwas vorsichtiger als sonst in den Ofen legen (ja nirgends dagegen rumpeln), Ofen anstellen, entweder die Kurve mit der Steuerung abfahren oder den Ofen auf 230 Grad einstellen und den Lötprozess starten. Ab jetzt wieder kritisch auf die Lötstellen von den Pads achten. Wenn die Paste zu schmelzen beginnt sicherheitshalber nochmals mindestens 10 – 15 Sekunden mehr Zeit geben, denn die Balls in der Mitte vom BGA brauchen natürlich etwas länger. Nur zulange sollte man die Chips nicht im Ofen backen. Aber nach dem ersten „Keks" hat man auch das gelernt :). Wenn man mit der automatischen Steuerung lötet kann man die Standardkurve verwenden - das hat bei mir bisher gut funktioniert.

Abb. 8: Lötpaste



Testplatine

Um zu prüfen wie gut das Löten klappt, habe ich eine Testplatine mit einem AP7000 erstellt. Auf der Platine war der BGA, die Stromversorgung, ein JTAG Stecker und der Quarz. Hier könnte man jetzt mit einem JTAG-Adapter Boundary-Scan Muster erzeugen. Jedoch habe ich keine „schnellere“ Lösung ohne größeren Aufwand gefunden. Daher habe ich Nachbar-Balls miteinander verbunden. Auf diese Weise kann mit einem einfachen C-Programm, das man in den SRAM des AP7000 per JTAG laden kann, die Pins nacheinander als Ein- und Ausgang schalten und sehen, ob die Lötverbindungen geklappt haben. Leider kann man so nur schlecht herausfinden, ob sich zwei fremde Nachbarn verbunden haben. Aber so etwas wird man spätestens beim Programmieren und Einsatz der Pins feststellen. Man muss sich nur daran erinnern, dass es auch das NICHT erfolgreiche Löten sein kann, warum etwas nicht funktionieren will, wie es aber funktionieren sollte. Zeit um das Testprogramm zu implementieren hatte ich leider noch nicht gefunden. In der Zwischenzeit habe ich einen LPC3131 samt BGA SDRAM Baustein gelötet und von dort ein Linux gestartet. Das Linux bootet soweit und daher kann ich sagen die Verbindungen stehen. Auch hier ging es ja zunächst zuerst um einen Prototyp.

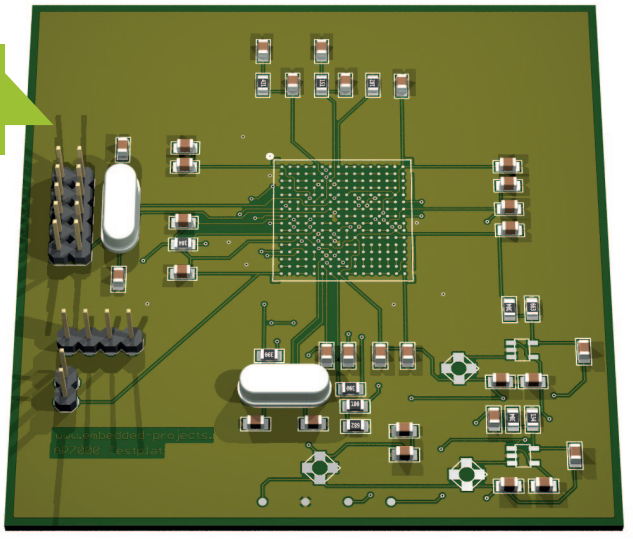


Abb. 9: Markierungen an den Ecken

Fazit

Ich kann das Löten auf diese Art und Weise nur empfehlen. Mit einem relativ kleinen Budget können erstaunlich gute Ergebnisse erzielt werden, vor allem auch bei Bausteinformen, an die ich mich zuerst nicht herangetraut hatte. Besonders für einzelne Prototypen oder Kleinserien ist diese Methode bestens geeignet. Sie erfordert anfangs zwar etwas Mut und Geduld, nach einiger Übungszeit sitzt der Baustein aber schon beinahe auf Anhieb richtig - erstaunlich, wie schnell man sich so etwas feinmotorisches antrainieren kann :)

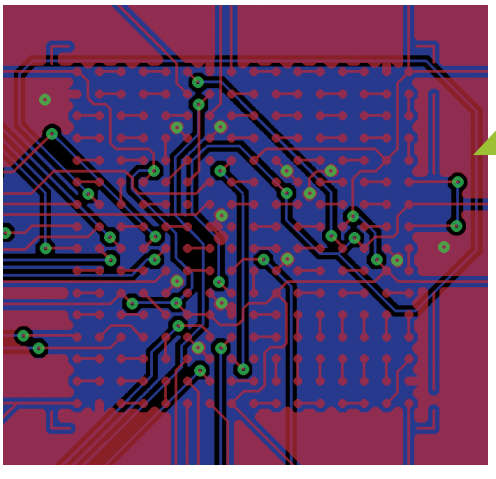


Abb. 10: Markierungen an den Ecken

Literatur / Links

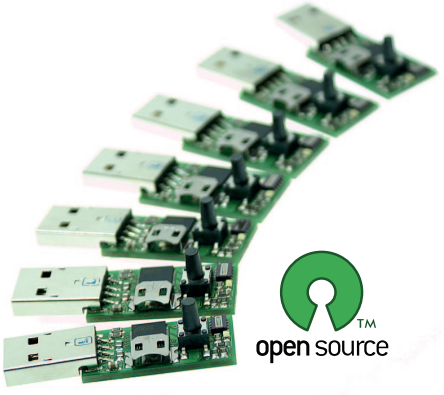
[1] PCB POOL http://www.pcb-pool.com/

[2] Bilex http://www.bilex-lp.com

[3] Elektor Stencil Maschine http://www.youtube.com/watch?v=zGc8dwUaUVo

Marktplatz / Neuigkeiten

Die Ecke für Neuigkeiten und verschiedene Produkte



Einmalpasswortgenerator

Die neue Version des Einmalpasswortgenerators (ehemals OpenKUBUS) mit Batterie und Echtzeituhr ist jetzt verfügbar. Der kleine Tipptoken kann als zusätzlicher Sicherheitsmechanismus in Kombination zu Benutzername und Passwort verwendet werden. Das mittels Token generierte Passwort ist eindeutig dem jeweiligen USB-Stick zuzuordnen. Dank der neuen Echtzeituhr kann der Zugang individuell zeitlich begrenzt oder auch komplett gesperrt werden. Nähere Infos gibt es auf der Projektseite.

Firma: embedded projects GmbH

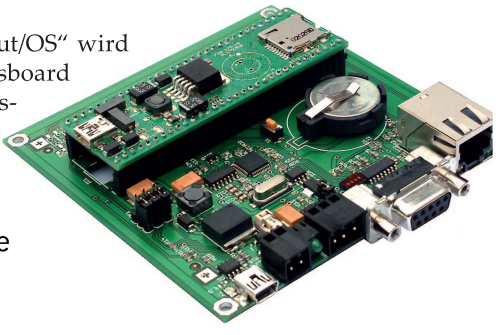
http://www.opencrypt.de

Mikrocontroller-Entwicklung leicht gemacht

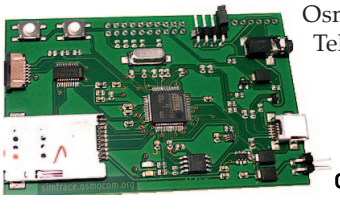
Mit dem eNet-PLC Board und dem OpenSource Mikrocontroller Betriebssystem„Nut/OS“ wird Messen, Steuern und Regeln mit Web-Interface zum Kinderspiel. Ideales Entwicklungsboard für das eNet-sam7X Modul z.B. als Ergänzung oder Ersatz für herkömmliche SPS Systeme.

Firma: Thermotemp GmbH

http://www.embedded-it.de/produkte/eNet-sam7X-kit.php



SIMtrace - Daten zwischen SIM Karte und Handy aufzeichnen



Osmocom SIMtrace ist ein protocol analyzer für SIM/USIM Karten. Er wird zwischen SIM-Karte und Telefon geschaltet und leitet sämtliche Kommunikation über USB an einen geschlossenen PC weiter, wo eine übersichtliche Darstellung mittels wireshark erfolgt.

Firma : sysmocom - systems for mobile communications GmbH

http://shop.sysmocom.de/products/simtrace

CANlog2 - Datenlogger

Der Datenlogger dient der automatischen Aufzeichnung von CAN-Nachrichten auf einer SD-Card. Die Aufzeichnung erfolgt Identifier basiert. Die Dateinamen werden vom Datenlogger automatisch generiert und setzen sich aus Datum und Uhrzeit zusammen. Diese Informationen stehen intern über eine Echtzeituhr (RTC) zur Verfügung.

Firma : Wack Engineering

http://www.wack-engineering.de/

Haben Sie interessante Produkte oder Neuigkeiten?

Dann senden Sie uns ein Foto, 3-4 Zeilen Text und einen Link mit Betreff „Marktplatz“ an:

journal@embedded-projects.net

embedded - projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Bitte frei machen

embedded projects GmbH
Holzbachstraße 4
D - 86152 Augsburg

Name / Firma

Straße / Hausnummer

PLZ / Ort

Email / Telefon / Fax

◯ Ich möchte jede zukünftige Ausgabe erhalten

◯ Wir möchten als Hochschule / Ausbildungsbetrieb jede weitere Ausgabe bekommen. Bitte gewünschte Anzahl der Hefte pro Ausgabe ankreuzen. 5 10

◯ Ich möchten im embedded projects Journal werben oder eine Stellenanzeige aufgeben. Bitte schicken Sie mir Infomaterial, Preisliste etc. zu.

Werdet aktiv!

Das Motto: Von der Community für die Community !

Das Magazin ist ein Open Source Projekt.

Falls Du Lust hast, Dich an der Zeitschrift durch einen Beitrag zu beteiligen, würden wir uns darüber sehr freuen. Schreibe deine Idee an:

journal@embedded-projects.net

Regelmäßig

Die Zeitschrift wird über mehrere Kanäle verteilt. Der erste Kanal ist der Download als PDF - Datei. Alle Ausgaben sind auf der Internetseite [1] verfügbar. Diejenigen, die lieber eine Papierversion erhalten möchten, können den zweiten Kanal wählen. Man kann sich dort auf einer Internetseite [2] in eine Liste für die gesponserten Abos oder ein Spendenabo eintragen. Beim Erscheinen einer neuen Ausgabe wird dank Sponsorengeldern an jeden auf der Liste eine Ausgabe des aktuellen Journal versendet. Falls man den Versandtermin verpasst hat, kann man das Heft auch über einen Online - Shop [2] beziehen.

[1] Internetseite (Anmeldeformular gesponserte Abos): http://journal.embedded-projects.net

[2] Online - Shop für Journal:
http://www.embedded-projects.net

Sponsoren gesucht!

Damit wir weiterhin diese Zeitschrift für jeden frei bereitstellen können, suchen wir dringend Sponsoren für Werbe- und Stellenanzeigen. Bei Interesse meldet Euch bitte unter folgender Telefonnummer: 0821 / 2795990 oder sendet eine E-Mail an die oben genannte Adresse.

Impressum

embedded projects GmbH
Holzbachstraße 4
D-86152 Augsburg
Telefon: +49(0)821 / 279599-0
Telefax: +49(0)821 / 279599-20

Veröffentlichung: 4x / Jahr
Ausgabenformat: PDF / Print
Auflagen Print: 2500 Stk.
Einzelverkaufspreis: 1 €

Layout / Satz: EP
Druck: flyeralarm GmbH
Titelfoto: Claudia Sauter

BESSER GLEICH ONLINE KALKULIEREN.

STARRE UND FLEXIBLE LEITERPLATTEN.

ab 12 h
Leiterplatten-Express! Pünktlich oder gratis!

LEITON
RECHNEN SIE MIT BESTEM SERVICE

Endlich wird's einfach und Sie bleiben flexibel. Den umständlichen und aufwändigen Kalkulationsprozessen machen wir einen dicken Strich durch die Rechnung. Sie kalkulieren Ihre Leiterplatten online – ganz bequem, ganz schnell. **Einzigartig: Die Online-Kalkulation gilt auch für Serien und flexible Leiterplatten.** Das macht uns weltweit so schnell keiner nach. Einmalig ist zudem der **Leiterplatten-Expressdienst** von LeitOn mit unserer Garantie: Wenn wir bei Expressdiensten den vereinbarten Übergabetermin an den Versender nicht einhalten können, schenken wir Ihnen die bestellten Platinen! Sie möchten mehr darüber wissen? Wir bieten persönliche Beratung am Telefon und einen kompetenten Außendienst. Sie können bei LeitOn immer mit dem besten Service rechnen.

www.leiton.de | **kontakt@leiton.de** | Info-Hotline **+49 (0)30 701 73 49 0**