

# **USBprog Handbuch**

**<http://www.embedded-projects.net/usbprog>**

---

# USBprog Handbuch

<http://www.embedded-projects.net/usbprog>

---

---

---

## Table of Contents

Preface .....	viii
1. Einsatzgebiet und Funktionsumfang .....	1
2. Montage und Aufbau der Hardware .....	2
Löten der restlichen Bauteile .....	2
Anschlussmöglichkeiten und Stiftleisten von USBprog .....	4
Programmierung des Bootloaders .....	6
Erster Funktionscheck .....	9
3. USBprog Flashtool .....	12
Installation unter GNU/Linux .....	14
Installation aus den Quellen heraus .....	14
Installation unter OpenSUSE .....	15
Installation unter Debian/Ubuntu .....	16
Zugriff auf USBprog als Benutzer .....	16
Installation unter Windows .....	16
Installation unter MAC/OS .....	17
4. Bootloader (Konzept und Bedienungshinweise) .....	18
5. Firmware wechseln .....	21
6. Beliebte Fehlerquellen .....	22
7. Anwendungen mit USBprog .....	23
AVR ISP Programmer (STK500 kompatibel) .....	23
AVR Studio 4 .....	23
avrdude unter Windows .....	23
avrdude unter GNU/Linux .....	24
avrdude Bedienhinweise .....	25
Beliebte Fehlerquellen bei AVR ISP Programmer .....	25
OpenOCD ARM7/ARM9 Debugger .....	25
OpenOCD unter Linux .....	25
OpenOCD unter Windows .....	26
Arbeiten mit dem OpenOCD Debugger .....	26
SimplePort .....	26
Anschlussbelegung .....	27
Bibliothek in C .....	27
Beispiel in C .....	28
Beispiel in Java .....	28
Beispiel in Python .....	29
SimplePort RS232 .....	29
Kommandos für die Ansteuerung der Leitungen .....	30
Beispiel in Python .....	31
Einsatz in C# .....	31
USB zu RS232 Wandler .....	32
Status .....	32
Linux .....	32
MacOS .....	33
AT89 Programmer .....	33
Status .....	33
JTAG Adapter .....	33
XSVF Player (Xilinx CPLDs und FPGAs programmieren) .....	34
Anschlussbelegung .....	34
XSVF Player unter Linux .....	34

XSVF-Dateien erstellen mit Xilinx ISE 9.2i WebPack .....	35
Logik Analysator (250 kHz, 8 Signale, Trigger) .....	35
Anschlussbelegung .....	36
Downloads .....	37
Aufzeichnung von Messungen mit logic2vcd .....	37
Datenanalyse mit GTKWave .....	38
8. Eigene Firmware entwickeln .....	39
Schritt für Schritt .....	39
USBprog für Entwickler .....	39
A. Schaltplan .....	40
9. Apeendix B: Lizenzen .....	42
Index .....	58

---

## List of Figures

2.1. USBprog Bausatz .....	3
2.2. USBprog fertig montiert .....	3
2.3. USBprog Übersicht .....	4
2.4. VCC Konfiguration 1 (Ohne Verbindung) .....	5
2.5. VCC Konfiguration 2 (5V direkt über USB) .....	5
2.6. VCC Konfiguration 3 (5V per Schottky-Diode über USB) .....	5
2.7. USBprog Reset Jumper für Programmierung des Bootloaders .....	6
2.8. USBprog PinBelegung als AVR ISP Buchse .....	7
2.9. AVR Studio Fuse Einstellung (Bootloader) .....	8
2.10. PonyProg Fuse Einstellung (Bootloader) .....	8
2.11. Takteingang ATmega32 (Pad 4 von oben gezählt am ATmega32) .....	10
3.1. USBprog GUI .....	12
3.2. USBprog Konsole .....	13
4.1. USBprog Bootloader .....	18
4.2. USBprog Bootloader Jumper (Bootloader Start-Jumper) .....	18
7.1. GTKWave .....	36
A.1. Schaltplan USBprog 3.0 .....	40

---

## List of Tables

2.1. Parallelport Kabel BSD (avrude -c bsd) .....	7
2.2. Fuse Bits .....	9

---

# Preface

Im folgenden Handbuch wird die Inbetriebnahme und der Einsatz von USBprog beschrieben.

An dieser Stelle möchte ich mich bei den viele Helfern rund um USBprog bedanken. Ohne der Community wäre das Projekt nicht das was es mittlerweile ist.

Die meisten Texte aus diesem Handbuch stammen von Benedikt Sauter. Es haben unter anderem Texte für diese Handbuch geschrieben: Robert Schilling, Bernhard Walle und Sven Lütke.

## Kontakt

Embedded Projects Shop

Inh. Dipl.-Inf. (FH) Benedikt Sauter

Kettengässchen 6

D-86152 Augsburg

shop@embedded-projects.net

<http://www.embedded-projects.net>

## Support

Support gibt es im Forum auf <http://forum.embedded-projects.net/> oder per E-Mail [sauter@ixbat.de](mailto:sauter@ixbat.de).

## Lizenz

Dieses Handbuch wurde unter der freien Lizenz GFDL veröffentlicht. Dies bedeutet jeder hat das Recht den Text zu bearbeiten, muss ihn jedoch ebenfalls wieder unter der freien Lizenz GFDL veröffentlichen.

Copyright (c) 2007 Benedikt Sauter, Robert Schilling, Bernhard Walle, Sven Lütke

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".



---

# Chapter 1. Einsatzgebiet und Funktionsumfang

## Was ist USBprog

usbprog ist ein freier Programmieradapter. Über USB kann man bequem verschiedene Firmware Versionen aus einem "Firmware-Archiv" einspielen. Der Adapter kann aktuell für das Programmieren und Debuggen von AVR und ARM Prozessoren, als USB zu RS232 Wandler, als JTAG Schnittstelle oder als einfacher I/O-Port (10 Leitungen) eingesetzt werden.

## Wer braucht usbprog?

Hardware- und Softwareentwickler die unter Windows, Linux & Co arbeiten.

## Was ist das Besondere an dem Konzept von usbprog?

Mit der Zeit soll ein immer gösserer Pool an Firmware Versionen entstehen. Idealerweise kann man dann als Besitzer von usbprog, wenn man einen neuen Baustein bekommt mit dem man arbeiten soll, in den Online Pool schauen und suchen, ob es eine passende Firmware zum Programmieren oder Ansteuern von diesem Baustein gibt.

Sollte es gerade keine passende Firmware für den benötigten Baustein geben, so sollte man nicht überstürzt hohe Einkaufspreise für Adapter auf sich nehmen, sondern zuerst lieber anfragen, ob es lohnenswert ist, hier eine neue Firmware für usbprog zu schreiben, um somit in den Firmware-Pool zu investieren. Oft ist dies relativ schnell geschehen, da man auf bestehenden Softwarewerkzeuge aufbauen kann. Wie im Falle vom AVR ISP 2 Klon, diesen kann man mit allen Anwendungen nutzen die den originalen Adapter von Atmel unterstützen, oder der Adapter für den OpenOCD war auch ein Produkt, das innerhalb weniger Tagen entwickelt worden ist.

Da alles unter einer Open Source Lizenz steht, sollte das Projekt lange Zeit interessant bleiben!

## Auszug aus der Firmwareliste

- AVR ISP Programmer (kompatibel zu AVR ISP mkII)
- OpenOCD Interface (ARM Debugging)
- Freier JTAG Adapter + Bibliothek
- AT89 Programmer
- SimplePort (10 I/O Leitungen)
- USB zu RS232 Wandler (ohne Treiber!!)
- XSVF Player (Xilinx CPLD/FPGA)
- Logikanalysator

---

# Chapter 2. Montage und Aufbau der Hardware

Die folgende Übersicht gibt einen groben Leitfaden für die Installation an.

- Auspacken des Bausatzes (vormontierte Platine, 10-poliger Stecker, USB-Buchse, 2x4-polige Stiftleisten, 1x3-polige Stiftleiste, 3xJumper)
- USBprog fertig montieren (löten)

Abhängig davon ob ein bereits vorprogrammierter oder nichtvorprogrammierter Adapter gekauft worden ist, gibt es zwei mögliche Wege.

- USBprog ist bereits vorprogrammiert
  1. Benötigte Anwendung (z.B. AVR Studio oder Yagarto für OpenOCD installieren)
  2. USBprog Flash Tool installieren (nur optional wenn Firmware gewechselt werden soll)
- USBprog ist noch nicht programmiert
  1. VCC Jumper entfernen falls externer AVR Programmierer keine Stromversorgung benötigt
  2. Externen AVR Programmierer anstecken
  3. Reset Jumper Stecken
  4. usbprog\_base.hex flashen
  5. Fuse Bits programmieren
  6. Jumper entfernen
  7. USBprog Tools installieren (inkl. Treiber)
  8. Am PC anstecken -> Neues Gerät sollte erkannt werden (Treiber automatisch installieren)
  9. Per Flashtool gewünschte Firmware einspielen

Dies war nur ein grober Leitfaden. In den weiteren Abschnitten werden die einzelnen Schritte genauer beschrieben.

## Löten der restlichen Bauteile

Der USBprog Adapter 3.0 ist anders als die vorherige Version bereits vormontiert. Die SMD Bauteile sind bereits gelötet und nur die "groben" Bauteile müssen selbst angebracht werden.

Der Adapter wurde so geplant, dass Anstelle der USB-B-Buchse auch ein USB-A-Stecker eingelötet

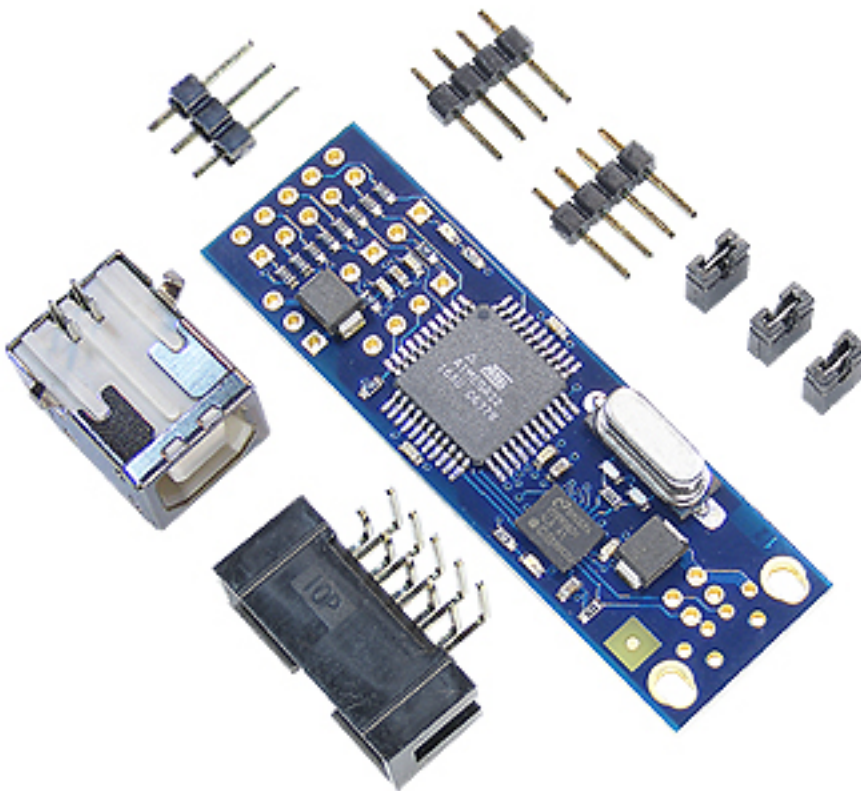
werden kann. Welches den Vorteil hat, dass USBprog in einem passendem Gehäuse als USB-Stick verwendet werden kann.

Folgende Schritte beim Löten sind somit zu erledigen:

- 10-polige Wannenbuchse einlöten
- USB Buchse montieren und einlöten
- 2x4-polige Stiftleiste fixieren und einlöten
- 3-polige Stiftleiste fixieren und einlöten

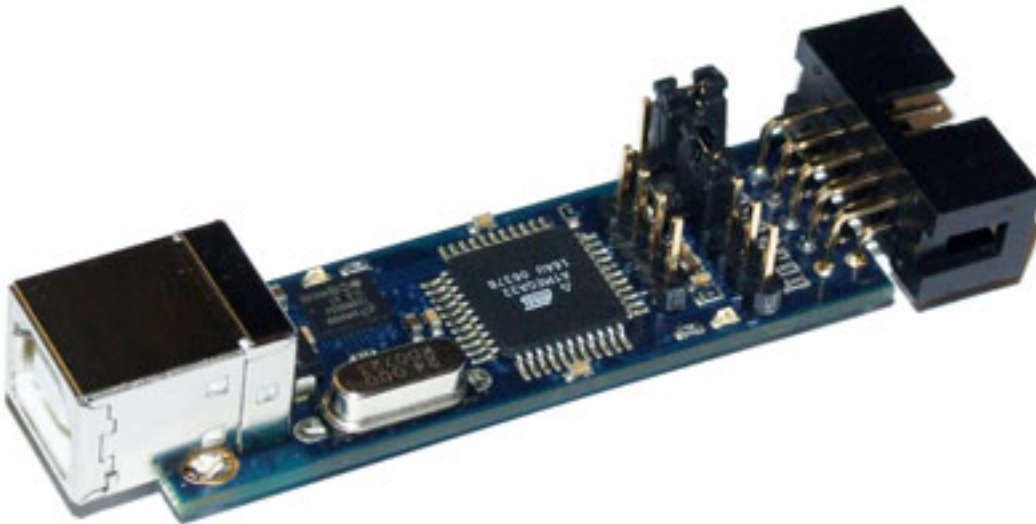
In der Abbildung USBprog Bausatz sind nochmal alle Bauteile gezeigt.

**Figure 2.1. USBprog Bausatz**



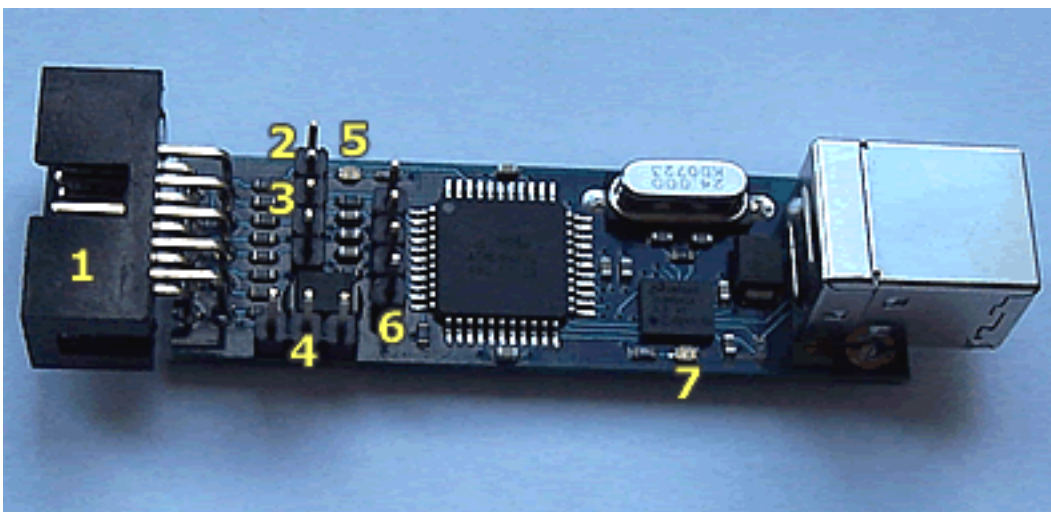
Nach den Lötarbeiten sollte USBprog nun wie folgt aussehen:

**Figure 2.2. USBprog fertig montiert**



## Anschlussmöglichkeiten und Stiftleisten von USBprog

Figure 2.3. USBprog Übersicht



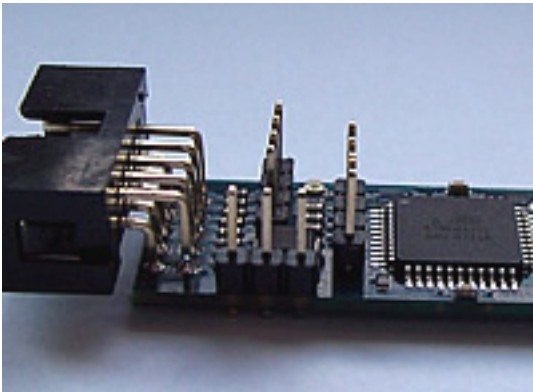
1. 8 IO Ports, (alternativ SPI) Port B vom ATMega32 + VCC und GND
2. Jumper kann als Schalter in einer Firmware verwendet werden (PA7)
3. Reset Jumper Ansteuerung (gesteckt ansteuerbar vom ISP Stecker)
4. VCC Spannungsversorgung Einstellung
5. LED Ansteuerbar aus den Firmware (PA4)
6. UART Anschluss + VCC und GND

## 7. Power LED

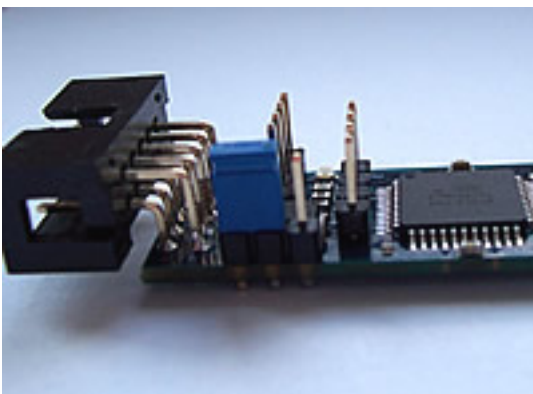
Mit dem VCC Jumper kann eingestellt werden, wie der VCC Pin am 10-poligen Stecker beschalten wird. Es werden drei Konfigurationen angeboten.

1. Ohne Verbindung
2. 5V direkt über USB
3. 5V per Schottky-Diode über USB

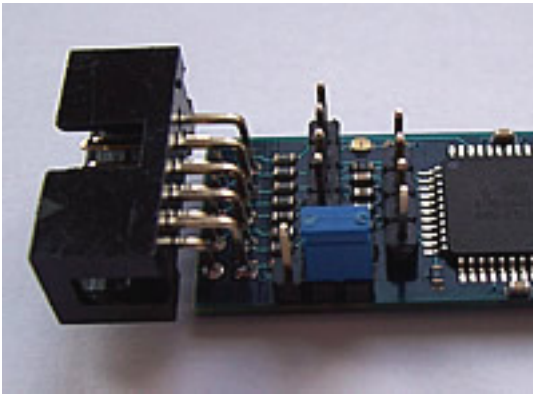
**Figure 2.4. VCC Konfiguration 1 (Ohne Verbindung)**



**Figure 2.5. VCC Konfiguration 2 (5V direkt über USB)**



**Figure 2.6. VCC Konfiguration 3 (5V per Schottky-Diode über USB)**



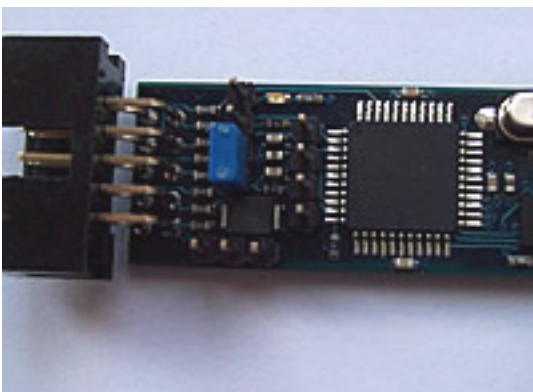
## Programmierung des Bootloaders

Der USBprog wird ohne programmierten Bootloader von keinem Betriebssystem erkannt. Das hat den einfachen Grund, da die Ansteuerung des USB-Bausteins in der Firmware des Bootloaders implementiert ist. Und wenn der Bootloader eben nicht im ATmega32 des USBprogs ist, hat das Betriebssystem keine Chance das Gerät zu erkennen.

Die Basis für USBprog ist daher der Bootloader, der es ermöglicht später ohne externen AVR Programmierer eine Firmware auf USBprog auszutauschen. Um den Bootloader auf USBprog übertragen zu können wird ein externer AVR Programmierer benötigt. Im folgenden werden die bekanntesten Möglichkeiten beschrieben.

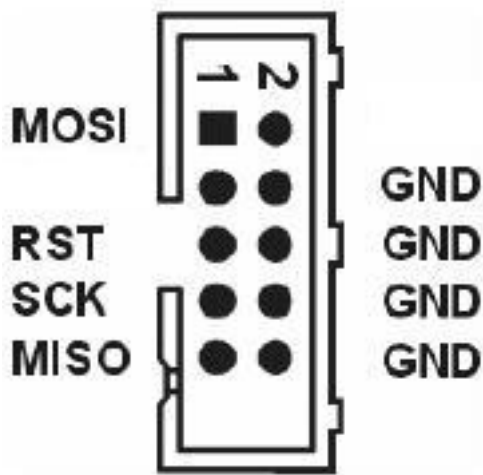
Die Datei `usbprog_base.hex` kann von der Internetseite heruntergeladen werden. Nun muss das letzte Mal mit einem externen Programmierer der ATmega32 auf der usbprog Platine programmiert werden. Dafür muss der Reset Jumper wie in der nachstehenden Abbildung beschrieben gesteckt werden. Jetzt kann das Programm normal eingespielt werden. Vielleicht verwundert die Dateigrösse von genau 32KByte (als .bin), dies kommt daher, dass der Bootloader `avrupdate` am Ende vom Flash Speicher hingeschrieben wird. Daher kann es etwas dauern bis `avrupdate` im ATmega32 ist.

### Figure 2.7. USBprog Reset Jumper für Programmierung des Bootloaders



Der Reset Jumper muss gesteckt sein, wenn der ATmega32 von extern programmiert werden soll.

Jetzt kann man sich mit einem Standard AVR-Programmiergerät mit usbprog verbinden. Die Belegung der 10-poligen ISP Buchse entspricht der Standardbelegung von Atmel.

**Figure 2.8. USBprog PinBelegung als AVR ISP Buchse**

### avrdude unter Linux und Windows

Falls die Firmware mit einem AVR ISP mkII (oder USBprog AVR Programmer) programmiert werden soll, muss als Parameter `-c avrispv2 -P usb` angegeben werden. Für ein einfaches Parallelportkabel reicht `-c bsd`.

- Firmware flashen: `avrdude -p m32 -c avrispv2 -P usb -B 8 -U flash:w:usbprog_bootloader.hex`
- Ifuse programmieren: `avrdude -p m32 -c avrispv2 -P -B 8 -U lfuse:w:0xe0`
- hfuse programmieren: `avrdude -p m32 -c avrispv2 -P -B 8 -U hfuse:w:0xd8`

Wird ein einfaches Parallelportkabel verwendet muss folgende Belegung gewählt werden:

**Table 2.1. Parallelport Kabel BSD (avrude -c bsd)**

Parallel Port	Programmer Function
Pin 7	AVR !Reset
Pin 8	AVR SCK (clock input)
Pin 9	AVR MOSI (instruction in)
Pin 10	AVR MISO (data out)
Pin 18	GND

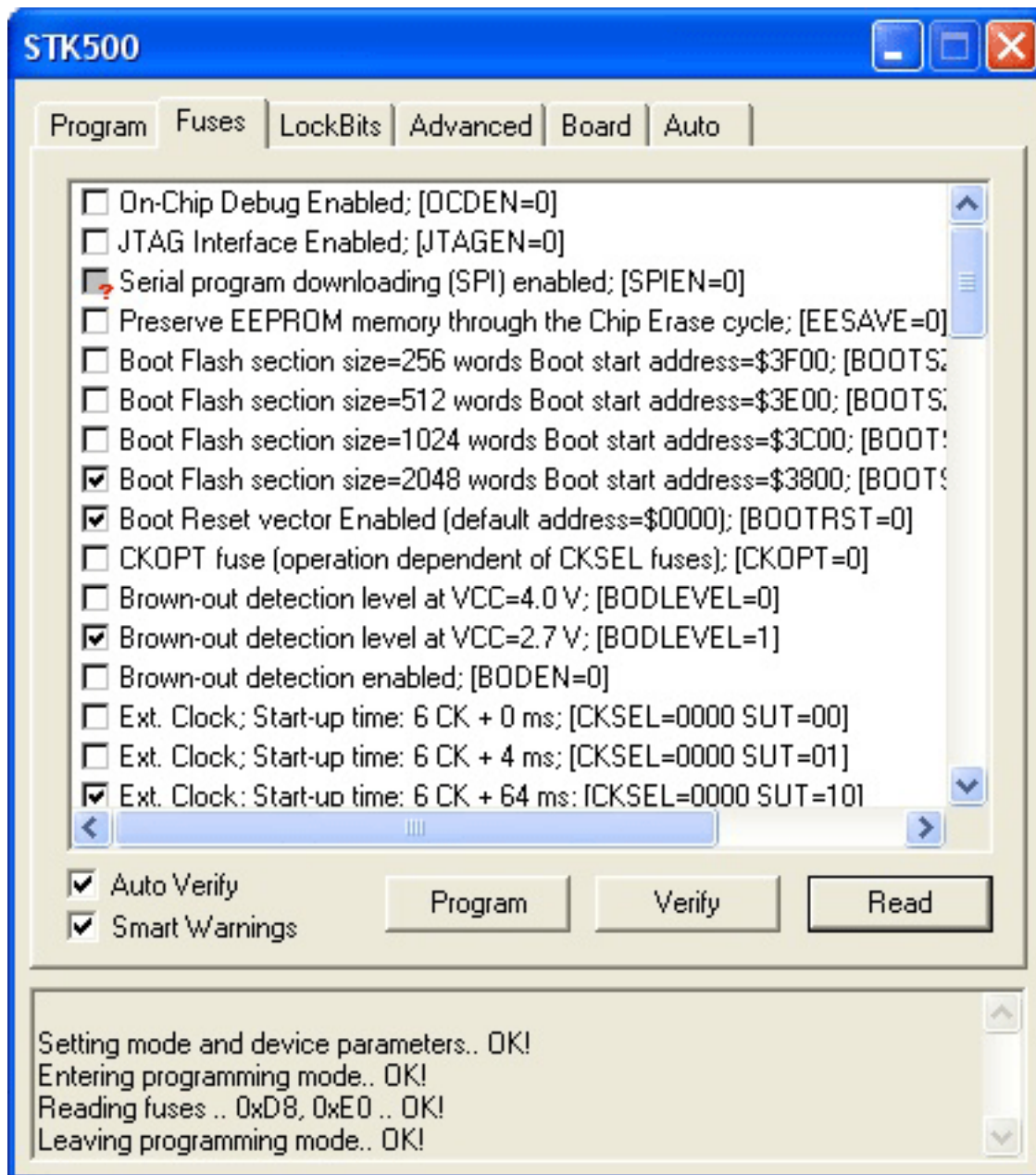
### AVR Studio

Falls ein Programmiergerät zum programmramieren des Bootloaders in USBprog eingesetzt wird, das vom AVR Studio unterstützt wird kann wie in der folgenden Abbildung die Fuse eingestellt werden. Als Zielprozessor muss der Typ ATmega32 angegeben werden. Ebenso muss die ISP Geschwindigkeit auf ca.



250 kHz gestellt werden, um sicherzustellen dass die ISP Schnittstelle mit max. 1/4 des CPU Taktes angesteuert wird.

**Figure 2.9. AVR Studio Fuse Einstellung (Bootloader)**



### PonyProg

Wird ein PonyProg kompatibles Programmiergerät für die Erstprogrammierung des USBprogs mit dem Bootloader verwendet sieht die Fuse-Einstellung wie folgt aus:

**Figure 2.10. PonyProg Fuse Einstellung (Bootloader)**



**Configuration and Security bits**

☐ 7 ☐ 6 ☐ BootLock12 ☐ BootLock11 ☐ BootLock02 ☐ BootLock01 ☐ Lock2 ☒ Lock1

☐ OCDEN ☐ JTAGEN ☒ SPIEN ☐ CKOPT ☐ EESAVE ☒ BOOTSZ1 ☒ BOOTSZ0 ☒ BOOTRST

☐ BODLEVEL ☒ BODEN ☐ SUT1 ☒ SUT0 ☒ CKSEL3 ☒ CKSEL2 ☒ CKSEL1 ☒ CKSEL0

☒ Checked items means programmed (bit = 0) ☐ UnChecked items means unprogrammed (bit = 1)

Refer to device datasheet, please

## Tabelle

**Table 2.2. Fuse Bits**

Nr.	Fuse	Wert	Beschreibung
1	BODLEVEL	1	Brown-Out-Detection
2	BODEN	0	keine Brown-Out-Detection
3	SUT0	0	Startup-Time=6 CLK + 64 ms
4	SUT1	1	Startup-Time=6 CLK + 64 ms
5	CKSEL3..0	0000	externer Takt
6	CKOPT	1	externer Takt
7	OCDEN	1	On-Chip-Debug disable - wichtig wegen Port C (USBN9604)
8	JTAGEN	1	JTAG aus
9	SPIEN	0	SPI an
10	EESAVE	1	Chip Erase löscht auch EEPROM
11	BOOTSZ1..0	00	Boot start address = \$3800
12	BOOTRST	0	Boot Reset

## Erster Funktionscheck

Nach dem flashen des Bootloader und dem einstelle der FUSE Bits kann USBprog bereits vom Betriebssystem erkannt werden.

Es empfiehlt sich die Jumper in folgende Grundstellung zu bringen:

1. Resetjumper entfernen. Der Jumper wird wirklich nur für das programmieren des Bootloaders in den USBprog benötigt.
2. Bootloader Jumper entfernen. Nach dem programmieren des Bootloaders befindet sich noch keine Firmware auf USBprog und es wird automatisch der Bootloader gestartet.
3. VCC Jumper entfernen. Hiermit kann gesteuert werden wie die nach aussen geführte VCC Leitung beschalten werden soll, also ob die Zielschaltung über USBprog versorgt werden soll. Um sicher zu stellen, dass es keinen Kurzschluss oder ähnlich gibt sollte der VCC Pin keinen Kontakt haben. Das heisst der Jumper sollte keine Verbindung herstellen.

Mit dem Auge kann geprüft werden, ob der Bootloader starten kann. Dies kann am Blinkrythmus der Status LED gesehen werden. Die LED blinkt im Rythmus: kurz an - kurz aus - kurz an - lang aus.

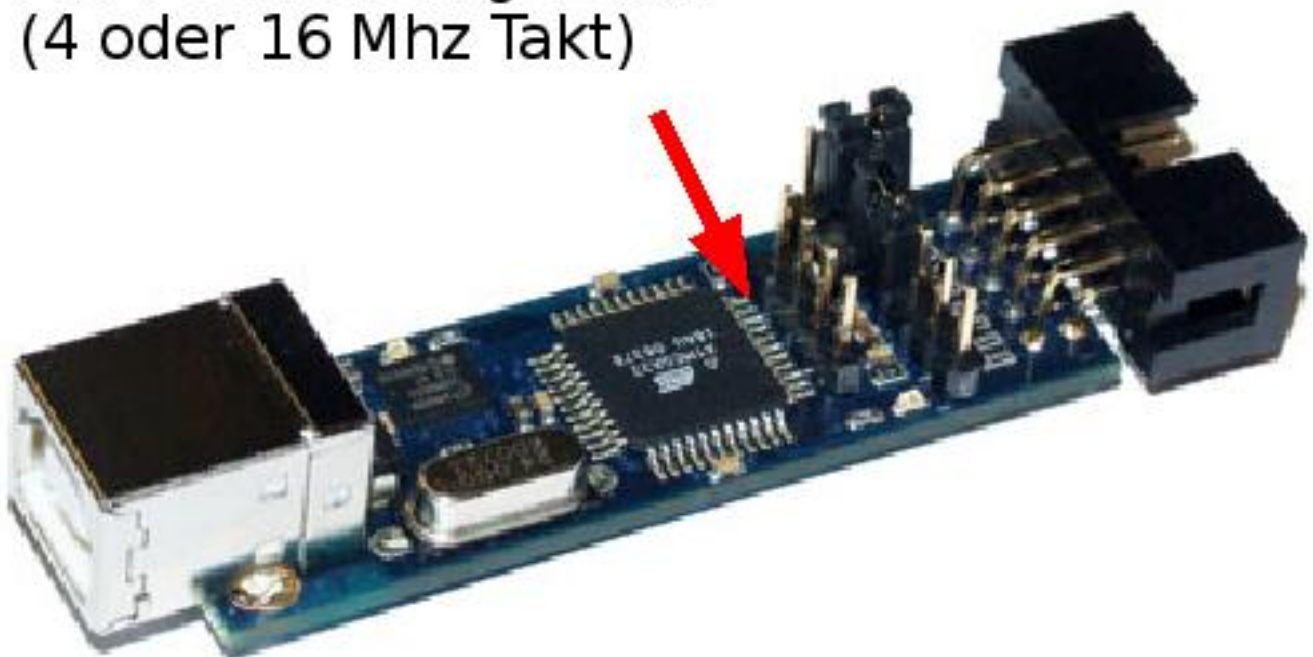
Wenn die LED entsprechend blinkt, bedeutet dies nur, dass der ATmega32 korrekt arbeitet. Ob die USB-Schnittstelle korrekt arbeitet kann aber nur mit einem Computer überprüft werden.

Blinkt die LED trotz einspielen des Bootloaders und einstellen der Fuse Bits nicht, kann mit einem Oszilloskop fest gestellt werden ob die Schaltung denn irgendetwas macht, oder ob ein grösseres Problem vorliegt.

Am einfachsten misst man dafür die Frequenz am Taktausgang des USB-Bausteins, welcher als Taktquelle für den ATmega32 dient und an dieser Seite auch einfacher gemessen werden kann (Da der USB-Baustein von unten angelötet worden ist, und man daher nicht direkt an die Pads kommt).

**Figure 2.11. Takteingang ATmega32 (Pad 4 von oben gezählt am ATmega32)**

Pin 4 von oben gezählt  
(4 oder 16 Mhz Takt)



Befindet sich kein Bootloader im ATmega32, oder ist die Kommunikation zwischen dem USB-Baustein

und dem ATmega32 gestört sollten dort 4 MHz anliegen. Ist der Bootloader aktiv so kann dies anhand der anliegenden 16 MHz festgestellt werden. Das Umschalten der Taktfrequenz von 4 MHz auf 16 MHz klappt nur dann wenn die Verbindung zwischen USB Baustein und ATmega32 in Ordnung ist und der Bootloader korrekt gestartet ist.

### GNU/Linux

Mit Hilfe des Kommandos `lsusb` aus den `usbutils` (Über die Paketverwaltung muss das Paket `usbutils` zuvor installiert worden sein) geprüft werden, ob USBprog mit dem Bootloader vom Betriebssystem erkannt wird.

```
big:/home/bene# lsusb
Bus 002 Device 035: ID 1781:0c62
```

Dies ist die Kennung der Bootloader (1781 = Hersteller ID, 0c62 = Produktnummer)

### Windows

In Windows sollte der typische Ping Pling Sound hörbar sein und der Treiber Installationsassistent nach dem Ort des Treibers fragen. An dieser Stelle sollte man abbrechen, falls noch nicht die USBprog Tools installiert worden sind.

---

# Chapter 3. USBprog Flashtool

Mit dem USBprog-Tool kann die Firmware des USBprog-Geräts einfach gewechselt werden. Es unterstützt

- das Herunterladen der Firmware von einem Online-Pool,
- einen Offline-Modus (Cache) zum Betrieb an PCs ohne Internet,
- das Hochladen von lokalen Firmwaredateien zum Testen,
- mehrere USBprog-Geräte an einem PC und
- das Anzeigen von Informationen über eine Firmware, inkl. der Pinbelegung.

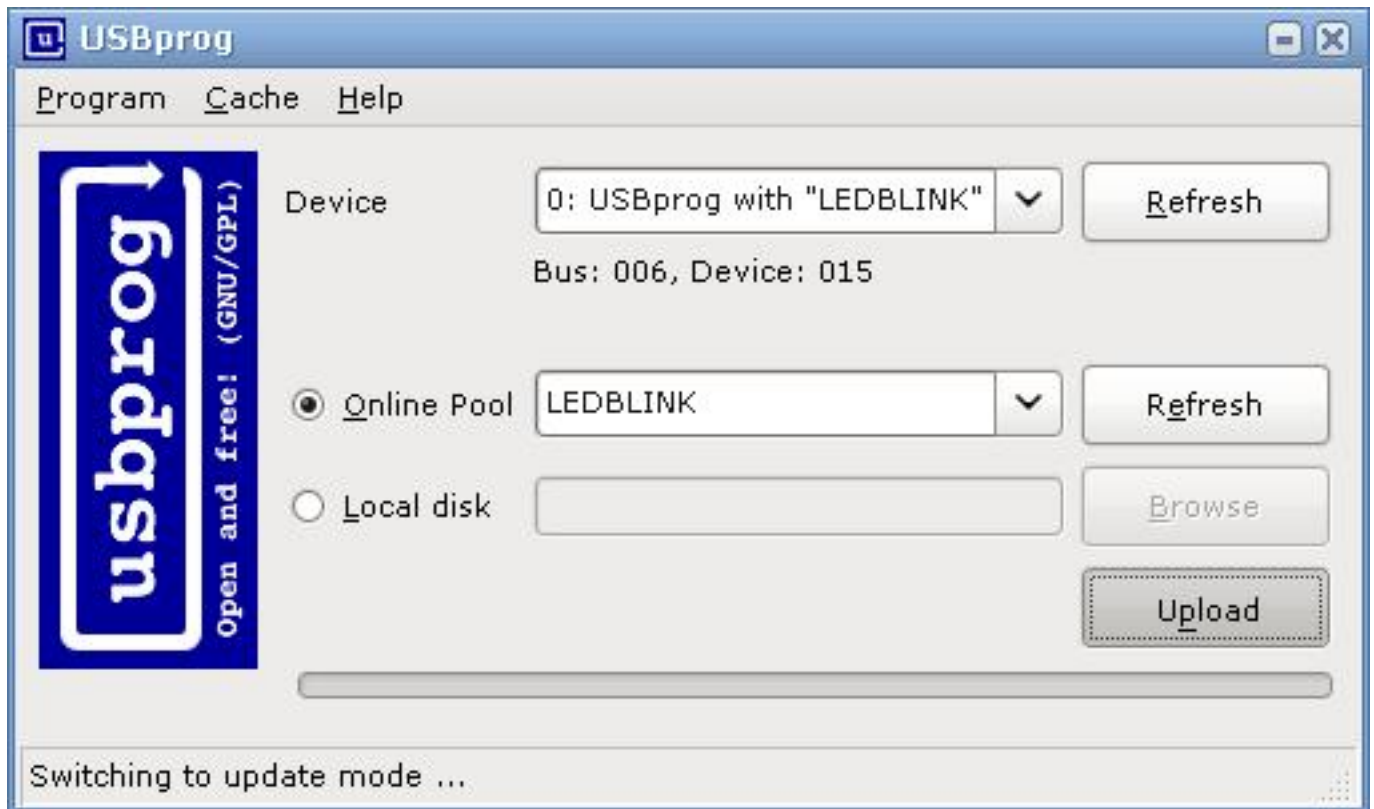
Von USBprog gibt es sowohl eine GUI-Version als auch eine Kommandozeilen-Version. Die Kommandozeilen-Version verfügt über einen interaktiven Modus (wie eine Shell) und einen Batch-Modus, kann daher problemlos in Skripte, Makefiles o.ä. integriert werden.

Beide Versionen verwenden die gleichen Funktionen, um auf das Gerät zuzugreifen und um den Firmware-Cache zu verwalten. Dies wurde damit realisiert, dass diese Funktionen in einer Bibliothek gekapselt sind. Somit wird die Konsistenz beider Versionen gewahrt und die Entwicklung vereinfacht.

Sowohl die GUI-Version als auch die Kommandozeilen-Version laufen unter Microsoft Windows und Linux. Eine Portierung auf anderen Unix-artige Betriebssysteme sollte sehr einfach möglich sein, wenn auch den Autoren die Zeit fehlt, dies zu testen.

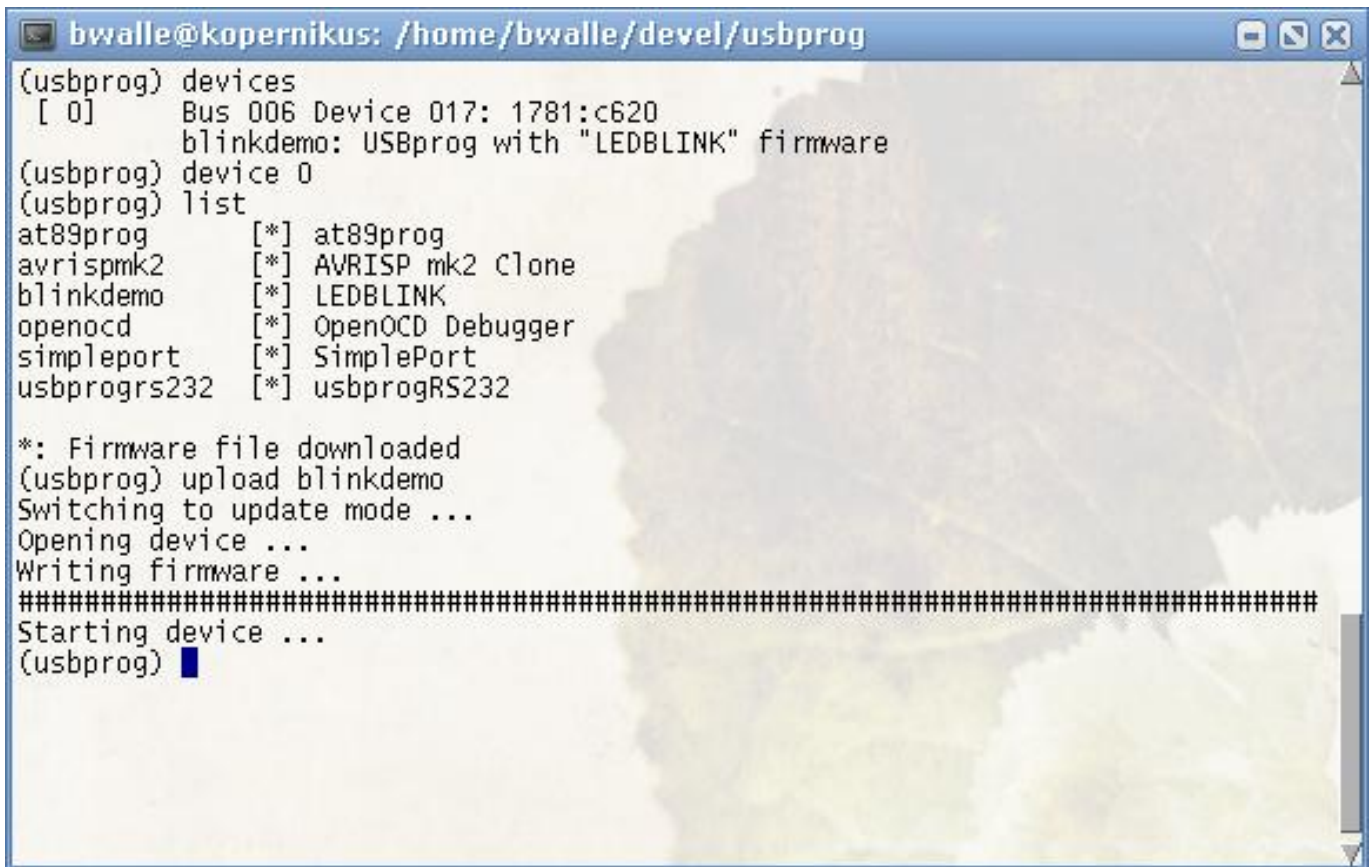
Da ein Bild bekanntlich mehr als tausend Worte sagt, hier ein Screenshot sowohl der GUI- als auch der Kommandozeilen-Version, um einen Eindruck der Applikation zu bekommen. GUI-Version (unter Linux)

## Figure 3.1. USBprog GUI



Kommandozeilenversion (unter Linux)

**Figure 3.2. USBprog Konsole**

A screenshot of a terminal window titled 'bwalle@kopernikus: /home/bwalle/devel/usbprog'. The terminal shows the following commands and output:

```
(usbprog) devices
[ 0]      Bus 006 Device 017: 1781:c620
        blinkdemo: USBprog with "LEDBLINK" firmware
(usbprog) device 0
(usbprog) list
at89prog      [*] at89prog
avrispmk2     [*] AVRISP mk2 Clone
blinkdemo     [*] LEDBLINK
openocd       [*] OpenOCD Debugger
simpleport     [*] SimplePort
usbprogrs232  [*] usbprogRS232

*: Firmware file downloaded
(usbprog) upload blinkdemo
Switching to update mode ...
Opening device ...
Writing firmware ...
#####
Starting device ...
(usbprog) █
```

## Installation unter GNU/Linux

Linux ist nicht gleich Linux. Das bedeutet die Installation ist stark abhängig von der eingesetzten Distribution.

## Installation aus den Quellen heraus

Zunächst sollte sichergestellt werden, dass das System folgende Abhängigkeiten erfüllt:

- libusb zum Zugriff auf das Gerät,
- libxml zum Parsen der Firmwarebeschreibungsdatei,
- libcurl zum Downloaden der Firmware,
- readline zum einfachen Editieren der Kommandozeile (optional),
- wxWidgets, falls die GUI-Version kompiliert werden soll und natürlich
- ein hinreichend neuer C++-Compiler wie den g++.

Nachdem dies sichergestellt wurde, erfolgt die Installation ganz üblich mit:

1. Auspacken des Tarballs und Wechseln in das Verzeichnis,
2. Ausführen von `./configure`,
3. `make` zum Kompilieren und schließlich
4. `make install` zum Installieren der Dateien (als Administrator).

## Installation unter OpenSUSE

Distributionsunabhängige Binärpakete unter Linux sind mit einem relativ großen Aufwand verbunden (praktisch alle Bibliotheken müssten statisch hinzugebunden werden) und werden von uns nicht angeboten. Die Installation aus dem Quellcode ist hinreichend einfach — und die Zielgruppe von USBprog sind ja letztendlich Entwickler.

Allerdings ist unser Ziel die Aufnahme in alle verbreiteten Distributionen. Pakete für openSUSE finden sich im openSUSE Build Service unter <http://download.opensuse.org/repositories/electronics/>. Das entsprechende Repository kann dem Paketmanager (YaST oder smart) hinzugefügt werden, somit kann mit relativ wenig Aufwand sichergestellt werden, dass immer die neuste USBprog-Version auf dem System installiert ist.

Beispiel für zypper (openSUSE 10.3)

```
$ zypper ar -r \
http://download.opensuse.org/repositories/electronics/openSUSE_10.3/electronics.repo
$ zypper ref
$ zypper install usbprog usbprog-gui
```

Das Angebot für openSUSE hängt schlicht damit zusammen, dass der Autor des USBprog-Programms bei SUSE arbeitet und soll keine Präferenz für eine Distribution darstellen. Von uns werden alle verbreiteten Distributionen unterstützt.

Wenn Sie in der Lage sind, Pakete für eine (verbreitete) Distribution zu bauen und auf Dauer zu pflegen, melden Sie sich bitte auf der USBprog-Mailingliste.

**Zugriffsrechte** Standardmäßig kann nur der Administrator unbeschränkt auf USB-Geräte zugreifen. Damit dies auch dem einfachen Benutzer möglich wird, müssen udev und HAL entsprechend konfiguriert werden, die jeweiligen Dateien unter `/dev/bus/usb` auch für normale Benutzer schreibbar zu machen.

Unter openSUSE reicht das Anlegen einer Datei `/etc/hal/fdi/policy/20customized` mit folgendem Inhalt:

```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
  <!-- USBprog in update mode or various demos -->
  <device>
    <match key="info.bus" string="usb_device">
      <match key="usb_device.vendor_id" int="0x1781">
        <match key="usb_device.product_id" int="0x0c62">
          <merge key="resmgr.class" type="string">scanner</merge>
        </match>
      </match>
    </match>
  </device>
  <!-- Atmel AVR ISP MKII -->
  <device>
    <match key="info.bus" string="usb_device">
      <match key="usb_device.vendor_id" int="0x03eb">
```

```
<match key="usb_device.product_id" int="0x2104">  
  <merge key="resmgr.class" type="string">scanner</merge>  
</match>  
</match>  
</device>  
</deviceinfo>
```

Damit werden USBprog-Geräte wie Scanner behandelt, was entsprechend bedeutet, dass der gerade (direkt, nicht über SSH) eingeloggte Benutzer auf das Gerät zugreifen darf.

## Installation unter Debian/Ubuntu

TDB

## Zugriff auf USBprog als Benutzer

Ohne ein weiteren Eingriff kann USBprog jetzt nur von root geöffnet und benutzt werden. Soll der Zugriff aber von einem anderen Benutzer aus möglich sein, muss dies entsprechend eingestellt werden, dass direkt nach dem anstecken von USBprog die Rechte entsprechend gesetzt werden. Eine Möglichkeit ist dies über das udev System zu machen. Dafür müssen folgende Schritte als root vollzogen werden:

- Neue Gruppe anlegen: *addgroup usbprog*
- Benutzer der Grupper zuweisen: *adduser benutzer usbprog* (umbedingt den Benutzer danach ein- und ausloggen)
- Öffnen und erweitern der Datei: */etc/udev/rules.d/80-usbprog.rules*
- Neustarten von udev: */etc/init.d/udev restart*

```
ATTRS{idVendor}=="1781", ATTRS{idProduct}=="0c62", GROUP="usbprog", MODE="0660"
```

Voraussetzung ist natürlich, dass udev installiert ist. In Debian geht dies per *apt-get install udev*.

Jetzt sollte der Zugriff mit jedem Benutzer der in der Gruppe usbprog ist funktionieren.

## Installation unter Windows

Die USBprog-Software läuft unter Windows 2000 und XP. Die DOS-basierten Windows-Versionen 95, 98 und ME werden nicht unterstützt. Da ein Installer verwendet wird, ist die Installation unter Windows denkbar einfach:

1. Laden Sie die Datei aktuelle USBprog-Version herunter.
2. Führen Sie die Installationsdatei aus und folgen Sie den Anweisungen. Beachten Sie, dass das Treiberpaket zwingend notwendig ist.
3. Beim Anstecken des USBprog sollte ihn Windows im "Updatemodus" erkennen und beginnt mit der Plug-&-Play-Treiberinstallation.



4. Wählen Sie bei der Treiberinstallation die Option Software automatisch installieren.
5. Nun kann USBprog mit Hilfe des Flashtools aktualisiert werden.

Achtung:

Wechseln in den Update-Modus! Unter Windows sollte vor dem Wechseln immer manuell in den Update-Modus gebracht werden! Mehr dazu gibt es im Kapitel der Bootloader.

## Installation unter MAC/OS

folgt (bitte im Forum nachfragen)

---

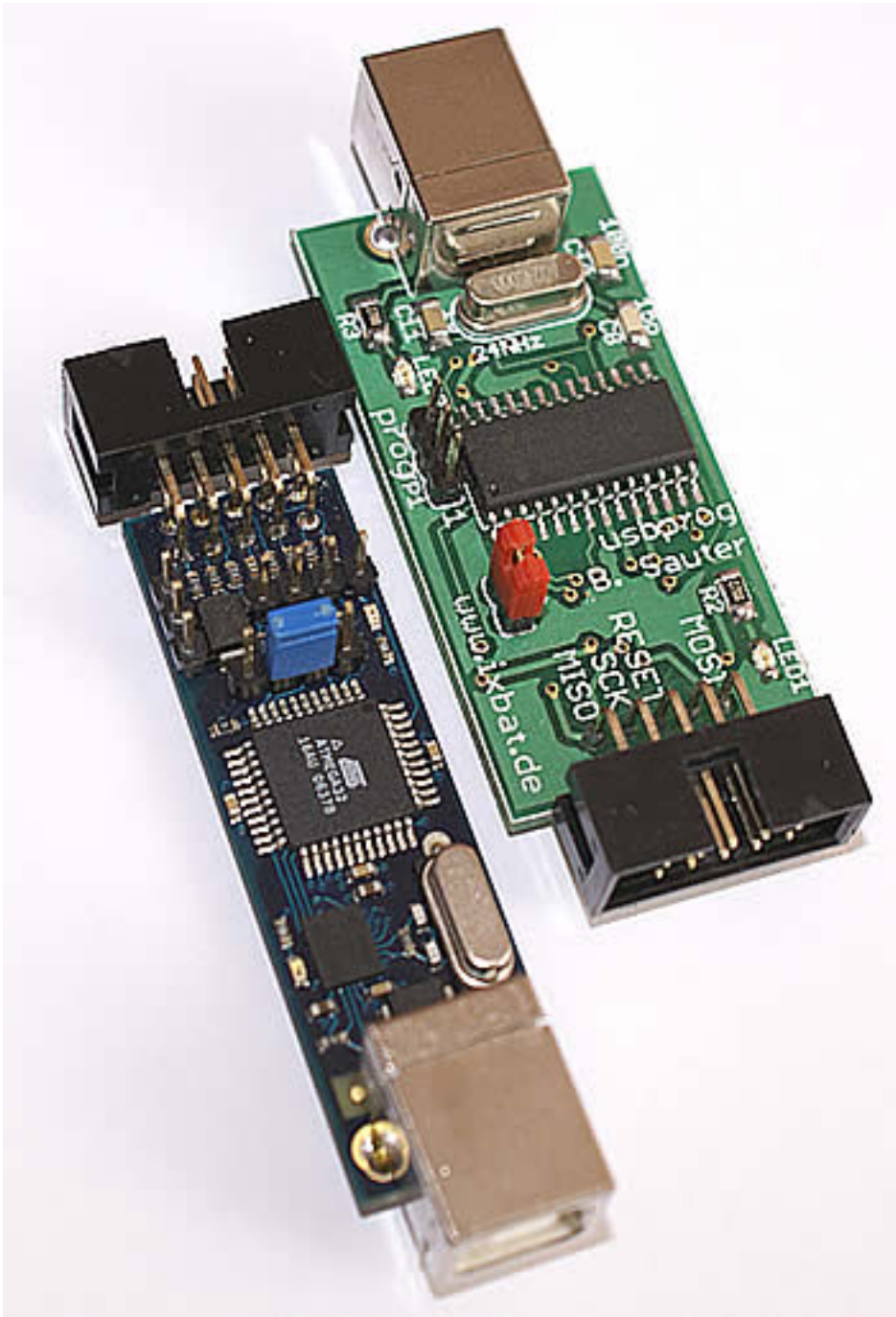
# Chapter 4. Bootloader (Konzept und Bedienungshinweise)

Der Bootloader bietet die Plattform für das einfache Austauschen der Firmware an. Da es mit der ersten Version des Bootloaders unter Windows immer wieder Probleme gab, wurde der neue Bootloader weiterentwickelt.

## Figure 4.1. USBprog Bootloader

Ursprünglich sollte der Bootloader einfach bequem per PC-Programm aktiviert werden können. Unter GNU/Linux klappt das auch einwandfrei. Mit Windows gibt es leider viele Probleme. Das liegt wohl am Treiberstack von Windows. Unter GNU/Linux kann man beispielsweise mit der Bibliothek libusb ohne extra Treiberinstallation auf jedes angeschlossene USB-Gerät zugreifen. In Windows muss man, wenn man mit der Bibliothek libusb arbeitet, dennoch einen Treiber für jedes Gerät unter der libusb installieren. Und das war eine beliebte Quelle für Fehler. Oft erkennt Windows nicht, welchen Treiber es laden muss, wenn während des Betriebs die Firmware und somit die Bezeichnung für das USB-Gerät gewechselt hat. Aus diesem Grund kann man jetzt den Bootloader manuell per Jumper starten. Dadurch kommt es zu diesem Problem nicht mehr, dass während der Session eines Geräts die Kennung wechselt und Windows einen Treiber umladen muss. Im nächsten Abschnitt wird erklärt, wie der Bootloader verwendet wird.

## Figure 4.2. USBprog Bootloader Jumper (Bootloader Start-Jumper)



Der Bootloader kann jetzt mittels eines Jumpers gestartet werden. Ist beim Anstecken von USBprog ein Jumper zwischen TX und RX, wird der Bootloader gestartet. Wenn der Bootloader aktiv ist, blinkt die Leuchtdiode in einem bestimmten Rythmus: ..lang aus - kurz an - kurz aus - kurz an - kurz aus - lang aus.

Nach dem der Bootloader gestartet worden ist, kann entweder mit dem grafischen oder konsolen Programm die Firmware ohne Probleme gewechselt werden. Der Jumper sollte danach wieder entfernt werden, da sonst immer erst der Bootloader und nicht die frisch eingespielte Firmware startet.

#### **Linux:**

Unter Linux kann der Bootloader weiterhin ohne den Jumper, also per USB gestartet werden.

#### **Windows:**

Mit einigen Windowsinstallationen kann ebenfalls ohne Jumper, also per USB der Bootloader gestartet werden (mittels Flashtools). Aber bei vielen Windowsversionen macht dies nur Probleme. Bei diesen Versionen sollte immer so vorgegangen werden:

- erst Jumper setzen
- dann USBprog anstecken
- mit einem Flashtool eine Firmware auswählen und übertragen
- USBprog abstecken
- Jumper entfernen
- USBprog anstecken

#### **Hinweis UART Anschluss und Bootloader**

Die Leitungen TX und RX können dennoch weiterhin in Firmwares (wie z.B. im USB zu RS232 Wandler) verwendet werden. Beim Starten sind die beiden Leitungen als UART aktiv, das heisst mit dem Setzen des Jumpers baut man sich sozusagen eine Nullmodem Kabelverbindung. Der Bootloader überträgt ein bestimmtes Zeichen und prüft ob er das auch wieder empfangen hat, wenn das der Fall ist, wird der USB Bootloader Modus aktiviert.

---

# Chapter 5. Firmware wechseln

Nach der Installation sollte das Gerät zunächst getestet werden. Hierzu bietet sich die “blinkdemo” Firmware an. Die Bedienung der GUI-Version sollte hinreichend selbsterklärend sein. Gestartet wird die Software unter Windows über das entsprechende Icon, unter Linux über die Programmdatei `usbprog-gui`.  
Kommandozeilenversion

1. Starten Sie die Kommandozeilenversion in einem Terminal mit dem Aufruf von `usbprog`.
2. Das Programm lädt nun online eine Indexdatei herunter. Lassen Sie sich mit `list` eine Liste aller verfügbaren Firmwaredateien anzeigen.
3. Laden Sie mit `download blinkdemo` die oben erwähnte “blinkdemo” Firmware herunter.
4. Zeigen Sie sich mit `devices` alle verfügbaren USBprog-Geräte an. Mindestens ein Gerät sollte in der Liste enthalten sein. Falls das USBprog-Device bereits im Updatemodus ist, sollte dies in der Liste bereits mit einem Stern markiert sein. Andernfalls geben Sie manuell mit `device 0` an, dass das erste Gerät als Updategerät verwendet werden soll.
5. Laden Sie nun mit `upload blinkdemo` die “blinkdemo” Firmware auf das Gerät. Die Firmware sollte automatisch starten.

## Hilfe

Das Kommandozeilenprogramm gibt mit dem Kommando `help` eine Liste aller verfügbaren Kommandos aus. Mit `helpcmd` Kommando kann man sich spezifische Hilfe zu einem Kommando ausgeben lassen.

Unter Linux sind zu beiden Programmen (also `usbprog` und `usbprog-gui`) Manpages installiert. Diese erhalten ebenfalls eine Kurzzusammenfassung der entsprechenden Befehle und Optionen.

---

# Chapter 6. Beliebte Fehlerquellen

- Schlechte Lötstellen an der USB-Buchse
- Falsche Konfiguration der FUSE Bits
- Firmware falsch in den Flash des Mikrocontrollers übertragen
- Flashtool 0.2 mit neuem Bootloader
- falsche ISP Geschwindigkeit eingestellt (max. 1/4 des CPU Taktes)
- 24 MHz Grundtonquarz
- Reset Jumper vergessen zu setzen oder zu entfernen

---

# Chapter 7. Anwendungen mit USBprog

Im folgenden sind die wichtigsten Firmwareversionen hier beschrieben. Da dieses Handbuch nicht immer auf dem neusten Stand ist, empfiehlt es sich bei Problemen auf den Internetseiten von Embedded Projects mögliche Hilfen zu suchen.

## AVR ISP Programmer (STK500 kompatibel)

Der AVR ISP Programmer wurde nach den freigegeben Datenblättern von Atmel entwickelt. Daher kann er mit allen Anwendungen die den AVR ISP mkII unterstützen genutzt werden.

Die bekanntesten Anwendungen sind:

- AVR Studio 4
- avrdude (GNU/Linux, Windows, MacOS)
- CodeVisionAVR <http://www.hpinfotech.ro/html/download.htm>

Im folgenden wird kurz erklärt was bei der Installation und Verwendung vom AVR Studio und avrdude wichtig zu beachten ist.

## AVR Studio 4

Das AVR Studio liefert eigene Treiber für den originalen AVR ISP Programmer mit. Bei der Installation muss jedoch explizit angegeben werden, dass diese auch installiert werden. Ist das AVR Studio bereits installiert, kann nachträglich über die Systemsteuerung -> Software -> Reparieren der Treiber aktiviert werden.

Steckt man den USBprog mit der AVR Programmer Firmware an, so sollte Windows automatisch den AVR Studio Treiber (Jungo) finden und aktivieren.

## avrdude unter Windows

Das bekannte Programm avrdude aus der Linux-Welt kann ebenfalls in Windows genutzt werden. Es befindet sich im Archiv von WinAVR. Das bedeutet es muss zuerst WinAVR aus dem Internet heruntergeladen und installiert werden.

### Verwendung des AVR Studio Treibers

Nachdem WinAVR Installiert ist, muss man dafür sorgen das Windows einen Treiber für USBprog mit dem AVR Programmer hat. Entweder verwendet man den originalen AVR Studio Treiber, dafür muss man aber das AVR Studio wie im Absatz AVR Studio 4 beschrieben installieren, oder man installiert einen freien libusb Treiber der bereits im WinAVR Paket mitgeliefert wird.

### Verwendung des freien libusb Treibers

Alternativ zum originalen Treiber kann auch der freie libusb Treiber installiert werden. Wenn der Windowsassistent sich öffnet muss gewählt werden, dass der Pfad zum Treiber manuell angegeben wird.

Die Treiberdateien befinden sich nach der Installation von WinAVR im Verzeichnis  
c:\WinAVR\utils\libusb.

## avrdude unter GNU/Linux

Entweder kann avrdude über die Paketverwaltung installiert werden oder es wird der klassische Linux Installationsprozess gewählt.

Installation per APT (Debian/Ubuntu/etc) (als root ausführen)

```
rechner:/home/sauter# apt-get install avrdude
Reading package lists... Done
Building dependency tree... Done
Suggested packages:
  avrdude-doc
The following NEW packages will be installed:
  avrdude
0 upgraded, 1 newly installed, 0 to remove and 47 not upgraded.
Need to get 0B/154kB of archives.
After unpacking 700kB of additional disk space will be used.
Selecting previously deselected package avrdude.
(Reading database ... 66442 files and directories currently installed.)
Unpacking avrdude (from .../avrdude_5.2-2_i386.deb) ...
Setting up avrdude (5.2-2) ...
```

### Klassische Linux Installation

Für die Installation muss das neueste Quelltextarchiv zuvor heruntergeladen werden. Die aktuellste Version kann von der Internetseite <http://download.savannah.gnu.org/releases/avrdude/> ausgewählt werden.

```
sauter:/home/sauter# cd /usr/src/
sauter:/usr/src# wget http://download.savannah.gnu.org/releases/avrdude/avrdude-5.5.tar.gz
--13:11:10-- http://download.savannah.gnu.org/releases/avrdude/avrdude-5.5.tar.gz
=> `avrdude-5.5.tar.gz'
Resolving download.savannah.gnu.org... 199.232.41.75
Connecting to download.savannah.gnu.org|199.232.41.75|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 453,614 (443K) [application/x-gzip]

100%[=====>] 453,614      211.89K/s

13:11:13 (211.44 KB/s) - `avrdude-5.5.tar.gz' saved [453614/453614]

sauter:/usr/src# tar xzf avrdude-5.5.tar.gz
sauter:/usr/src# cd avrdude-5.5
sauter:/usr/src/avrdude-5.5# ./configure
....
config.status: creating doc/Makefile
config.status: creating windows/Makefile
config.status: creating avrdude.spec
config.status: creating Makefile
config.status: creating avrdude.conf.tmp
config.status: creating ac_cfg.h
config.status: executing depfiles commands
sauter:/usr/src/avrdude-5.5#
```

An dieser Stelle kann es zu Fehler kommen, wenn die Bibliothek libusb nicht installiert ist. Auf einem Debian basierenden System kann wieder per APT die fehlenden Komponenten installiert werden. Ebenso werden die Programme bison, flex und g++ für die Übersetzung des Quelltextes benötigt.

```
rechner:/usr/src/avrdude-5.5# apt-get install libusb-0.1-4 libusb-dev bison flex g++
```



Der letzte Befehl muss zwingend als root ausgeführt werden.

```
sauter:/usr/src/avrdude-5.5# make
sauter:/usr/src/avrdude-5.5# make install
```

## avrdude Bedienhinweise

Ein Aufruf um die Signatur eines ATmega32 auszulesen sieht wie folgt aus:

```
avrdude -p m32 -c avrispv2 -P usb
```

## Beliebte Fehlerquellen bei AVR ISP Programmer

- VCC Jumper falsch gesteckt
- Reset Jumper gesteckt obwohl er entfernt sein muss
- Falsche ISP Geschwindigkeit (max 1/4 des CPU Takts)
- Angabe der richtigen ISP Geschwindigkeit mit avrdude mit dem Parameter -B 8 (125 kHz) oder -B 1 (1MHz)
- Angabe der richtigen ISP Geschwindigkeit mit dem AVR Studio 4 über das Register Board (und Write drücken nicht vergessen)

## OpenOCD ARM7/ARM9 Debugger

Mit der OpenOCD-Firmware für den USBprog Adapter lassen sich viele ARM-basierte Mikrocontroller im eingebauten Zustand (in-circuit) programmieren. Der Adapter ermöglicht Echtzeitdebugging, das Setzen von Breakpoints und Ausführen von Einzelschritten, also die ganze Funktionspalette, welche zur erfolgreichen Anwendungsentwicklung und zum effizienten Debugging benötigt werden. Angesteuert wird er über OpenOCD von Dominic Rath.

NEU: Im Testing-Zweig von Debian Sarge ist mittlerweile OpenOCD samt USBprog unterstützung integriert! D.h. es kann einfach über die Paketverwaltung alles benötigte installiert werden.

Dieser Adapter ist nicht der Schnellste! Aber die Geschwindigkeit reicht für die meisten einfachen Programmierungen und Debugsessions völlig aus! Ein Singlestep auf der Telnetkonsole geht fast ohne Verzögerung.

## OpenOCD unter Linux

Debian (Sarge Testing)

1. apt-get install openocd

Aus dem Quelltext installieren

1. Quelltextarchiv herunterladen mit Subversion: svn checkout  
<http://svn.berlios.de/svnroot/repos/openocd/trunk>
2. Kompilierung vorbereiten: ./configure —enable-usbprog
3. Kompilieren: make
4. Installieren in das Dateisystem: make install
5. Rechte anpassen: chmod +s /usr/local/bin/openocd (als Root)

## OpenOCD unter Windows

Für Windows pflegt Michael Fischer eine Installationversion von OpenOCD. Diese ist über die Homepage <http://www.yagarto.de/> erreichbar. Da USBprog noch relativ frisch ist gibt es erstmal hier auf meiner Seite ausschliesslich eine Yagarto OpenOCD USBprog Version. Die Datei muss einfach heruntergeladen und installiert werden.

Entweder kann man anschliessend openocd von der Dos-Box aus starten oder in eine Entwicklungsumgebung die aus Eclipse besteht integrieren.

## Arbeiten mit dem OpenOCD Debugger

### GNU/Linux

Vor etwas längerer Zeit habe ich mal ein kleines Demo hier zusammengeschrieben. Ich denke mal es sollte als Leitfaden ausreichend sein. LPC2103(ARM7) mit OpenOCD unter Linux entwickeln

### Windows

Da ich selber aus der Linux Ecke komme verweise ich direk auf die Seiten von Michael Fischer. Hier wird beschrieben wie man unter Windows mit OpenOCD und dem GCC entwickeln kann:  
<http://www.yagarto.de/howto.html>

## SimplePort

Mit dieser Firmware können ganz einfach die 10 nach außen geführten Leitungen als Ein- und Ausgabeleitungen verwendet werden. Dazu gibt es eine kleine Bibliotheken in C, Python und Java die in Windows, Linux, und wenn notwendig in Mac einsetzbar sind.

Für erste Tests kann man die Leitung 11 als Ausgang verwenden, daran hängt die LED! So könnte ein C Beispiel aussehen:

```
struct simpleport * sp_handle;
/* open connection to simpleport */
sp_handle = simpleport_open();

simpleport_set_pin_dir(sp_handle,11,1);

for(i=0;i<4;i++){
    simpleport_set_pin(sp_handle,11,1); // LED an
    sleep(1);
}
```

```
simpleport_set_pin(sp_handle,11,0); // LED aus
sleep(1);
}
simpleport_close(sp_handle);
```

Im Downloadbereich gibt es Beispiele, in denen gezeigt wird, wie die Bibliotheken in den verschiedenen Sprachen eingesetzt werden können.

Die verschiedenen Ansteuerungen aus den einzelnen Sprachen wie Java und Python wurden mit SWIG realisiert. Jederzeit können ohne grossen Aufwand weitere Anbindungen erzeugt werden. Mehr dazu gibt es hier.

SWIG kann aktuell Anbindungen für die folgenden Sprachen erzeugen: Allegro CL, C#, Chicken, Guile, Java, Modula-3, Mzscheme, OCAML, Perl, PHP, Python, Ruby, Tcl.

## Anschlussbelegung

10-polige Stecker

Pin	Bezeichnung	Aufrufname
1	IO1	1
2	VCC	
3	IO2	2
4	IO3	3
5	IO4	4
6	IO5	5
7	IO6	6
8	IO7	7
9	IO8	8
10	GND	
1-0	IO9 (JP3)	9
1-1	IO10 (JP3)	10
LED	IO11	11

Auf der Platine befindet sich zusätzlich eine rote LED. Diese kann wenn IO11 als Ausgang konfiguriert ist, angesteuert werden.

## Bibliothek in C

Verbindung mit SimplePort aufbauen:

```
struct simpleport* simpleport_open();
```

Verbindung beenden:

```
void simpleport_close(struct simpleport *simpleport);
```

Datenrichtung der Signale definieren (IO 1 - IO 8) 1 = Ausgang, 0 = Eingang:

```
void simpleport_set_direction(struct simpleport *simpleport, unsigned char direction);
```

Bsp: IO 1 - IO 4 = Taster, und IO 5 - IO 8 = LED: (als Hexzahl ist das: 0x0F)

Mit der Funktion können nur die Datenrichtungen für IO 1 - IO 8 angegeben werden! Die für IO 9 - IO 11 müssen mit der Funktion `void simpleport_set_pin_dir(struct simpleport *simpleport, int pin, int dir)` einzeln angegeben werden.

Datenrichtung einer einzelnen Leitung (auch IO 9 - IO 11) 1=Ausgang, 0=Eingang:

```
void simpleport_set_pin_dir(struct simpleport *simpleport, int pin, int dir);
```

Port ausgeben (IO 1 - IO 8):

```
_void simpleport_set_port(struct simpleport *simpleport, unsigned char value);_
```

Port lesen (IO 1 - IO 8):

```
unsigned char simpleport_get_port(struct simpleport *simpleport);
```

Eine einzelne Leitung setzen (IO 1 - IO 11):

```
void simpleport_set_pin(struct simpleport *simpleport, int pin, int value);
```

Eine einzelne Leitung lesen (IO 1 - IO 11):

```
int simpleport_get_pin(struct simpleport *simpleport, int pin);
```

## Beispiel in C

```
#include <stdio.h>
#include "simpleport.h"

int main()
{
    struct simpleport * sp_handle;
    /* open connection to simpleport */
    sp_handle = simpleport_open();

    if(sp_handle==0)
        fprintf(stderr, "unable to open device\n");

    simpleport_set_direction(sp_handle, 0xFF);

    while(1){
        simpleport_set_port(sp_handle, 0xFF);
        simpleport_set_port(sp_handle, 0x00);
    }
    simpleport_close(sp_handle);
    return 0;
}
```

## Beispiel in Java

```
class demo
```

```
{
    public static void main(String[] args){
        try {
            // tell the system to load the shared library into memory
            System.load("/lib/_simpleport.so");
            // the functions of '_simpleport.so' are accessed over the java-class
            // 'simpleport', that was created by SWIG.
            // 'simpleport_open()' returns a instance of 'SWIGTYPE_p_simpleport' if
            // a suitable hardware was found.

            SWIGTYPE_p_simpleport sp_handle = simpleport.simpleport_open();
            // set the port-direction to 'write'
            simpleport.simpleport_set_direction(sp_handle, (short) 0xFF);
            System.out.println("... blink!");

            // periodically set entire port to '00000000' and '11111111'

            while(true){
                simpleport.simpleport_set_port(sp_handle, (short) 0xFF, (short) 0xFF);
                Thread.sleep(1000);
                simpleport.simpleport_set_port(sp_handle, (short) 0x00, (short) 0xFF);
                Thread.sleep(1000);
            }
        } catch (Exception e) {
            e.toString();
        }
    }
}
```

## Beispiel in Python

```
import simpleport
import time

if __name__ == "__main__":
    # call simpleport_open() to retrieve a handle
    sp_handle = simpleport.simpleport_open()

    # periodacally set entire port to '11111111' and '00000000'
    while 1:
        simpleport.simpleport_set_port(sp_handle, 0xFF, 0xFF)
        time.sleep(1)
        simpleport.simpleport_set_port(sp_handle, 0x00, 0xFF)
        time.sleep(1)

    # close handle (never reached in this case)
    simpleport.simpleport_close(sphand)
```

## SimplePort RS232

Mit SimplePortRS232 kann können einfach und bequem die IO-Pins von USBprog über ein Terminal oder Bibliotheken für die serielle Schnittstelle angesteuert werden.

Das Gerät meldet sich in Windows als virtueller Comport und in GNU/Linux als /dev/ttyACM0 an. Jetzt kann mit jeder Programmiersprache die ein Interface für die serielle Schnittstelle anbietet gearbeitet werden.

Die Durchnummerierung der einzelnen Pins sieht wie folgt aus:

Pin	Bezeichnung	Aufrufname
1	IO1	1

Pin	Bezeichnung	Aufrufname
2	VCC	
3	IO2	2
4	IO3	3
5	IO4	4
6	IO5	5
7	IO6	6
8	IO7	7
9	IO8	8
10	GND	
LED	IO11	B

## Kommandos für die Ansteuerung der Leitungen

Die Kommandos werden als ASCII-Zeichen übertragen. Das hat den Vorteil, das die Funktionalität bereits mit einem einfachen Terminal überprüft werden kann.

### Datenrichtung einer einzelnen Leitung definieren

Kommando: d<Aufrufname><Richtung>\*

- Aufrufname - (siehe Tabelle Pin/Bezeichnung/Aufrufname)
- Richtung - 1=Ausgang, 0=Eingang (mit internen Pullups)

Rückgabewert: keiner

Beispiel: dB1\* (Status LED als Ausgang), d10\* (IO1 als Eingang)

### Signale einer Ausgangsleitung setzen

Kommando: p<Aufrufname><Wert>\*

- Aufrufname - (siehe Tabelle Pin/Bezeichnung/Aufrufname)
- Wert - 1 = 5V (high) , 0 = GND (low)

Rückgabewert: keiner

Beispiel Aufruf: pB1\* (Status LED an), pB0\* (Status LED aus)

### Signal an einer Eingangsleitung lesen

Kommando: i<Aufrufname>\*

- Aufrufname - (siehe Tabelle Pin/Bezeichnung/Aufrufname)

Rückgabewert: 2 Bytes abholen

Als Rückgabewert müssen für die Funktion immer 2 Werte sofort nach dem Ausführen des Kommandos abgeholt werden. Die Antwort ist wie folgt zu lesen: i0 = 0V (low), i1 = 5V (high).

Beispiel Aufruf: i1\* (Abfrage Signal IO1) Beispiel Antwort: i0 (Signale hatte den Wert low), i1 (Signal hatte den Wert high)

### Ersten 8 IO Leitungen auf einmal abfragen

Kommando: g\*

Sollen zu einem Zeitpunkt mehrere Leitungen abgefragt werden, um beispielsweise bei mehreren angeschlossenen Tastern eine Tastenkombination zu einem Zeitpunkt zu ermitteln, kann dies mit der aktuellen Funktion geschehen. Es werden beim Aufruf des Kommandos die Werte zum gleichen Zeitpunkt gemessen.

Rückgabewert: 8 Bytes abholen

Das Ergebnis ist eine Reihe von acht 0 und 1 Werten. Die ganz linke Zahl entspricht IO1 und ganz rechts IO8. Ist entsprechend eine 1 gesetzt war ein High am Signal angelegt, bei einer 0 entsprechend ein Low.

Beispiel Aufruf: g\* (IO1 - IO8 zu einem Zeitpunkt abfragen) Beispiel Antwort: 10001000 (IO1 und IO5 waren high, der Rest low)

## Beispiel in Python

Das Paket serial für Python muss zuvor installiert werden. In Debian reich ein einfaches *apt-get install python-serial*.

```
import serial
import time

ser = serial.Serial('/dev/ttyACM0', 19200, timeout=1)

ser.write("")
ser.write("*dB1*")

while(1):
    ser.write("pB0*")
    time.sleep(1)
    ser.write("pB1*")
    time.sleep(1)
```

## Einsatz in C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO.Ports;
using System.Threading;

namespace Test1
{
```

```
class Program
{
    static void Main(string[] args)
    {
        // open comport: name (COM16) depends on your system
        SerialPort USBProg = new SerialPort("COM16", 9600,
                                            Parity.None, 8, StopBits.One);
        USBProg.Open();

        //set direction of Pin 11 (B)
        USBProg.Write("dB1*");

        char[] buffer = new char[255];
        for (int i = 0; i < 5; i++)
        {
            //disable LED
            USBProg.Write("pB0*");
            Console.WriteLine("Answer: ");
            Console.WriteLine(USBProg.ReadExisting());
            Thread.Sleep(500);
            //enable LED
            USBProg.Write("pB1*");
            //sensorStream.Read(buffer,0,2);
            Console.WriteLine("Answer: ");
            Console.WriteLine(USBProg.ReadExisting());
            Thread.Sleep(500);
        }
        //close comport
        USBProg.Close();

        //keep console open
        Console.Read();
    }
}
```

TDB

## USB zu RS232 Wandler

Mit dieser Firmware Version kann man usbprog als einfachen RS232 Wandler in allen bekannten Betriebssystemen nutzen. Er arbeitet mit den Standardtreibern vom Betriebssystem.

Um ein RS232 Gerät ansteuern zu können muss man sich nur einen Adapter von JP2 auf einen entsprechenden Stecker (evtl. 9 polig SUB-D) basteln.

## Status

- Mit einer festeingestellten Baudrate (fix in der Firmware) von 9600 8N1 getestet und einsatzfähig unter Linux und Windows XP
- Das man zwischen verschiedenen Baudraten hin und herschalten kann ist nicht mehr viel Aufwand. Wenn das jemand dringend braucht, schreibt einfach eine Mail, dann kann ich das dann schnell machen.

## Linux



cdc-acm als Modul oder fest im Kernel (/dev/ttyACMx)

## MacOS

Das Gerät sollte als /dev/cu.usbmodem\*\*\* erscheinen. Mit Mac habe ich es noch nicht getestet, aber es sollte eigentlich funktionieren. Wenn nicht gebt mir kurz bescheid und ich schau mir das dann an.

## AT89 Programmer

Mit der at89prog Firmware kann mit dem usbprog Adapter der AT89S8252 programmiert, gelöscht und resetet werden. Falls Bedarf an anderen Controllern der AT89 Familie besteht, meldet dies mir einmal. Zu der Firmware gibt es ein kleines Konsolenprogramm, über das man den Adapter ansteuern kann. Wie dies genau zu verwenden ist, ist in dem Abschnitt "Hilfe für at89prog" beschrieben.

## Status

Die Firmware ist mit dem AT89S8252 auf Windows und Linux erfolgreich getestet worden. Da es noch wenig Feedback von Benutzern gibt, würde ich sagen sie befindet sich noch im Beta-Status. Aktuell kann man mit der Firmware eine .Bin Dateien in den Flashspeicher übertragen, den Flash löschen und den AT89 reseten. Bei Bedarf an weiteren Funktionen einfach melden (sauter@ixbat.de). Die Struktur steht, ja d.h. alles andere sollte schnell programmiert sein.

### GNU/Linux

- Programm löschen `./at89prog -e`
- Programm hochladen `./at89prog -u /home/bene/test1.BIN`
- CPU Reset `./at89prog -r`

**Windows Programm löschen** `at89prog.exe -e` Programm hochladen `at89prog.exe -u c:\test1.BIN *`  
CPU Reset `at89prog.exe -r`

## JTAG Adapter

Der universale JTAG Adapter hier dient als Basis für die JTAG Kommunikation. Die Bibliothek für den USB JTAG Adapter kann einfach in andere Projekte integrieren.

```
/* low level functions */
void usbprog_jtag_read_tdo(struct usbprog_jtag *usbprog_jtag, char * buffer, int size);
void usbprog_jtag_write_tdi(struct usbprog_jtag *usbprog_jtag, char * buffer, int size);
void usbprog_jtag_write_and_read(struct usbprog_jtag *usbprog_jtag, char * buffer, int size);
void usbprog_jtag_write_tms(struct usbprog_jtag *usbprog_jtag, char tms_scan);

/* single io function! An emulated JTAG connection is very slow!!! */
void usbprog_jtag_set_direction(struct usbprog_jtag *usbprog_jtag, unsigned char direction);
void usbprog_jtag_write_slice(struct usbprog_jtag *usbprog_jtag, unsigned char value);
unsigned char usbprog_jtag_get_port(struct usbprog_jtag *usbprog_jtag);
void usbprog_jtag_set_bit(struct usbprog_jtag *usbprog_jtag, int bit, int value);
int usbprog_jtag_get_bit(struct usbprog_jtag *usbprog_jtag, int bit);
```

```
/* basic jtag tap functions */
void usbprog_jtag_tap_goto_reset(struct usbprog_jtag *usbprog_jtag);
void usbprog_jtag_tap_goto_capture_dr(struct usbprog_jtag *usbprog_jtag);
void usbprog_jtag_tap_goto_capture_ir(struct usbprog_jtag *usbprog_jtag);

void usbprog_jtag_tap_shift_register(struct usbprog_jtag *usbprog_jtag, char * in, char *
```

## XSVF Player (Xilinx CPLDs und FPGAs programmieren)

XSVF-Dateien stellen ein standardisiertes Format dar, um prinzipiell beliebige JTAG-Operationen zu beschreiben. Mit dieser Firmware für usbprog ist es möglich solche XSVF-Dateien "abzuspielen", das heißt die enthaltenen JTAG-Operationen über den usbprog-Adapter auszuführen. Damit kann man beispielsweise CPLDs, FPGAs oder Mikrocontroller mit JTAG-Schnittstelle programmieren, löschen, testen usw. Voraussetzung dafür ist, dass man eine Software hat, die entsprechende XSVF-Dateien für das Target-Device erstellen kann. Status

Der XSVF Player funktioniert für den Fall, dass keine einzelne XSVF-Instruktion länger als 64 Bytes ist. Getestet wurde unter Linux (openSUSE 10.3 x86\_64, Debian/Sarge) mit einem Xilinx XC9572 CPLD sowie mit einem XC9572XL CPLD.

## Anschlussbelegung

1. TDI
2. VCC
3. leer
4. leer
5. TMS
6. leer
7. TCK
8. leer
9. TDO
10. GND

## XSVF Player unter Linux

Quelltextarchiv herunterladen mit Subversion:

- svn checkout <http://svn.berlios.de/svnroot/repos/usbprog/trunk/usbprogXSVF>

Kommandozeilen-Tool kompilieren:

- cd lib
- make

Benutzung:

- ./xsvfplayer <XSVF-Dateiname>

## XSVF-Dateien erstellen mit Xilinx ISE 9.2i WebPack

Um XSVF-Dateien zu erstellen, mit denen ein Xilinx CPLD oder FPGA konfiguriert werden kann, kann man folgendermaßen vorgehen:

In der ISE-Projektansicht die Top-Entity auswählen und dann "Implement Design" -> "Optional Implementation Tools" -> "Generate SVF/XSVF/STAPL File" ausführen. Es öffnet sich ein neues Fenster, dort "Prepare a Boundary-Scan File" aktivieren und als Format "XSVF" auswählen. Auf "Finish" klicken. Dann der zu erzeugenden XSVF-Datei einen Namen geben und im nächsten Fenster "Ok" klicken. In dem sich danach öffnenden Fenster ("Add Device") die Datei mit gleichem Namen wie die Top-Entity und Endung .jed im Projektordner auswählen. Anschließend im Hauptfenster Rechtsklick auf das CPLD- oder FPGA-Symbol in der JTAG-Chain und auf "Program" klicken, mit "Ok" bestätigen. Zum Schluss auf "Output" -> "XSVF File" -> "Stop Writing to XSVF File" und fertig ist das XSVF.

Alternativ zum "Program"-Schritt kann man natürlich auch beliebige andere JTAG-Operationen ausführen und in der XSVF-Datei aufzeichnen.

Interessante und hilfreiche Linkadressen

<http://www.ethernut.de/en/xsvfexec/index.html> (Fertige Routinen)

<http://www.xilinx.com/bvdocs/appnotes/xapp058.pdf> (Beschreibung des XSVF Formats)

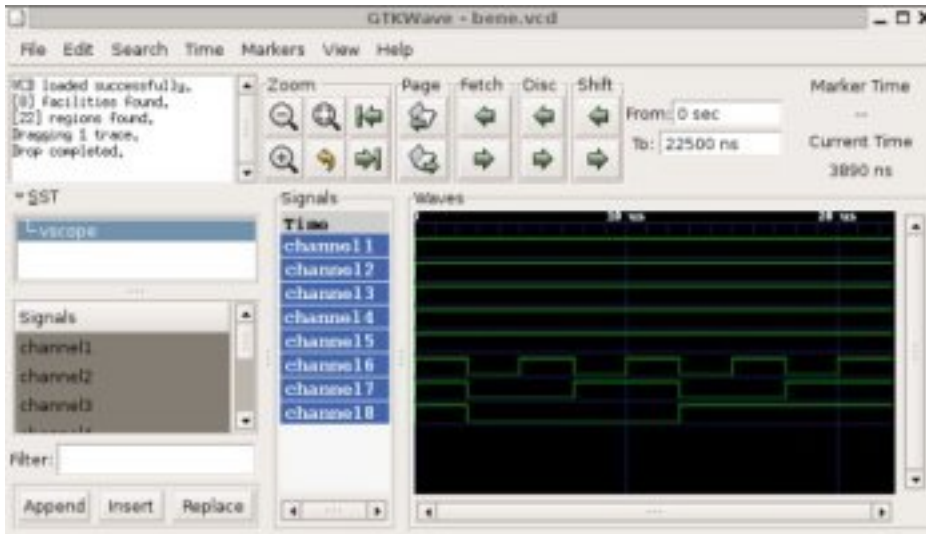
<http://www.xilinx.com/bvdocs/publications/ds300.pdf>

## Logik Analysator (250 kHz, 8 Signale, Trigger)

- 8 Kanäle
- Online Modus ( Daten werden direkt während der Messung abtransportiert)
- Speicher Modus (es werden intern bis zu 1000 Messungen aufgezeichnet)
- Snapshot Modus (für langsame gezielte Aufzeichnungen z.B. Counter, Logiktests ... )
- einstellbare Abtastrate von 5µs bis 100 ms (max 250kHz)

- einstellbare Trigger (Flanke an einer Leitung, Muster auf allen Leitungen)
- einfache Konsolenanwendung zum Aufzeichnen für Linux und Windows
- als Ausgabeformat werden vcd-Dateien erzeugt. Diese kann man mit vielen Tools bearbeiten. (Bsp. GTKWave)

**Figure 7.1. GTKWave**



Das Gerät wurde nicht als Profi-Logikanalyser, sondern für einfache und relativ langsame Messungen (bis 250kHz) geplant. Interessant ist dieses Gerät für Bastler, die gerne mit kleinen Mikrocontrollern arbeiten und ab und an gerne in eine UART, SPI oder I2C Verbindung schauen möchte, oder einfach nur für Versuche oder den Schulunterricht.

Das Projekt besteht aus drei Teilen. Der Hardware, die Bestandteil dieses Projektes ist (die Pläne dazu stehen im Downloadbereich zur Verfügung). Dann gibt es das Programm logic2vcd, um Messungen auf dem Gerät zu starten und zu steuern (gehört ebenfalls zu diesem Projekt). Dieses Programm erzeugt sogenannte .vcd-Dateien, die mit dem dritten Programm GTKWave analysiert werden können. GTKWave ist nicht Bestandteil dieses Projektes aber ebenfalls ein Open Source Projekt. Beide Programme gibt es für Linux und Windows. Status

Der Logikadapter wurde ausgiebig mit allem getestet was ich hier so gefunden hab. Bis jetzt ist mir noch kein Fehler bekannt. Da ihn aber noch einfach zu wenige getestet haben würde ich ihn als Beta einstufen

## Anschlussbelegung

1. Kanal 6
2. VCC
3. Kanal 5
4. Kanal 4

- 5. Kanal 1
- 6. Kanal 3
- 7. Kanal 8
- 8. Kanal 2
- 9. Kanal 7
- 10. GND

## Downloads

GTKWave:

Homepage: <http://home.nc.rr.com/gtkwave> Download: <http://www.dspia.com/gtkwave.html>

## Aufzeichnung von Messungen mit logic2vcd

Das Programm logic2vcd dient der Steuerung der Messung mit Hilfe der Hardware. Man startet das Programm mit den entsprechenden Kommandozeilenargumenten, und bekommt als Resultat eine .vcd Datei. Dieses Format kommt aus der Hardware Entwicklung. Es dient normalerweise dazu um Logikschaltungen nach einer Simulation analysieren zu können. Der Vorteil dieses Datei Formates ist, dass es bereits einige Programme zum be- und verarbeiten gibt (unter anderem GTKWave).

Einfache Online Messung:

```
./logic2vcd -f messung.vcd -R online -s 1ms -n 1000
```

Im Detail bedeutet dies:

```
-f Namer der Datei in die die Werte geschrieben werden sollen  
-R (Recordtype) Aufnahmemodus = online  
-s Abtastrate (jede Millisekunde wird ein neuer Wert gelesen )  
-n Anzahl der Messungen
```

Bei der Online Messung werden so schnell wie möglich die Daten von der Hardware abgeholt, so dass es im Analysator zu keinem Stau kommt. Kommt es jedoch zwischenzeitlich zu kurzen Unterbrechungen gehen Messdaten verloren. Dieses passiert bei hohen Abtastraten häufiger. Wenn es um hundertprozentige Genauigkeit geht, muss man auf den sogenannten internen Modus wechseln (siehe unten).

Online Messung mit Start-Trigger:

Bei der einfachen Messung beginnt die Aufzeichnung mit dem Starten des Kommandos. Da man so jedoch schwer den Bereich erwischt, den man wirklich aufzeichnen möchte, kann man einen Start-Trigger definieren. Erst wenn das Signal - wie im Trigger definiert erkannt wird, beginnt die Aufzeichnung mit den entsprechenden Parametern.

```
./logic2vcd -f messung.vcd -R online -s 1ms -n 1000 -T edge -c 1 -t 1
```

Im Detail bedeutet dies:

```
-T Art des Triggers, entweder kann man die Flanke eines Kanals beobachten (edge)
    oder man kann den ganzen Port mit einem Muster vergleichen (pattern)
-c Kanalnummer (1-8) funktioniert nur beim Edge-Trigger
-t Der zu vergleichende Wert
    * bei Edge=1 für einen Übergang von low - high und eine 0 für high - low
    * bei Pattern das Hexmuster für den Port, wenn port 1,2 und 8 high sein sollen,
      dann muss als Wert 193 (Hex: C1, Binär: 1100 0001) angegeben werden
-i Wenn man bestimmte Kanäle beim Pattern Trigger ignorieren will, kann man
  diese hier angeben, genau gleich wie bei -t. Wenn man Kanal 1-4
  ignorieren möchte, muss man entsprechend 240 (Hex: 0xf0 und Binär 11110000) angeben
```

Die restlichen Parameter steuern wie auch bei der einfachen Messung die Aufzeichnung, die ab der erkannten Triggerbedingung startet.

Genauere interne Messung mit Start-Trigger:

Im internen Modus werden maximal 1000 Messwerte in der Hardware aufgezeichnet. Danach stoppt die Messung und man kann die Messwerte abholen. 1000 Messwerte ist nicht gerade viel, aber dank der Trigger kann man sich gut an die entsprechenden Stelle in der Messung hinarbeiten.

## Datenanalyse mit GTKWave

Mit GTKWave kann man einfach Messungen analysieren. Gestartet wird das Program direkt mit dem Dateinamen der Messung als Parameter:

```
gtkwave messung.vcd
```

Als erstes muss auf vscope geklickt werden, um die Kanäle im Feld Signals einzublenden. Anschliessend kann man alle Kanäle makieren, und muss sie dann nur noch einfügen. Wenn man oben auf die Lupe klickt wird die komplette Messung in dem Fenster angezeigt. Jetzt kann man sich mit den restlichen Knöpfen an die entsprechende Stelle hinarbeiten.

---

# Chapter 8. Eigene Firmware entwickeln

Hier soll gezeigt werden wie eigene Firmwares entwickelt werden können.

## Schritt für Schritt

1. Herunterladen skeleton
2. entpacken

## USBprog für Entwickler

Alle Quelltexte befinden sich im Versionsverwaltungssystem Subversion bei Berlios. Berlios bietet für Open-Source Anwendungen eine Entwicklerplattform an.

Anonymous SVN Access via SVN

This project's BerliOS Developer SVN repository can be checked out through anonymous (svnserve) SVN with the following instruction set.

```
svn checkout svn://svn.berlios.de/usbprog/trunk
```

Anonymous SVN Access via HTTP

This project's BerliOS Developer SVN repository can be checked out through anonymous HTTP with the following instruction set.

```
svn checkout http://svn.berlios.de/svnroot/repos/usbprog/trunk
```

Developer SVN Access via SSH

Only project developers can access the SVN tree via this method. SSH2 must be installed on your client machine. Substitute developername with the proper value. Enter your site password when prompted.

```
svn checkout \  
svn+ssh://developername@svn.berlios.de/svnroot/repos/usbprog/trunk
```

Developer SVN Access via HTTPS

Only project developers can access the SVN tree via this method. Substitute developername with the proper value. Enter your site password when prompted.

```
svn checkout \  
https://developername@svn.berlios.de/svnroot/repos/usbprog/trunk
```

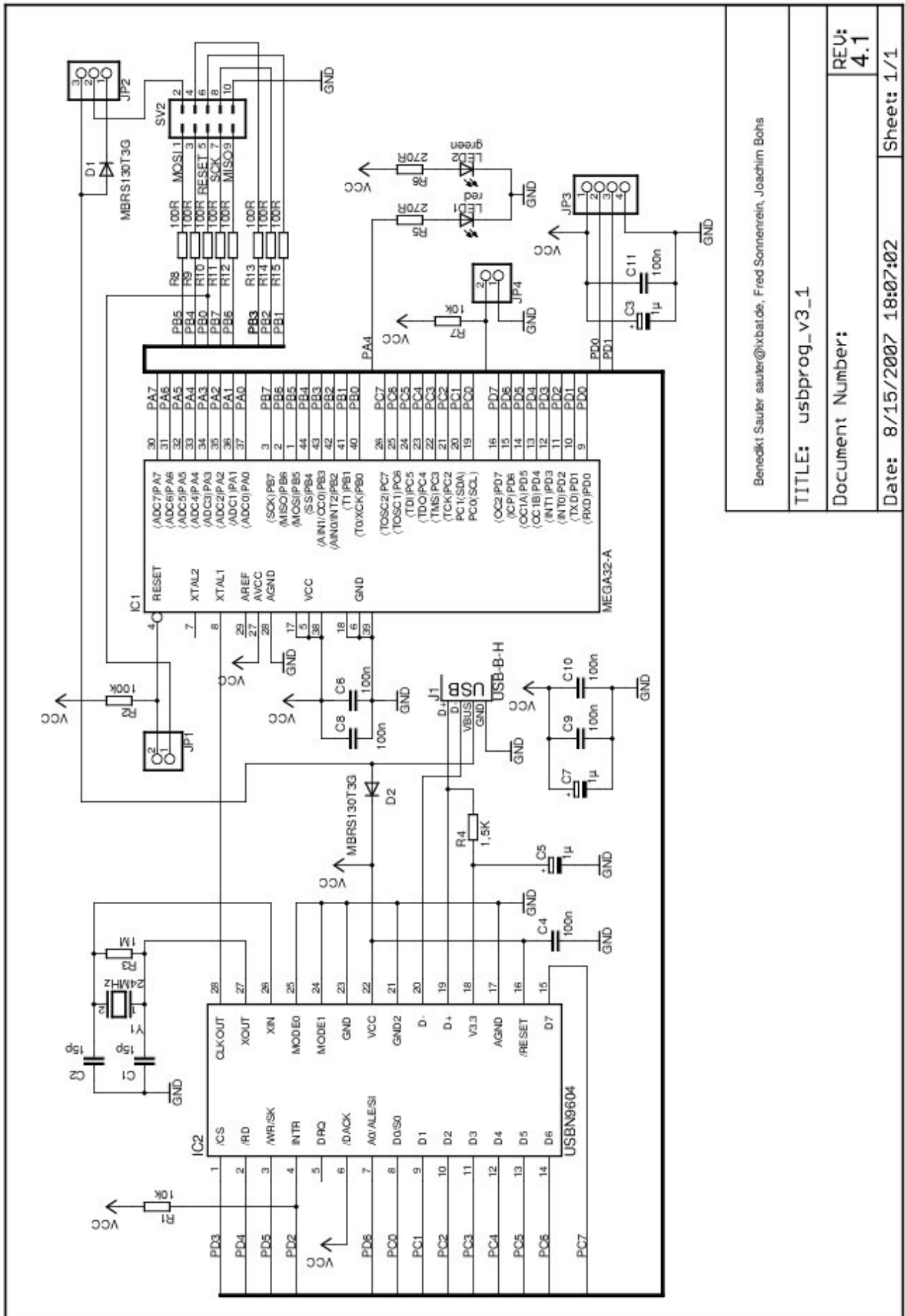
---

# Appendix A. Schaltplan

Zu beachten ist, dass es zwei Platinen gibt. Die sogenannte Version 2.0 und 3.0. Alle Firmwares sind auf beiden einsetzbar. Der grösste Unterschied ist nur der, dass auf der Version 3.0 an der 10-poligen Buchse der komplette Port B anliegt. Die Reihenfolge der Pins ist leider etwas durcheinander, aber das ist dafür das er eben kompatibel zur Version 2.0 ist. Damit müssen wir wohl lange leben müssen. Mir ist klar, dass das nicht sehr schön ist, aber es ist fair den Leuten gegenüber die Version 2.0 haben.

## Figure A.1. Schaltplan USBprog 3.0





Benedikt Sauter sauter@ixbat.de, Fred Sonnenrein, Joachim Böhs

TITLE: usbprog\_v3\_1

Document Number:

REV:  
4.1

Date: 8/15/2007 18:07:02

Sheet: 1/1

---

# Chapter 9. Apeendix B: Lizenzen

Im folgenden sind die Lizenzen für alle Quelltexte rund um USBprog und die Dokumentationen mit abgedruckt.

## GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>>  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

### Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to

avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

#### TERMS AND CONDITIONS

##### 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

##### 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all

the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

### 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

### 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

### 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,

and you may offer support or warranty protection for a fee.

#### 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

#### 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the

Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

### 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall

be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

### 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright

holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

### 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

### 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and



propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

## 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

## 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the

combination as such.

### 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

### 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

### 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

### 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least

the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>
```

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program> Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

### GNU Free Documentation License Version 1.2, November 2002

```
Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

#### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free

software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that

edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

#### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

### 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

### 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

### 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

### 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

### 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

### 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
```



Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the  
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

---

# Index