Proggy II, an in-system programmer for AVR microcontrollers

By Jurij Mikeln

At AX elektronika, we have made many programmers, both for 89C/89Sxx51 and AVRs. To tell the truth, we have sold many more AVR programmers - which is no coincidence. AVRs are more popular than their 89C/89Sxx51 counterparts. It was about time to replace the old Proggy-AVR's nice blue but somehow bulky programer.

Proggy-AVR has been very popular for the last five years or so. It had a nice blue enclosure which made it attractive and distinctive from other programmers (that often had no enclosure at all). Now we are making a renovated, faster AVR programmer that is both less expensive and smaller. No wonder we expect it to be well accepted within the AVR programmer's community.

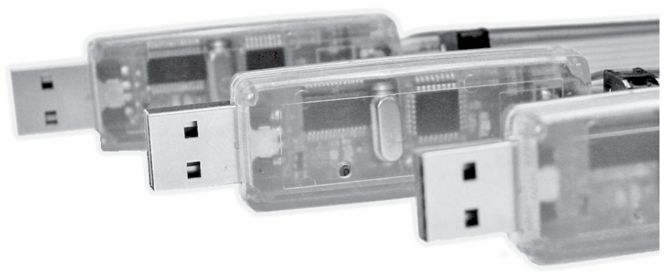
Schematic diagram

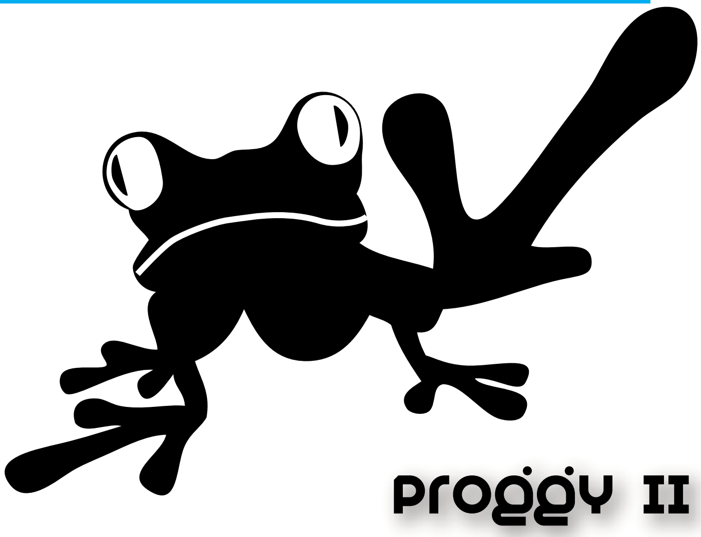
The schematic diagram shown in Figure 1 can be divided into two parts: the FT232RL USB interface circuitry and the ATmega8 microcontroller circuitry.

In the schematic diagram, you can see that we have replaced the FT232BL, manufactured by FTDI [1] with the more modern FT232RL (which needs fewer support components and offers more functionality). One such feature is a clock output pin. The FT232RL can generate different clock signals. We can use such a clock signal to unlock an incorrectly-programmed AVR chip. This is a handy feature because one doesn't always have a clock source available. We have added two LEDs to indicate Rx and Tx signals. This is handy to quickly see whether or not the programmer is alive and programming.

The second part of the schematic diagram is the ATmega8 microcontroller. The ATmega8 handles all of the signals for communications with the PC host, via FT232RL chip and also generates all the signals necessary to program the target microcontroller.

Surrounding the ATmega8 you can see two connectors: CON1 is meant for the factory programming of the ATmega8 microcontroller, while CON2 represents the ISP connector that serves to program the AVR microcontroller on the target board. You'll notice there is also a CLK12 pin, coming from the clock signal that the FT232RL generates.





Beside the Rx and Tx LEDs, there is also a LED which indicates whether the target microcontroller was programmed correctly: if the LED blinks after programming, it means that the target microcontroller was not programmed correctly. If the LED lights steadily the target microcontroller was programmed correctly.

The programmer also has three built-in protective resistors on the MISO, MOSI & SCK lines which protect both the target and host microcontrollers. Note that an 18.432 MHz crystal was used in this circuit, although ATmega8 is specified to only work at up to 16 MHz. That is true in harsh environments with extremely low or high temperatures. But, that is not likely with this programmer, so this higher frequency clock is perfectly satisfactory.

PCB and soldering

When designing the PCB we wanted it to be small - and fit into a "nice" translucent USB stick enclosure as shown in Figure 6. Hence, we designed a double-sided PCB that fits nicely into such a USB enclosure. The PCB is shown in Figures 2 through 4.

Due to the relatively small enclosure, we had to use small components: typically 0603 SMD components. Also we

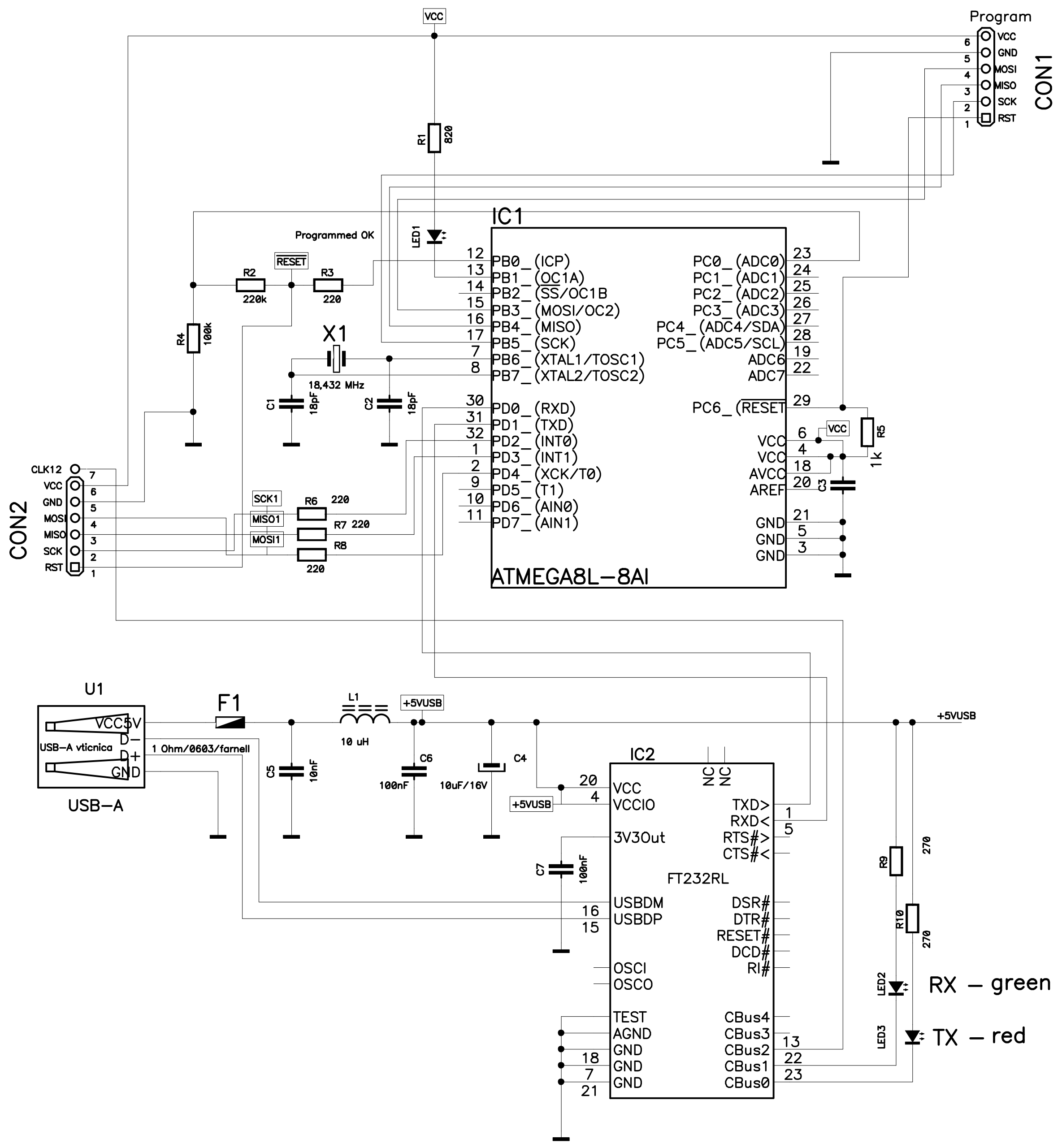


Figure 1: The Schematic diagram of Proggy II

used an SMD crystal with a low profile. A standard profile crystal would not have fit into such a small enclosure. The FT232RL we used is not the smallest one FTDI makes, but it's in an SMD case which can be hand-soldered. There is also an FT232RL in a QFN case, which is virtually impossible to hand-solder.

When soldering, use a good soldering iron with a very fine tip to enable you to solder theFT232RL and 0603 parts. When soldering, I strongly suggest you do so under a suitable stereo microscope. If you're not confident soldering such a small components under a microscope, leave that task to the professionals.

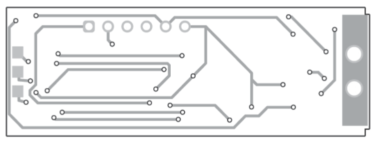


Figure 2: PCB layout, solder side

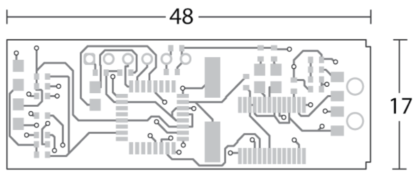


Figure 3: PCB layout, top side

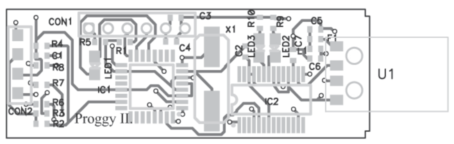


Figure 4: PCB layout, top silk side

Solder the two ICs first, followed by the smaller parts, like the 0603 resistors and capacitors, then the LEDs. Make sure that the LEDs are properly oriented. You can check this with a multimeter on the diode - check range, by connecting the positive terminal of the multimeter to one LED's pin and the negative terminal to the other. If the LED glows then the anode is the one connected to the positive terminal of the multimeter. Finally, you can solder the crystal and connector U1. As mentioned, CON1 is meant for factory programming - you need not mount this component.

All components are placed on the top side, including USB connector U1. The programming connector (CON2) consists of 6 wires (one more if you need the CLK signal). All these wires are part of a 10-pin flat cable, which is terminated with an IDC male connector. The CON2 wiring is shown in Figure 5.

When all the soldering is done, check the PCB under a microscope, or magnifying glass, for short circuits, noting especially the FT232RL chip. Note that pins 25 and 26 are shorted together, which is normal, but if any other pins are shorted you should remove that solder bridge with desoldering wick.

Before testing, you must program an ATmega8 using the ISP firmware, which can be downloaded from our web page. If you do not have programming capability already, you can always contact our distributor to purchase a pre-programmed ATmega8.

At this stage, the Proggy II programmer is ready to start programming AVR devices. Optionally, you can elect to "customize" the FT232RL with Proggy II firmware. This can be done using the free MPROG software that can be downloaded from FTDI's web page. This customizes the various USB device descriptors to make the Proggy II identify itself as such, and not an FTDI USB-serial bridge. The programming procedure follows, see Figure 7.

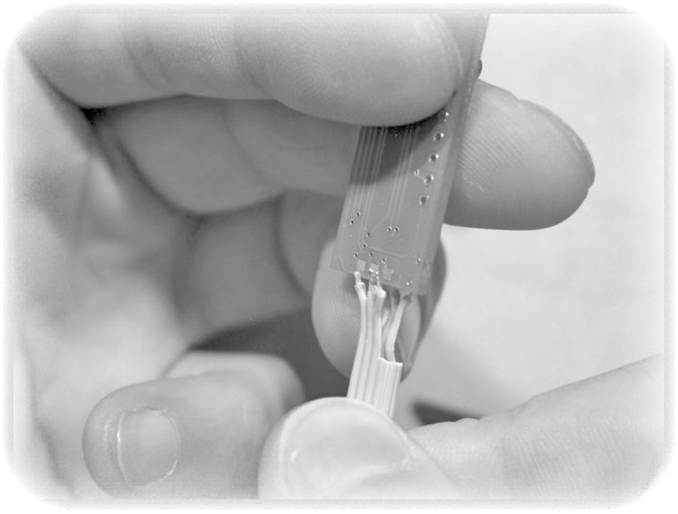


Figure 5: Detail of CON2 soldering

Figure 6: A suitable enclosure for Proggy II

When initially programming the FTDI chip, click F5 or Devices/Scan and parse.

The window will fill with data from the empty chip. Now load the template (for example for Proggy II) by clicking File/Open template. Chose the Proggy II template and click ***Device Program***. You will see the data being programmed into FT232RL chip.

You are now ready to connect your programmer to your

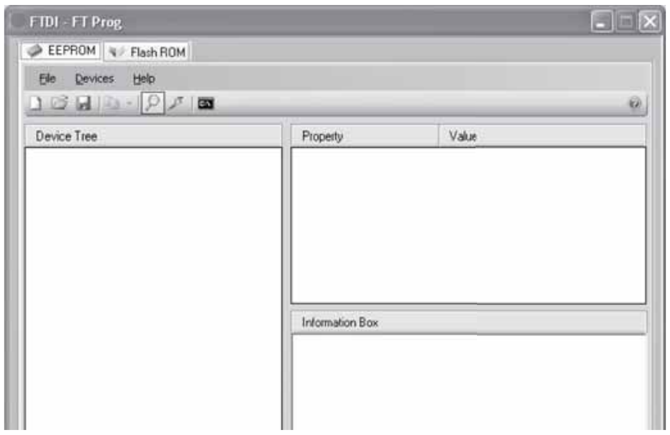


Figure 7: F-PROG programmer for the FT232RL chip

PC using a USB cable, and to your MiniPin II development board or your target microcontroller board. When you connect the programmer to the USB port on your PC, you should see the Rx and Tx LEDs blink few times, which indicates that USB communication has been established.

Using Proggy II

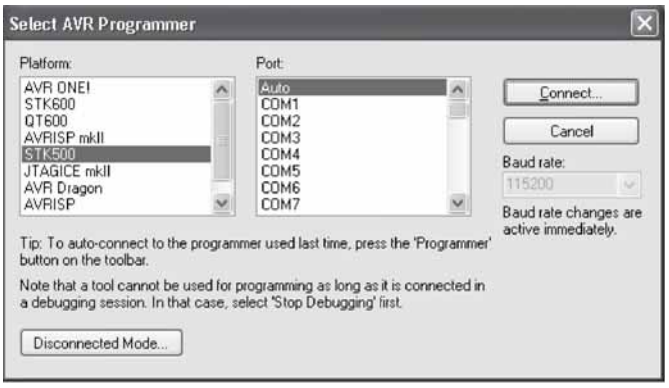
Installation

Upon initial connection to your PC, MS Windows® will check to see if suitable drivers are already installed. If not, you can find them on web-site (www.ftdichip.com) or on our web page.

After a successful driver installation MS Windows®, will assign a COM port to your Proggy II. Please note that for proper software operation, it ***must*** be in the range from 1 to 9. If it's not, then you should change it using Device manager.

NOTE

For a detailed description of the process of changing a COM port's assigned number, see the chapter on the MegaPin development board.



Select AVR Programmer

Platform:
- AVR ONE!
- STK600
- QT600
- AVRISP mkII
- STK500
- JTAGICE mkII
- AVR Dragon
- AVRISP

Port:
- Auto
- COM1
- COM2
- COM3
- COM4
- COM5
- COM6
- COM7

Connect... | Cancel

Baud rate: 115200

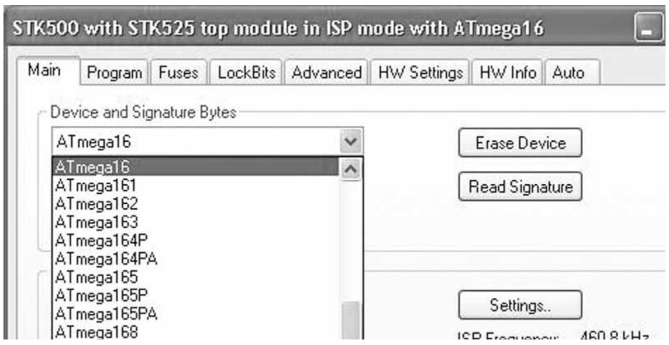
Baud rate changes are active immediately.

Tip: To auto-connect to the programmer used last time, press the 'Programmer' button on the toolbar.

Note that a tool cannot be used for programming as long as it is connected in a debugging session. In that case, select 'Stop Debugging' first.

Disconnected Mode...

Figure 8: Choose COM port



STK500 with STK525 top module in ISP mode with ATmega16

Main | Program | Fuses | LockBits | Advanced | HW Settings | HW Info | Auto

Device and Signature Bytes

ATmega16

- ATmega16
- ATmega161
- ATmega162
- ATmega163
- ATmega164P
- ATmega164PA
- ATmega165
- ATmega165P
- ATmega165PA
- ATmega168

Erase Device | Read Signature | Settings..

ISP Frequency: 460.8 kHz

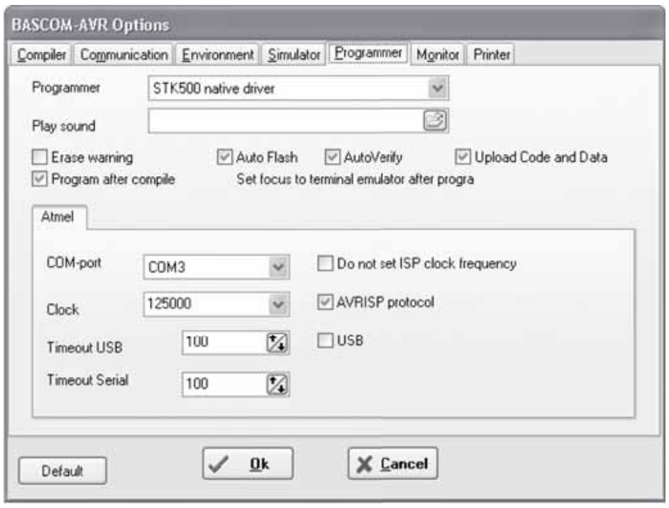
Figure 9: Choose device

Programming

Programming is simple. Start AVR Studio (Ver. 4 or lower) and click the icon ***Con***, set the COM port to AUTO, choose STK500 or AVRISP and click ***Connect***.

Figure 10: Proggy II

Then click Program and choose the device that you wish to program. To test the connection between Proggy II and the microcontroller you may click ***Fuses***, to see if the programmer reads the fuses correctly. If all is OK, you'll see how the fuses are set. If not, you should change the ISP frequency to a lower value. Keep dropping the frequency until you get valid connection.



BASCOM-AVR Options

Compiler | Communication | Environment | Simulator | Programmer | Monitor | Printer

Programmer: STK500 native driver

Play sound:

☐ Erase warning ☑ Auto Flash ☑ AutoVerify ☑ Upload Code and Data

☑ Program after compile Set focus to terminal emulator after progra

Atmel

COM-port: COM3 ☐ Do not set ISP clock frequency

Clock: 125000 ☑ AVRISP protocol

Timeout USB: 100 ☐ USB

Timeout Serial: 100

Default | Ok | Cancel

Figure 11: Set up Bascom to use the STK500 native driver

Alternatively, for use with Bascom-AVR, you can set the programmer to ***STK500 Native driver*** as shown in Figure 11.

Note that you must check off all of the boxes shown in Figure 11, and set the COM port to the value you found using Device manager, and then click OK. Although Proggy II is connected to USB do not tick the USB window (since Proggy II is using the FT232RL in serial port emulation mode)!

Programming with the STK500 native driver is similar to programming within the AVR Studio program. You can also set the fuse bits as shown in Figure 12.

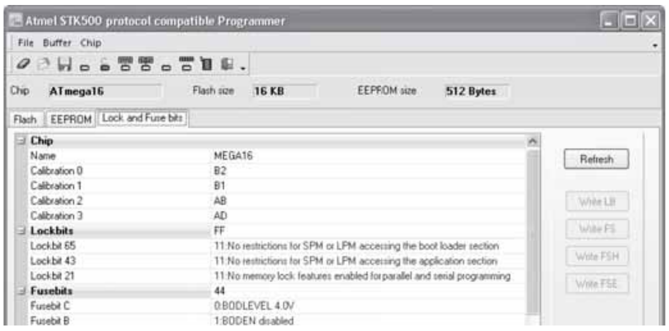
You can check the EEPROM's content, erase the microcontroller and do everything that can be done within AVR Studio, except that you cannot program ELF files. Elf files (***E***xecutable and ***L***inkable ***F***ormat) are special files, produced by some compilers, which contain the image data for Flash, EEPROM and Fuse bits, all in one file

Conclusion

The Proggy II programmer is a simple, but effective ISP programmer for AVR microcontrollers. It's STK500 compatible so it works very well with Bascom-AVR and AVR Studio ver. 4 or lower. Proggy II can also power target device like MiniPin II or any other microcontroller board providing that it does not draw more than 150 mA.

For protection of the USB port on your PC there is a fuse built into Proggy II. This fuse has done its job many times many times when I overloaded it!

__Enjoy programming with your Proggy II!__



Chip ATmega16 Flash size 16 KB EEPROM size 512 Bytes

Chip	
Name	MEGA16
Calibration 0	B2
Calibration 1	B1
Calibration 2	AB
Calibration 3	AD
Lockbits	FF
Lockbit 65	11:No restrictions for SPM or LPM accessing the boot loader section
Lockbit 43	11:No restrictions for SPM or LPM accessing the application section
Lockbit 21	11:No memory lock features enabled for parallel and serial programming
Fusebits	44
Fusebit C	0:BODLEVEL 4.0V
Fusebit B	1:BODEN disabled

Figure 12: Setting Fuse bits from within the STK500 native driver

Links

1. Future Technology Devices International LTD., URL: www.ftdichip.com
2. http://www2.atmel.com - ATMEL AVR studio