

USBprog Handbuch

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

USBprog Handbuch

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

Table of Contents

Preface	vii
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	8
3. USBprog Flashtool	10
Installation unter GNU/Linux	12
Installation aus den Quellen heraus	12
Installation unter OpenSUSE	13
Installation unter Debian/Ubuntu	14
Zugriff auf USBprog als Benutzer	14
Installation unter Windows	14
Installation unter MAC/OS	15
4. Firmware wechseln	16
5. Beliebte Fehlerquellen	17
6. Anwendungen mit USBprog	18
AVR ISP Programmer (STK500 kompatibel)	18
AVR Studio 4	18
avrdude unter Windows	18
avrdude unter GNU/Linux	19
avrdude Bedienhinweise	20
Beliebte Fehlerquellen bei AVR ISP Programmer	20
OpenOCD ARM7/ARM9 Debugger	20
OpenOCD unter Linux	20
OpenOCD unter Windows	21
Arbeiten mit dem OpenOCD Debugger	21
SimplePort	21
Status	22
Downloads	22
Anschlussbelegung	22
Bibliothek in C	23
Beispiel in C	23
Beispiel in Java	24
Beispiel in Python	24
SimplePort RS232	25
Kommandos für die Ansteuerung der Leitungen	25
Beispiel in Python	26
Einsatz in C#	27
USB zu RS232 Wandler	27
Status	28
Linux	28
MacOS	28
AT89 Programmer	28
Status	28
JTAG Adapter	29
XSVF Player (Xilinx CPLDs und FPGAs programmieren)	29
Anschlussbelegung	29

XSVF Player unter Linux	29
XSVF-Dateien erstellen mit Xilinx ISE 9.2i WebPack	29
Logik Analysator (250 kHz, 8 Signale, Trigger)	30
Anschlussbelegung	31
Downloads	31
Aufzeichnung von Messungen mit logic2vcd	31
Datenanalyse mit GTKWave	32
7. Eigene Firmware entwickeln	33
A. Datenblatt	34
B. Schaltplan	35
8. Apeendix C: Lizenzen	36
Bibliography	37
Index	38

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. AVR Studio Fuse Einstellung (Bootloader)	6
2.8. PonyProg Fuse Einstellung (Bootloader)	7
2.9. Takteingang ATmega32	9
3.1. USBprog GUI	10
3.2. USBprog Konsole	11

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, 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.

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.

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

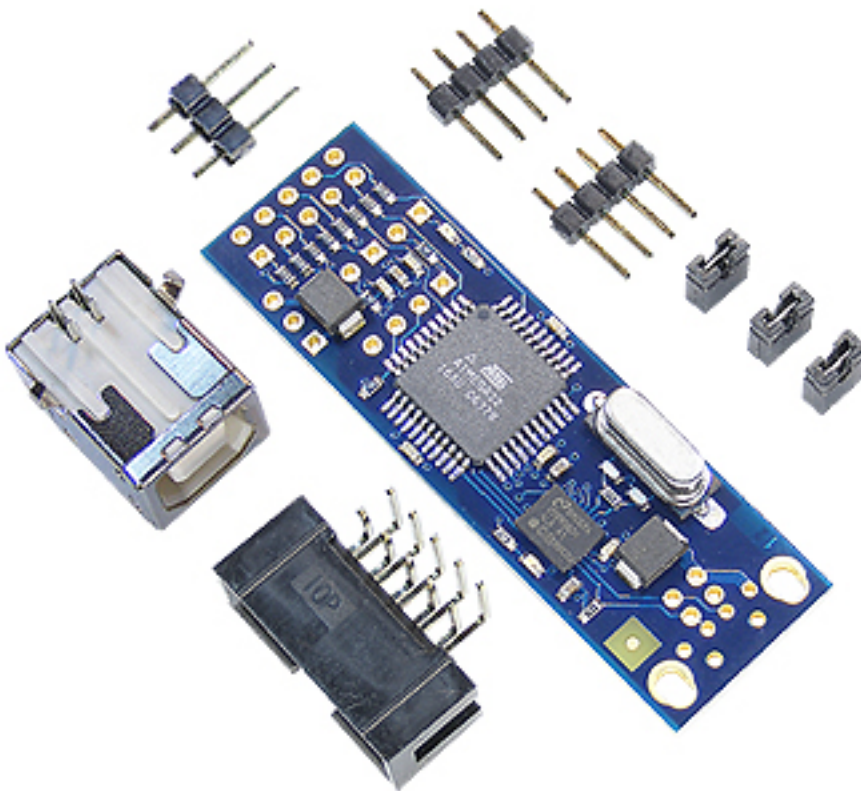
werden kann. Welches den Vorteil hat, dass USBprog in einem passendem Gehäuse als USB-Stick verwendet werden kann.

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

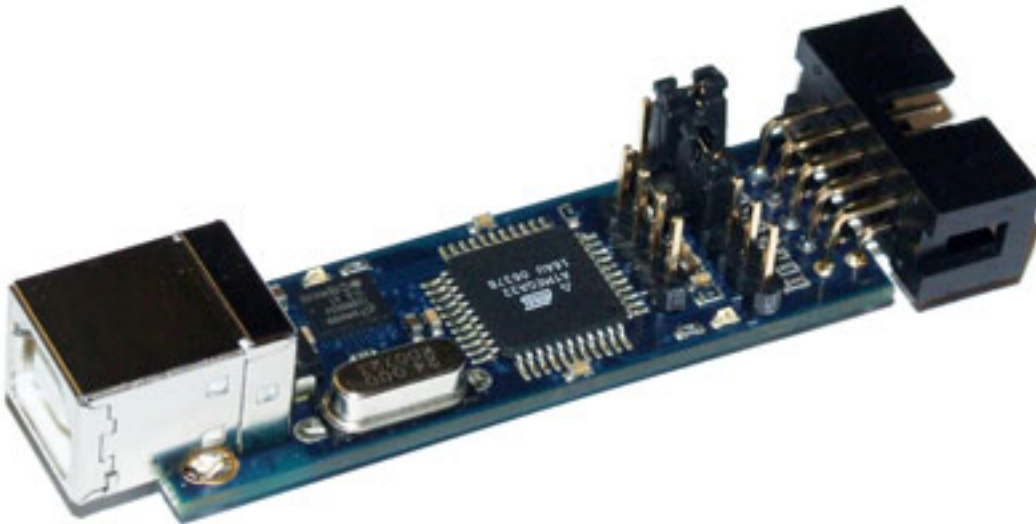
In der Abbildung USBprog Bausatz sind nochmal alle Bauteile gezeigt.

Figure 2.1. USBprog Bausatz



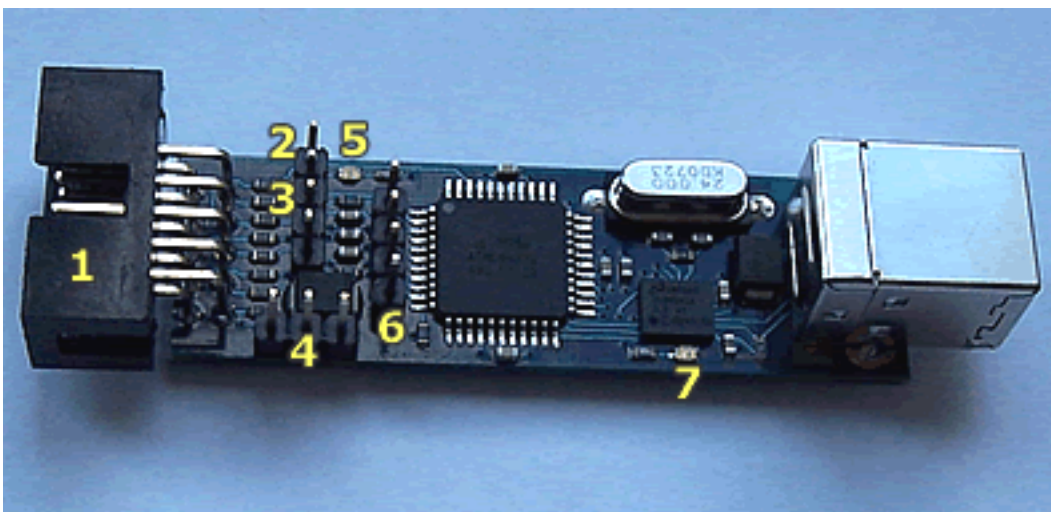
Nach den Lötarbeiten sollte USBprog nun wie folgt aussehen:

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)

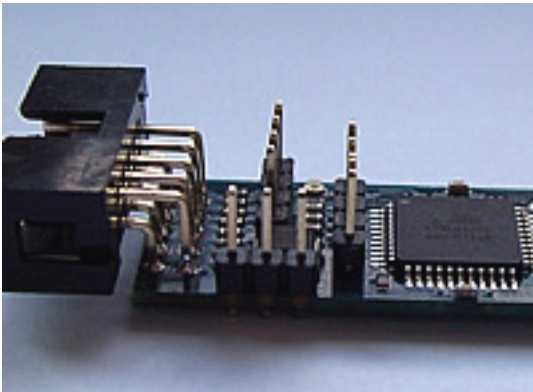


Figure 2.5. VCC Konfiguration 2 (5V direkt über USB)

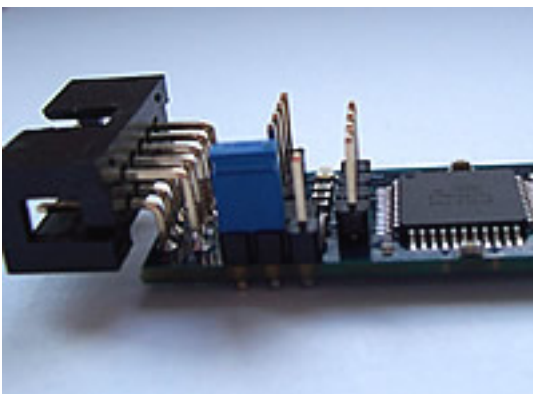
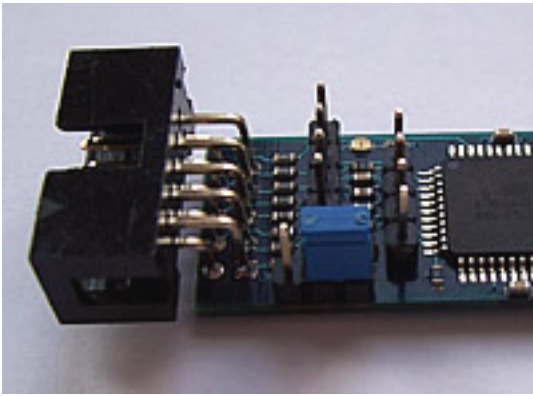


Figure 2.6. VCC Konfiguration 3 (5V per Schottky-Diode über USB)



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.

avrdude unter Linux und Windows

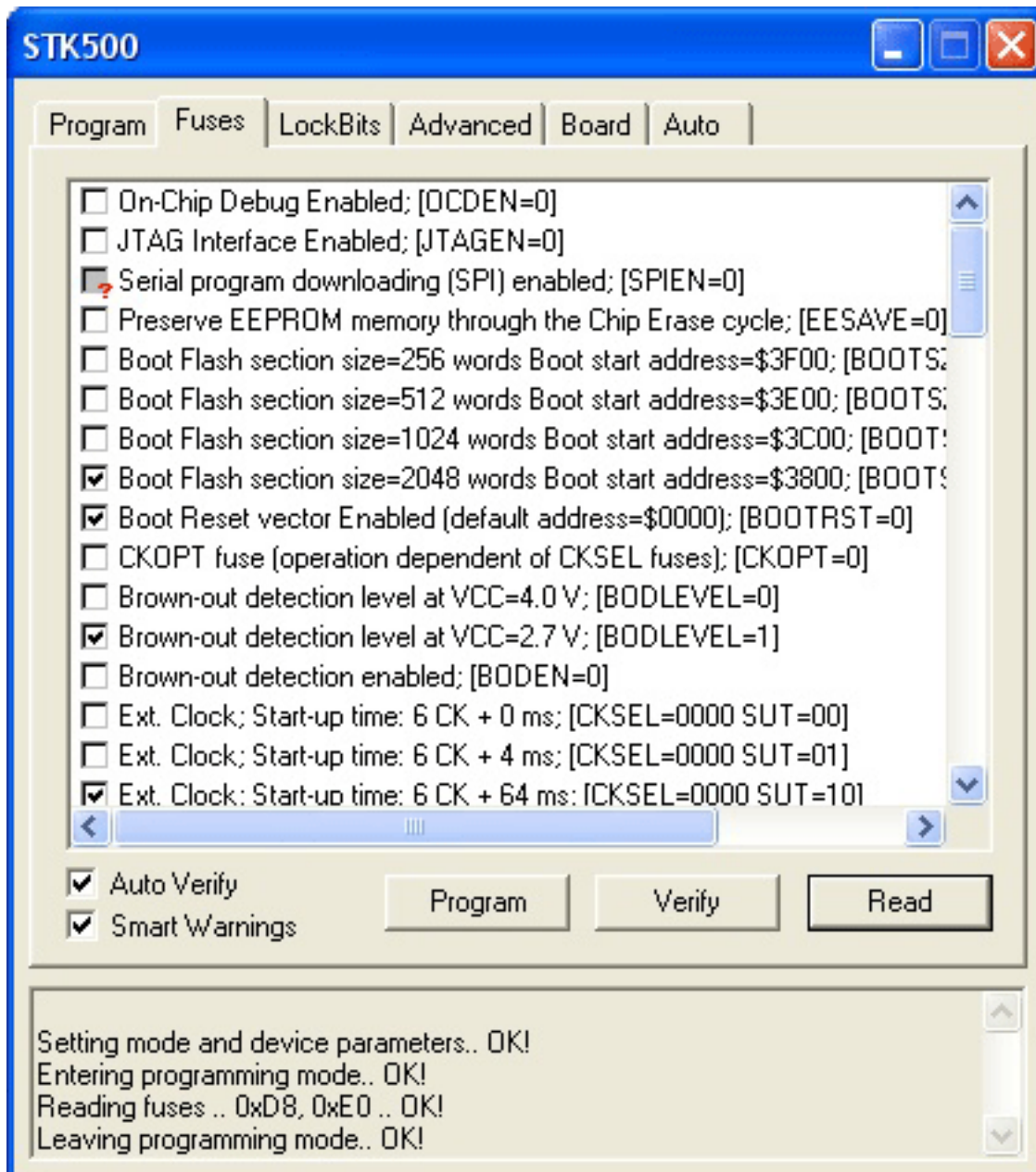
Falls die Firmware mit einem AVR ISP mkII (oder USBprog AVR Programmer) programmiert werden soll, muss als Parameter `-c avrispv2 -P usb` angegeben werden. Für ein einfaches Parallelportkabel reicht `-c bsd`.

- Firmware flashen: `avrdude -p m32 -c avrispv2 -P usb -B 8 -U flash:w:usbprog_bootloader.hex`
- lfuse 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`

AVR Studio

Falls ein Programmiergerät zum programmieren 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 2.7. 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 2.8. 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

Cancel OK Clear All Set All Write Read

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.

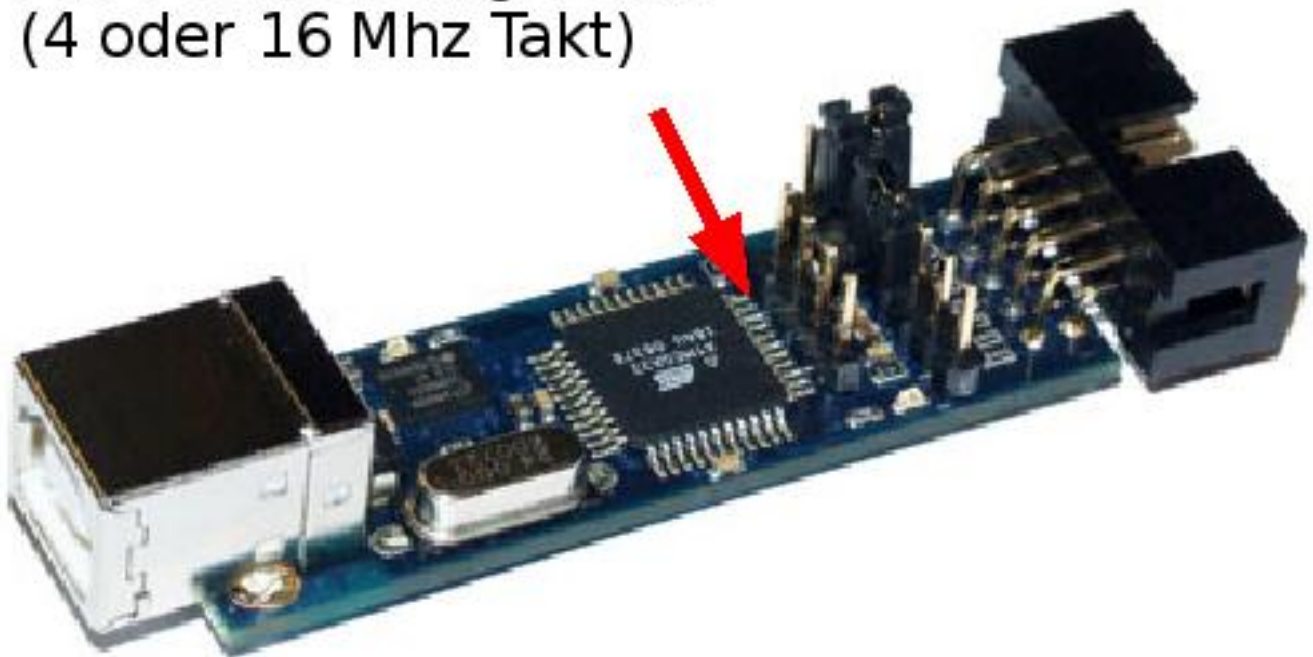
Blinkt die LED trotz einspielen des Bootloaders und einstellen der Fuse Bits nicht, kann mit einem Oszilloskop fest gestellt werden ob die Schaltung denn irgendetwas macht, oder ob ein grösseres Problem vorliegt.

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).

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 Taktgeschwindigkeit 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.

Figure 2.9. Takteingang ATmega32

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



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.

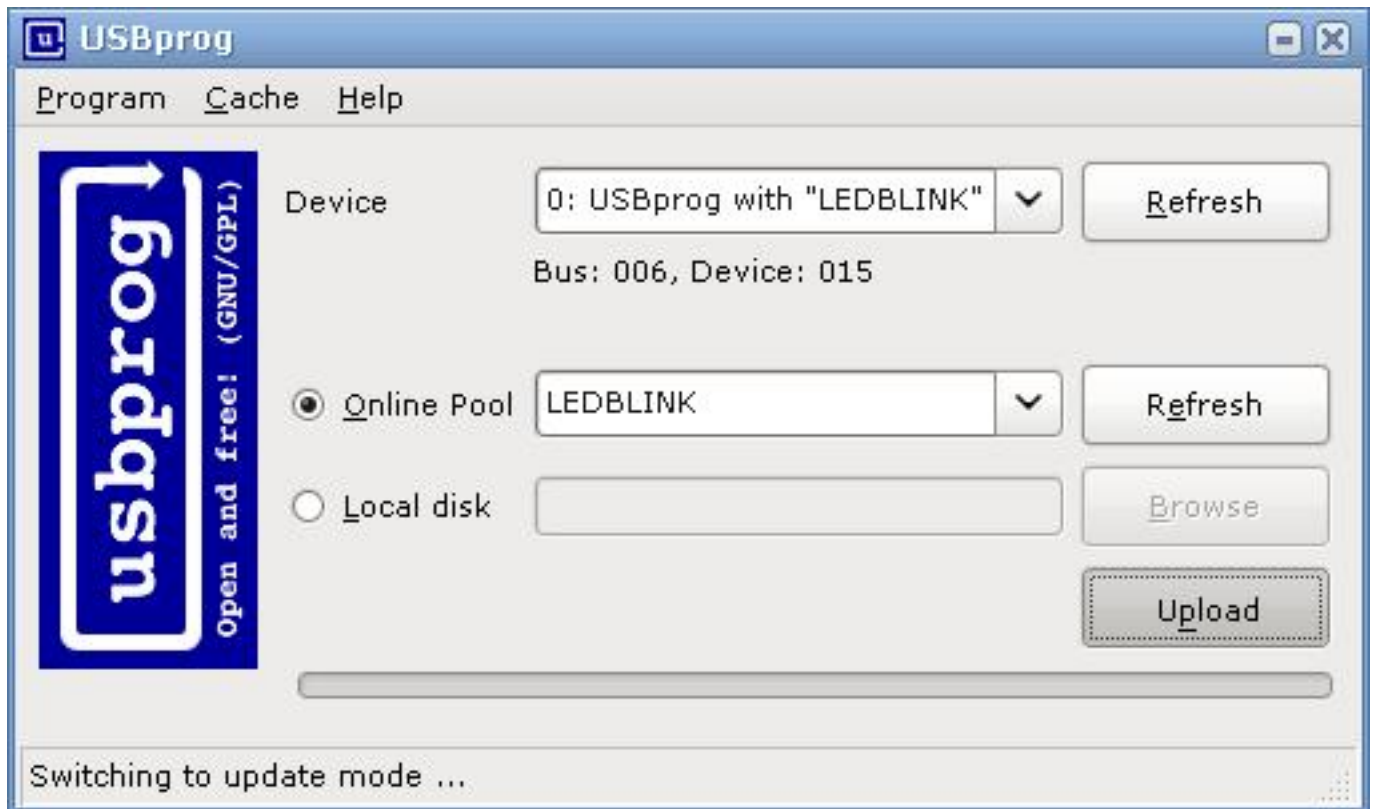
Von USBprog gibt es sowohl eine GUI-Version als auch eine Kommandozeilen-Version. Die Kommandozeilen-Version verfügt über einen interaktiven Modus (wie eine Shell) und einen Batch-Modus, kann daher problemlos in Skripte, Makefiles o.ä. integriert werden.

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.

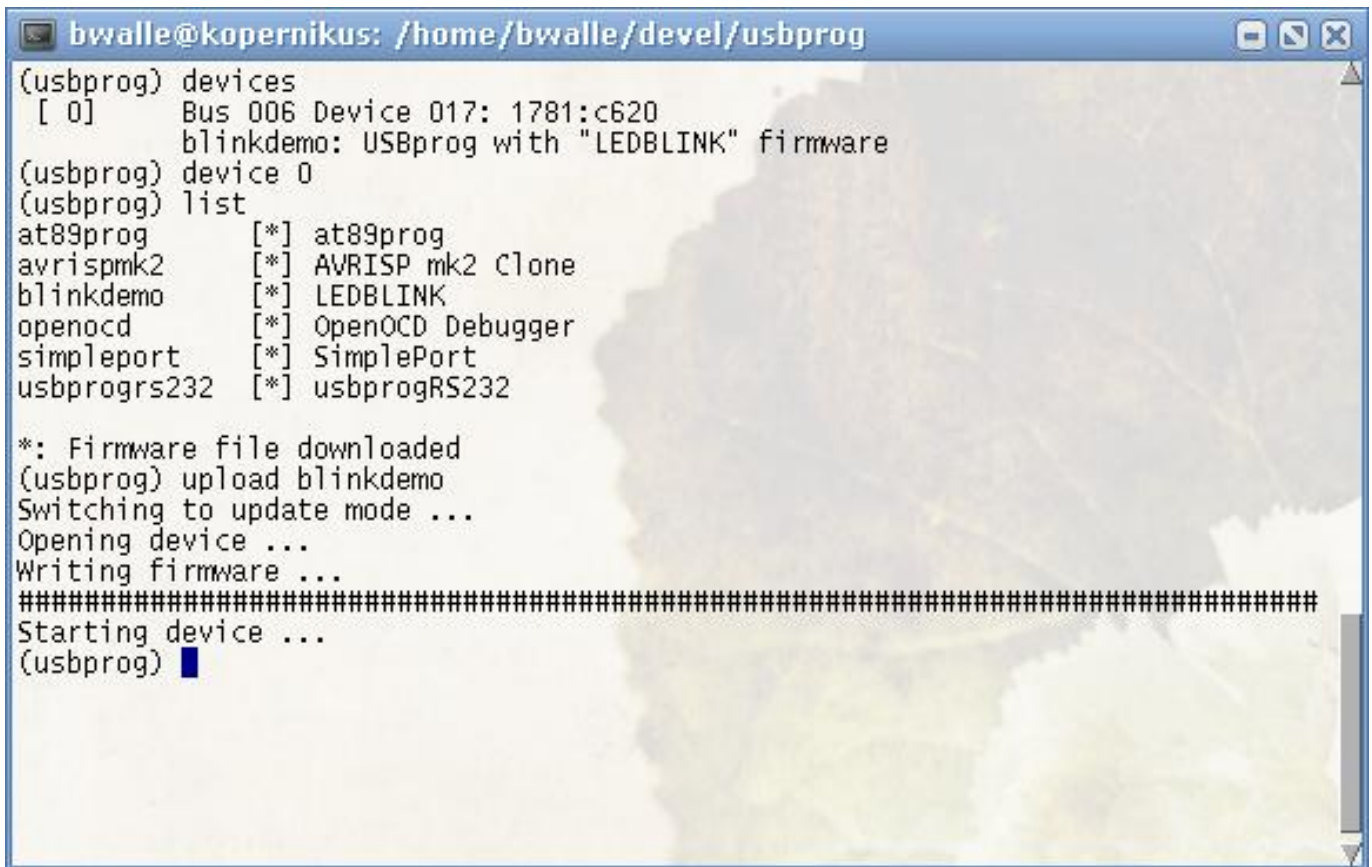
Da ein Bild bekanntlich mehr als tausend Worte sagt, hier ein Screenshot sowohl der GUI- als auch der Kommandozeilen-Version, um einen Eindruck der Applikation zu bekommen. GUI-Version (unter Linux)

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>  
</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. 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 5. 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

Chapter 6. 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

TDB

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

BLA

- 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.

Status

- getestet unter Linux mit C, Java und Python
- Windowstest fehlt noch, der Betrieb sollte aber klappen

Downloads

Beschreibung Download Hex-Firmware simpleport.hex Bin-Firmware simpleport.bin Bibliotheken SimplePort Bibliothek (C,Java,Python,Firmware) Quelltextarchiv SVN Repository

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	

Pin	Bezeichnung	Aufrufname
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
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 Datei 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 heraufladen `at89prog.exe -u c:\test1.BIN *`
CPU Reset `at89prog.exe -r`

JTAG Adapter

TDB

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

TDB

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 Analsator (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 5us 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)

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

TDB

Downloads

GTKWave:

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

Snapshot Messung:

noch nicht fertig implementiert

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 7. Eigene Firmware entwickeln

Nachrichten dienen zum Austausch von Kommandos und Daten zwischen dem Computer und der Hardware. Die genauen Codes fuer die Funktionen und Kommandos stehen in der Header-Datei octopus.h.

Appendix A. Datenblatt

TDB

Appendix B. Schaltplan

TBD

Chapter 8. Apeendix C: Lizenzen

TBD

Bibliography

The bibliography list is an example of an AsciiDoc SimpleList, the AsciiDoc source list items are bulleted with a + character. The first entry in this example has an anchor.

[taoup] Eric Steven Raymond. *The Art of Unix Programming*. Addison-Wesley. ISBN 0-13-142901-9.

[walsh-muellner] Norman Walsh & Leonard Muellner. *DocBook - The Definitive Guide*. O'Reilly & Associates. 199. ISBN 1-56592-580-7.

Index