USBprog Handbuch

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

sauter@ixbat.de

USBprog Handbuch

by http://www.embedded-projects.net/usbprog

Revision History

Revision 0.1 November 2007 Revised by: H

Table of Contents

List of Figures

Preface

Im folgenden Handbuch wird die Inbetriebnahme und der Einsatz von Octopus in den Betriebssystemen GNU/Linux und Windows beschrieben.

Mit kleinen Beispielen wird gezeigt, wie von den verschiedensten Programmiersprachen aus auf Octopus zugriffen werden kann.

Support im Forum

Support gibt es im Forum auf http://forum.embedded-projects.net/

Weitere Hilfetexte/Beschreibungen

Weitere Hilfetexte bzw. Beschreibungen stehen im Wiki unter http://wiki.embedded-projects.net. Es können jederzeit die bestehenden Artikel angepasst, oder neue verfasst werden.

Chapter 1. Einsatzgebiet und Funktionsumfang

TDB

Chapter 2. Montage und Aufbau der Hardware

TDB

2.1. Löten der restlichen Bauteile

- 10-polige Wannenbuchse einlöten
- USB Buchse montieren und einlöten
- 4-polige Stiftleiste fixieren und einlöten

2.2. Programmierung der Firmware

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 m128 -c avrispv2 -P usb -B 8 -U flash:w:octopus.hex
- lfuse programmieren: avrdude -p m128 -c avrispv2 -P -B 8 -U lfuse:w:0xe0
- hfuse programmieren: avrdude -p m128 -c avrispv2 -P -B 8 -U hfuse:w:0xdd
- efuse programmieren: avrdude -p m128 -c avrispv2 -P -B 8 -U efuse:w:0xff

AVR Studio

TDB mit fuse ISP Speed

PonyProg

TDB mit fuse ISP Speed

Chapter 3. Installation der Treiber und Software für USBprog

Die zentrale Kompomente für Octopus auf dem Computer ist die Zugriffsbibliothek liboctopus. Sie bietet Funktionen für den Zugriff auf die einzelnen Schnittstellen und Möglichkeiten von Octopus an. Die Bibliotthek liboctopus realisiert die Kommunikation über die betriebssystemunabhängige Bibliothek libusb. libusb ist eine Funktionsbibliothek, die Anwendungsprogrammen den Zugriff auf USB-Geraete ermoeglicht. Daher kann Octopus auf jedem Betriebssystem, für das es die libusb gibt, eingesetzt werden.

Aktuell gibt es Portierungen für:

- GNU/Linux
- Windows (Win98SE, WinME, Win2k, WinXP)
- FreeBSD
- NetBSD
- OpenBSD
- Darwin
- MacOS X

Die Installation der Bibliothek ist die Grundvorraussetzung für Octopus.

3.1. Installation unter Windows

Unter Windows wurde die Installation mit einem Installer weitestgehend automatisiert.

- Octopus abstecken
- Download Installer (http://www.embedded-projects.net/octopus -> Unternavigation Downloads)
- Starten des setup.exe
- Anschliessend Octopus anstecken
- Treiber automatisch wählen selektieren
- Windows sollte den passenden Treiber finden

SCREENSHOT

- Wichtig ist, dass es auch die .dll datei gibt + .h
- Testprogramm
- Octopus immer noch nicht anstecken

- Wie octopus installiert wird, steht im Kapitel 5 Montage und Hardware.
- libusb muss installiert sein! So wie in Kapitel .. beschrieben
- Neuste Version herunterladen (Installer.exe)
- Octopus vom Computer entfernen
- Setup starten und durchfuehren
- Octopis an Computer anstecken
- Treiber aus dem Ordner auswaehlen . . . ..
- Fertig
- libusb testprogram
- Schauen in bla solle .dll und .h liegen (system32)
- Mit Test Base Programm kann man alles mal checken

Davon abhaengig, wie nun weiter gearbeitet werden soll, muss die Bibliothek in die entsprechende Programmiersprache eingebunden werden. Fuer einfache erste Tests kann das Beispielprogramm Octopus Base installiert werden.

3.2. Installation unter Linux

Die Installation unter Linux ist aufgeteilt in zwei Bereiche. Zuerst die Installation der Bibliothek libusb, was durch die verschiedenen Paketmanager unterschiedlich realisiert werden muss. Und als zweiten Teil, die Installation der Bibliothek liboctopus.

3.2.1. Die Bibliothek libusb installieren

Unter Linux sollte die Bibliothek libusb und wenn möglich die passenden Entwicklerdateien (Header-Dateien) über die Paketverwaltung installiert werden.

Zusätzlich ist empfehlenswert, sich ausserdem die Pakete usbview und usbutils zu installieren. usbview ist ein sehr praktisches Programm, mit dem angezeigt werden kann, was auf dem USB steckt. In den usbutils sind kleine Hilfsprogramme wie lsusb enthalten, mit denen geschaut werden kann, was vom Betriebssystem auf dem USB gefunden worden ist.

Debian/Ubuntu

apt-get install libusb-dev libusb-0.1-4 usbview usbutils

openSUSE und sonstige SUSE Versionen

- YAST starten
- libusb, libusb-dev, usbutils und usbview suchen
- Markieren für die Installation
- Installation starten

Fedore/Redhat

- Paketmanager starten
- libusb, libusb-dev, usbutils und usbview suchen
- Markieren für die Installation
- Installation starten

Aus den Quellen installieren

- Download http://libusb.sourceforge.net/download.html
- Entpacken: *tar xvzf libusb-0.1.12.tar.gz*
- Verzeichnis wechseln: *cd libusb-0.1.12.t*
- Übersetzung vorbereiten:*./conigure*
- Übersetzung starten: *make*
- Bibliothek installieren (als root): *make install*

Sind die Bibliothek libusb und die Tools installiert, kann geprüft werden, ob nach dem Anstecken von Octopus dieser ordnungsgemäss erkannt wird.

In der Ausgabe von lsusb sollten folgende Nummern mit auftauchen:

Example 3-1. lsbusb Aufruf

```
big:/home/bene# lsusb
Bus 003 Device 002: ID 1781:0c65
```

Das Programm usbview hingegen ermöglicht eine detailiertere Ansicht:

```
big:/home/bene# usbview
```

Figure 3-1. usbview Ansicht

usbview mit Octopus

Mögliche Ursachen, wenn Octopus nicht erkannt wird:

- Schlechte Lötstellen an der USB-Buchse

- Falsche Konfiguration der FUSE Bits
- Firmware falsch in den Flash des Mikrocontrollers übertragen

3.2.2. Die Bibliothek liboctopus installieren

- Neuste Version herunterladen (octopus.tar.gz)
- Entpacken: *tar xvzf octopus.tar.gz*
- Anpassen der Installation auf aktuelle Umgebung: *sh bootstrap*
- Vorbereiten der Uebersetzung: *./configure*
- Uebersetzung starten: *make*
- Bibliothek + Zugriffsrechte installieren (als root): *make install*
- Fertig
- schauen in bla solle .so und .a und .h liegen

Abhängig vom weiteren Arbeiten muss die Bibliothek in die entsprechende Programmiersprache eingebunden werden. Für einfache erste Tests kann das Beispielprogramm Octopus Base installiert werden.

3.3. Installation unter MAC/OS

folgt (bitte im Forum nachfragen)

Chapter 4. Firmware wechseln

so gehts. zu beachten ist in windows und linux bla bla

4.1. usbprog-gui

in Windows erst manuell Bootloader starten! in Linux Rechte beachten

4.2. usbprog Konsole

in Windows erst manuell Bootloader starten! in Linux Rechte beachten

Chapter 5. Anwendungen mit USBprog

TDB

5.1. 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.

5.1.1. 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 Stuido Treiber (Jungo) finden und aktivieren.

5.1.2. 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 Stuido 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 Vereichnis c:\WinAVR\utils\libusb.

5.1.3. 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 neuste Quelltextarchiv zuvor heruntergeladen werden. Die aktuelleste 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
```

5.1.4. avrdude Bedienhinweise

TDB

Ein Aufruf um die Signatur eines ATMega32 auszulesen sieht wie folgt aus:

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

5.1.5. 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)

5.2. OpenOCD ARM7/ARM9 Debugger

TDB

5.3. 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.

5.3.1. Status

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

5.3.2. Downloads

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

5.3.3. 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.

5.3.4. 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:
 IO 1
    IO 2    IO 3    IO 4    IO 5
    IO 6
    IO 7
    IO 8
 0   0       0       0       0       1       1       1
```

```
(als Hexzahl ist das: 0x0F)
```

```
Mit der Funktion können nur die Datenrichtungen für IO 1 - I0 8 angebeben werden! Die für I
```

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);
```

5.3.5. 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;
  }
```

5.3.6. 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();
      }
    }
  }
```

5.3.7. Beispiel in Python

```
import simpleport
  import time

  if __name__ == "__main__":
      # call simpleport_open() to retrive 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)
```

5.4. 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

5.4.1. 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 Eingansleitung 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)

5.4.2. 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)
```

5.4.3. 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.Write("Answer: ");
                        Console.WriteLine(USBProg.ReadExisting());
                        Thread.Sleep(500);
                        //enable LED
                        USBProg.Write("pB1*");
                        //sensorStream.Read(buffer,0,2);
                        Console.Write("Answer: ");
                        Console.WriteLine(USBProg.ReadExisting());
                        Thread.Sleep(500);
                }
                //close comport
                USBProg.Close();

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

TDB

5.5. USB zu RS232 Wandler

TDB

5.6. AT89 Programmer

TDB

5.7. JTAG Adapter

TDB

5.8. 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.

5.8.1. Anschlussbelegung

10-polige Stecker Beschreibung (Pin Nr.) Pin Pin Beschreibung TDI 1 2 VCC

```
        3       4
TMS
        5       6
TCK
        7       8
TDO
        9       10      GND
Belegung gilt für usbprog 2.0 und 3.0
```

5.8.2. 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>

5.8.3. 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

5.9. 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-Dateinen, 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

5.9.1. Anschlussbelegung

10-polige Stecker Beschreibung (Pin Nr.) Pin Pin Beschreibung Logik 6 1 2 VCC Logik 5 3 4 **Logik 4 Logik 1 5 6** Logik 3 Logik 8 7 8 **Logik 2 Logik 7 9 10 GND** Belegung gilt nicht für usbprog 2.0

5.9.2. Downloads

GTKWave:

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

5.9.3. 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 Simmulation analysieren zu können. Der Vorteil dieses Datei Formates ist, dass es bereits einige Programme zum be- und verarbeiten gibt (unteranderem 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

5.9.4. 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 6. 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 7. 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 Definative Guide*. O'Reilly & Associates. 199. ISBN 1-56592-580-7.

Index