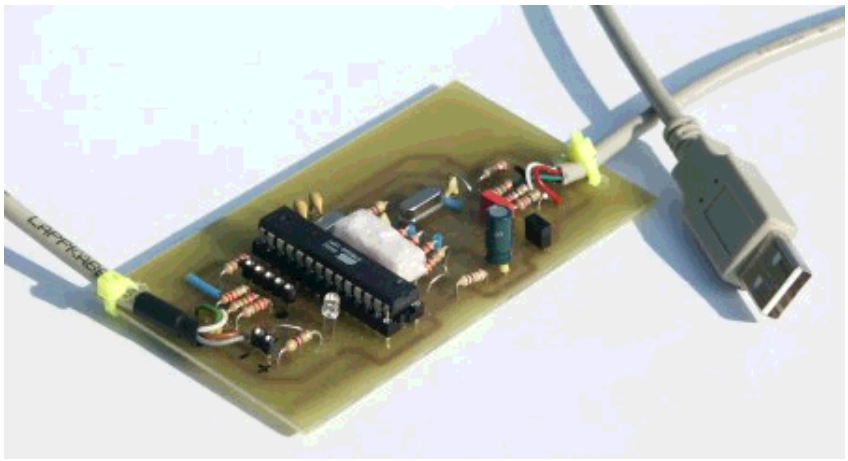
Content:

AvrUsb500 -- an open source Atmel AVR Programmer, stk500 V2 compatible, with USB interface



By Guido Socher
<guido_at_tuxgraphics.org>

Abstract:

It took me about 4 month to develop the software and hardware presented in this article. Especially coding the stk500 specification from scratch in C was not easy. The result was however worth the effort. I really like this new programmer and I am sure you will like it too.

In this article we will design a state of the art USB programmer for the AVR microcontrollers from Atmel. The programmer firmware has no device dependent data. Therefore it works for almost any AVR microcontroller on the market and possible future microcontrollers.

This USB programmer has, unlike other programmers, no "chicken and egg problem". That is: you can build it from scratch without the need of another programmer to load the initial firmware.

The firmware is open source and programmed in C according to the AVR068 specification from Atmel.

The avrusb500 is available as a kit from http://shop.tuxgraphics.org

---

Why Stk500 and USB?

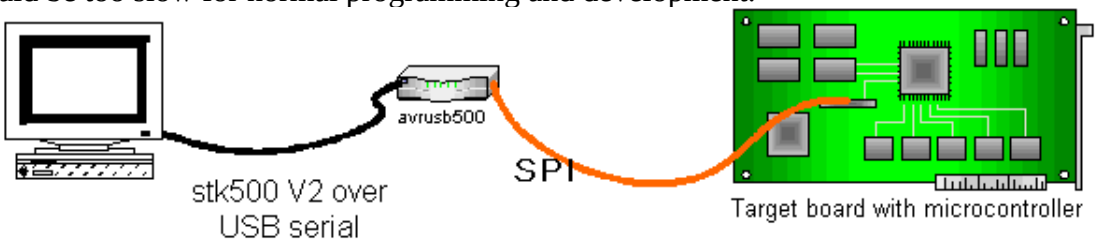
Until the beginning of this year a simple parallel port programmer was the only good programmer as it could be used for any device. All device dependent information is stored in the programmer software on your computer. The problem is however that the parallel port is slowly disappearing. Apple does not have it at all and the smaller laptops don't have it either anymore. It's time to look for alternatives.
Atmel had at last a good idea and published together with the avrstudio version 4.11 a completely redesigned stk500 protocol. The new stk500 protocol is totally incompatible with version 1 of the stk500 protocol but it is the right solution.

The stk500 AVR068 specification needs no longer device dependent data structures in the programmer hardware. All device dependent logic is now in the programmer software running on the PC. As of today avrstudio (for windows) and avrdude (for multiple OSs including Linux) support this version 2 of the stk500 protocol.

The physical hardware interface to the PC is USB. This way we can build a fast and modern programmer which can be used under Linux, BSD, Windows and MacOS X.

The design idea behind this USB AVR programmer

The programmer will be an In System Programmer. That is: you do not have to remove the microcontroller from the circuit. Almost all Atmel microcontrollers have for this purpose an interface called SPI (Serial Peripheral Interface). The microcontrollers can be either spi master or spi slave. A spi slave is the microcontroller which is programmed and the programmer is the master. The master is controlling the clock (pin SCK) on the SPI interface. The purpose of the programmer is therefore to "convert" the stk500 protocol to spi commands. To run spi directly over usb would be too slow for normal programming and development.



The spi protocol is implemented in the microcontroller hardware. We just enable the spi master as described in the datasheet (see references at the end).

The stk500 protocol defines a message format and appropriate commands such as CMD_SET_PARAMETER, CMD_LOAD_ADDRESS, CMD_PROGRAM_FLASH_ISP etc... The

protocol sends chunks of data and this is very suitable for USB. The whole protocol is documented in the application note AVR068 (stk500 V2) from Atmel. This stk500 protocol must not be confused with AVR061, which is version 1 of the stk500 protocol and has absolutely nothing to do with version 2. Atmel has done a complete re-design.

Solving the chicken and egg problem

Since this programmer contains already a microcontroller we must find a way to initially program it. For this we need a programmer ... the thing that we are about to build...

For the usb interface we use a FT232bm chip. This chip has an interesting mode called "bit-bang" mode. I have written a library called ftdibb (only 2 files .c and .h, available from http://linuxfocus.org/~guido/) which implements this bitbang mode on top of the libUSB library. LibUSB (http://libusb.sourceforge.net) uses the /proc file system to send and receive custom usb messages to any usb device on the usb bus. One problem is that this requires root permissions under Linux, the other problem is that we will send very short messages. Initially (when there is no firmware on loaded in the microcontroller) we have absolutely no storage place and no logic in the programmer. We must send all the commands directly from the PC, bit by bit. USB is fast when you send long chunks of data but it is very slow when you send only 1 bit at a time.

This is however acceptable as it saves us from having to get first a different programmer to load the initial firmware into this programmer. Therefore it is acceptable and you load the firmware only once.

In other words the avrusb500 consists really of two programmers: One internal for the initial loading of the firmware and the actual avrusb500 programmer which is a fast and stk500 V2 based programmer for every day use.

I called the bitbang programmer for the initial loading "bbpg". I have modified the uisp programmer software for this purpose. You need to download uisp-20050207.tar.gz and then apply the uisp-20050207-usb-bbpg-patch.txt patch (cd uisp-20050207;patch -p1 < uisp-20050207-usb-bbpg-patch.txt) or you can take the already patched sources (uisp-20050207-usb-bbpg.tar.gz, download at the end). Libusb needs to be installed before you compile the bbpg programmer. The avrusb500-X.Y.tar.gz package contains also a patched and pre-compiled binary. There is no guarantee that a pro-compiled will run on any linux distribution due to the dependencies on the library versions but it will run on many. So if you want to save some compile time then try this one first. You must name this version of uisp "uisp_bbpg". This is what the makefiles and scripts expect.

The CD which is sold from http://shop.tuxgraphics.org together with the parts for this programmer can also be used for the initial loading of the firmware. It has the additional advantage that you can do this from the CD without root permissions.

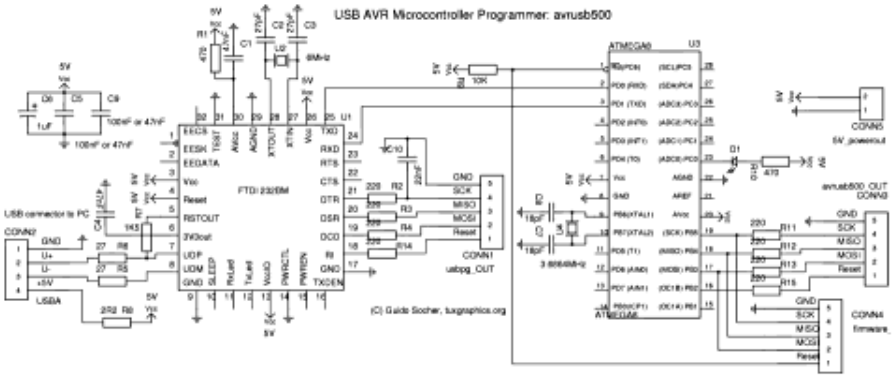
The hardware

The specification from Atmel for the STK500 communication protocol is 37 pages long. It is however not a problem to fit it into an atmega8 microcontroller. It fills about half of the available memory.

Here is the schematic drawing. The left side is the usb to rs232 conversion circuit to provide usb

connectivity to the atmega8. The whole circuit is self powered. That is: unlike may other programmers (actually all, except for the simple parallel port programmers) you do not need additional pins to draw the power from the target circuit.

We can therefore continue to use the small little connector introduced with the (Programming the AVR microcontroller with GCC, libc 1.0.4). This is in my opinion the right right solution as you don't waste space on the PCB for a bulky connector which is used only once.



We need an external crystal for the atmega8. This is because the UART for the serial communication via usb needs to run at 115.2K baud and this can only be done with a 3.6864MHz crystal. Why do we need 115.2K baud? This is mainly because AVRstudio from atmel is really stubborn. You can't change the baud rate there. If you plan to use this programmer only in combination with avrdude then you can set the baud rate to 19200 and use the internal 4MHz oscillator. The speed will be a little slower but you will hardly notice it as the baud-rate is not the bottleneck.

Soldering SMD chips

I have designed the hardware such that conventional parts can be used for almost all components. Only the ft232 is a SMD chip. It is not available in any other package and that is also true for all other usb chips on the market.

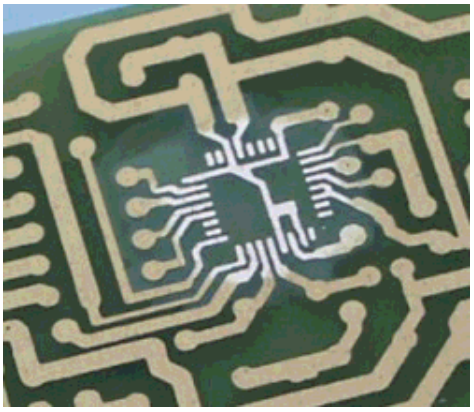
Soldering a SMD chip is a little challenge. The problem is that the chips are rather small and there is little space between the pins. If you are not careful you can easily solder several pins together. To correct such a mistake is not easy and you may destroy not only the chip but also the PCB. If you are not sure if you have the skills and the right equipment then buy a board with the SMD parts already soldered on.

In any case here are some smd soldering tips:

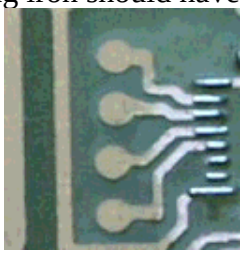
- Solder all smd chips first. You can do this before or after drilling the holes but in any case the smd chips should be soldered on before any other parts.

- The board should be 100% clean and the pads for the smd chip coated with solder flux. Do

not use flux which contains aggressive chemicals. Such flux is for repairing a water spout. What you need is e.g known as "no clean flux". It is a solder flux which will mostly evaporate and any remains can stay on the board.

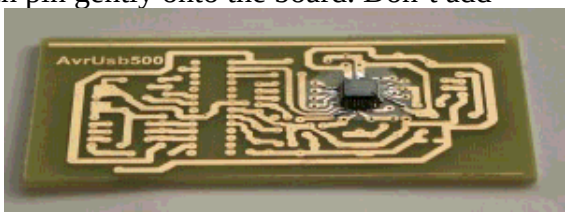


- Put tiny amounts of solder with soldering iron onto each pad. The soldering iron should have a 0.8mm tip (or smaller) and you should use 0.5mm SMD solder wire.



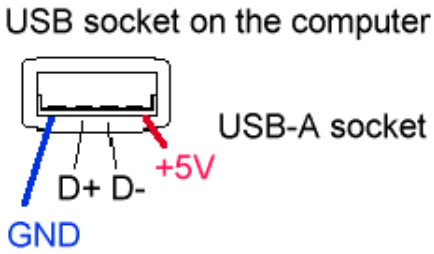
- Position the chip on the pads and solder just one pin onto the board. Check carefully the position again and then solder a pin on the opposite side onto the pcb. Just press the pin gently with the tip of the soldering iron and don't add any more solder.

- Now go around the chip pin by pin and press each pin gently onto the board. Don't add solder.



Testing the hardware

Check again the polarity of the USB connection. Incorrect polarity will destroy the circuit.



The first test is to just plug in the usb connector and check (under linux) the file /proc/bus/usb/devices. You should see a new entry which looks like this:

```
guido@brain
T: Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 2 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0403 ProdID=6001 Rev= 2.00
S: Manufacturer=FTDI
S: Product=USB <-> Serial
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr= 90mA
```

I: If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=serial
E: Ad=81(I) Atr=02(Bulk) MxPS= 64 Ivl=0ms
E: Ad=02(O) Atr=02(Bulk) MxPS= 64 Ivl=0ms

This shows that the ft232 is working.

Unpack the avrusb500-X.Y package (can be downloaded at the end of this article). E.g:

```
tar zxvf avrusb500-0.6.tar.gz
cd avrusb500-0.6
```

The avrusb500-X.Y contains two more test programs which test the microcontroller and the whole communication to your PC. Load them also. How to do this is described in the README file inside the avrusb500-X.Y package. You will basically just connect the bridge cable (CONN1 to CONN4) and execute a command like

```
make load_test_1
```

but please have a look at the readme file (command: more README).

BitBang loading of the final firmware

Loading of firmware is done under Linux only. The programmer is OS independent once you have loaded the firmware.

Plug in the small bridge cable between the connectors CONN1 and CONN4. Pay attention to correct polarity. If you have ordered the kit and the CD from the tuxgraphics shop then you just execute the command

```
make load
```

from the unpacked avrusb500-X.Y package. That’s all.

If you do not have this CD then you need to build first the special version of uisp which contains the ppbg programmer (see above). The README file inside the avrusb500 package describes this procedure also.

Loading of the firmware takes very long time due to the huge overhead you get on the USB bus when you send just one bit at a time. You can calculate approximately 20minutes for loading and 20minutes for verification.

Using the USB AVR programmer

This programmer is designed to be developed in a Linux environment. However once build it is truly OS independent. You can use it with the AVRstudio4.11 for windows or under Mac OSX or Linux or BSD Unix ...

The software to use the avrusb500 programmer under Linux is avrdude (http://savannah.nongnu.org/projects/avrdude/). You need version 5.0. As of this writing one a beta version of avrdude-5.0 was available. By the time you read this article the final version may already be available. The beta version has a few bugs. You still need a patch to get it to work. Both the beta version and the patch can be downloaded at the end of this article.

The command to load the code MyCode.hex into an ATmega8 would be:

```
avrdude -p m8 -c avrusb500 -e -U flash:w:MyCode.hex
```

The configuration file entry in the avrdude.conf file is:

```
default_serial     = "/dev/usb/tts/0";
#or
#default_serial     = "/dev/ttyUSB0";

# ... and further down:
programmer
  id    = "avrusb500";
  desc  = "Atmel AVR ISP V2 programmer from tuxgraphics";
  type  =  stk500v2;
;
#
```

Conclusions

This is the first modern USB based AVR programmer which can be build from scratch without the need to find a programmer to program the programmer.

If you like this type of articles then have also a look at http://shop.tuxgraphics.org. It is always nice to see that there are people who support my work.

Have fun and happy soldering!

References

- Datasheet of the atmega8 [2.5Mb, pdf] the SPI interface is described in this datasheet.
- The AVR068 specification (description of the stk500 V2 protocol): stk500_spec_AVR068.pdf, 0.5Mb
- If you want to learn more about the BitBang mode of the ftdi chips then take a look at my BitBang library (ftdibb): http://linuxfocus.org/~guido/
- **Software, documents and future updates:Download page for this article**
- avrdude-5.0-BETA.tar.gz, avrdude-5.0-BETA-stk500-patch.txt
- uisp-20050207-with-usb-bbpg-patch.tar.gz, uisp with the bbpg patch already applied.
- A complete kit to build this programmer is available from shop.tuxgraphics.org

---

<--, tuxgraphics Home	Go to the index of this section
