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

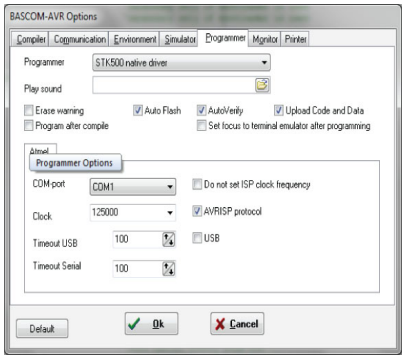
The schematic diagram shown in Figure 1 can be divided into two parts: the FT232RL USB interface circuitry and the ATmega8 microcontroller circuitry.

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.



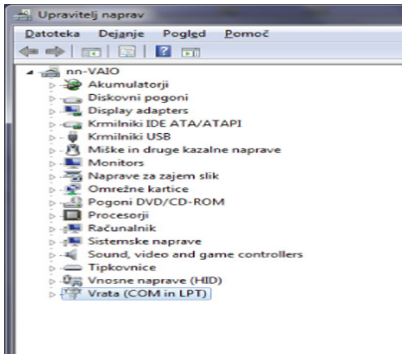
Beside the Rx and Tx LEDs, there is also a LED which indicates whether the target microcontroller was programmed correctly: if the LED blinks after programming, it means that the target microcontroller was not programmed correctly. If the LED lights steadily the target microcontroller was programmed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

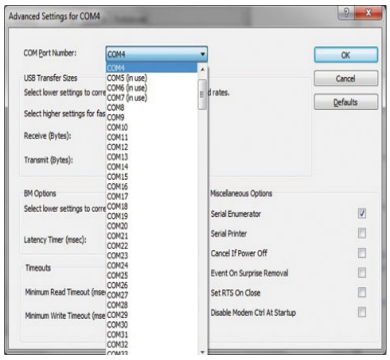
Due to the relatively small enclosure, we had to use small components: typically 0603 SMD components. Also we used an SMD crystal with a low profile. A standard profile crystal would not have fit into such a small enclosure. The FT232RL we used is not the smallest one FTDI makes, but it’s in an SMD case 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.

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 - c

heck 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. 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.

Proggy II, an in-system programmer for AVR microcontrollers

2012_AVR_UK_129

Continue reading

Shop area