The Low Cost AVR Starter Guide

By Colin O'Flynn

So your interested in getting into Atmel AVR microcontrollers, but want to do so on a budget, as best expressed by Adam Johnson as "nothing squared". If you can afford it, you really should consider the STK500 for $80 US, or any of the other development kits for the AVR (there is a large list on AVRFreaks.net under the tools menu). A development kit makes things much easier if you are just starting out.

Don't fear though, it IS possible to get into AVR's for little more than the cost of the microcontroller. This guide should help you do this, and will provide a handy guide to doing so without really knowing what your doing (don't worry, it will all come together in the end).

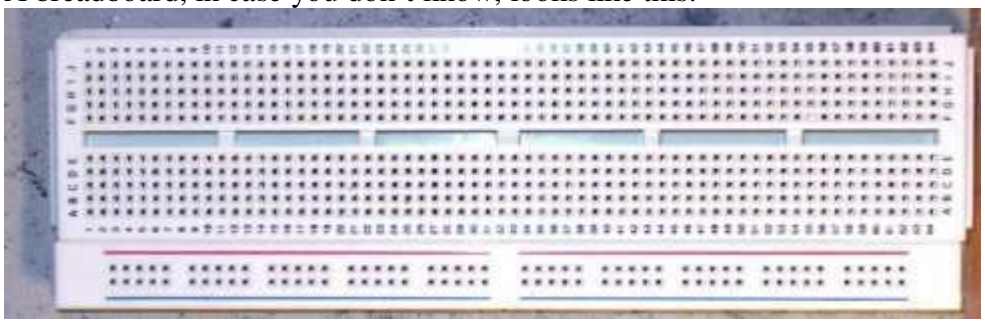
Tools You Will Need

There are basically only a few tools you will need to get started with the AVR microcontrollers. These are:

- Breadboard
- Five volt regulator for the AVR
- Programmer for the AVR
- Programming Software for the AVR
- Compiler for the AVR

Breadboard

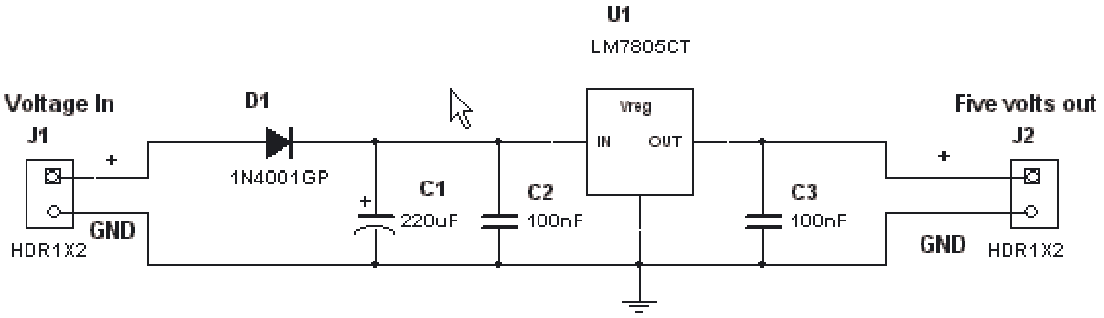
A breadboard, in case you don't know, looks like this:



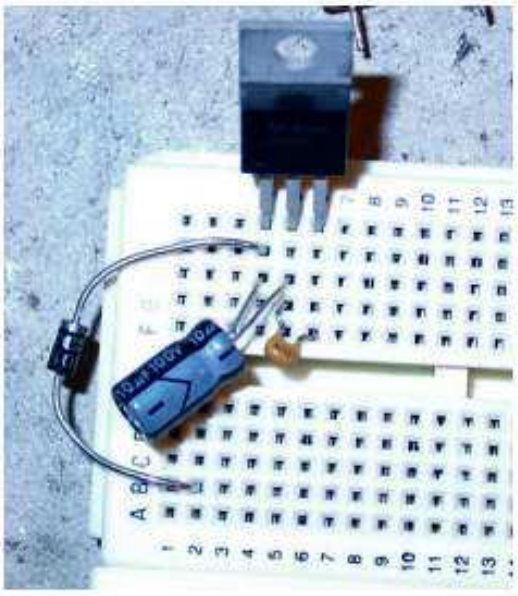
You really will need one of these, its pretty important to getting started. Otherwise you will have to solder everything, and that is really not preferred. You can pick one up at Radio Shack, or any local electronics store.

Five Volt Regulator

The easiest way to get five volts is from a hand little chip called a LM7805, which is a five volt regulator chip. It provides a stable five volts from a varying supply voltage of at least 7.5 volts up to around 15 volts (some can run off lower voltages, but to be safe keep the input to the LM7805 above 7.5 volts). Here is a schematic diagram showing how to hook up the regulator:



And here is the pin out of the regulator, as well as what the circuit will look like:



You might also want to mount the LM7805 on a heat sink, or if you don't have a heat sink a scrap of metal with a hole in it. However if you aren't going to be drawing much current than you should be fine without the heat sink. Also don't omit the small value capacitors on the LM7805, they are needed in case your input supply isn't that clean. The exact value of the capacitors isn't that important, so if you use a 100uF instead of a 220uF that will be fine. Note that 100nF = 0.1uF, 10nF = 0.01uF, 1nF = 0.001uF

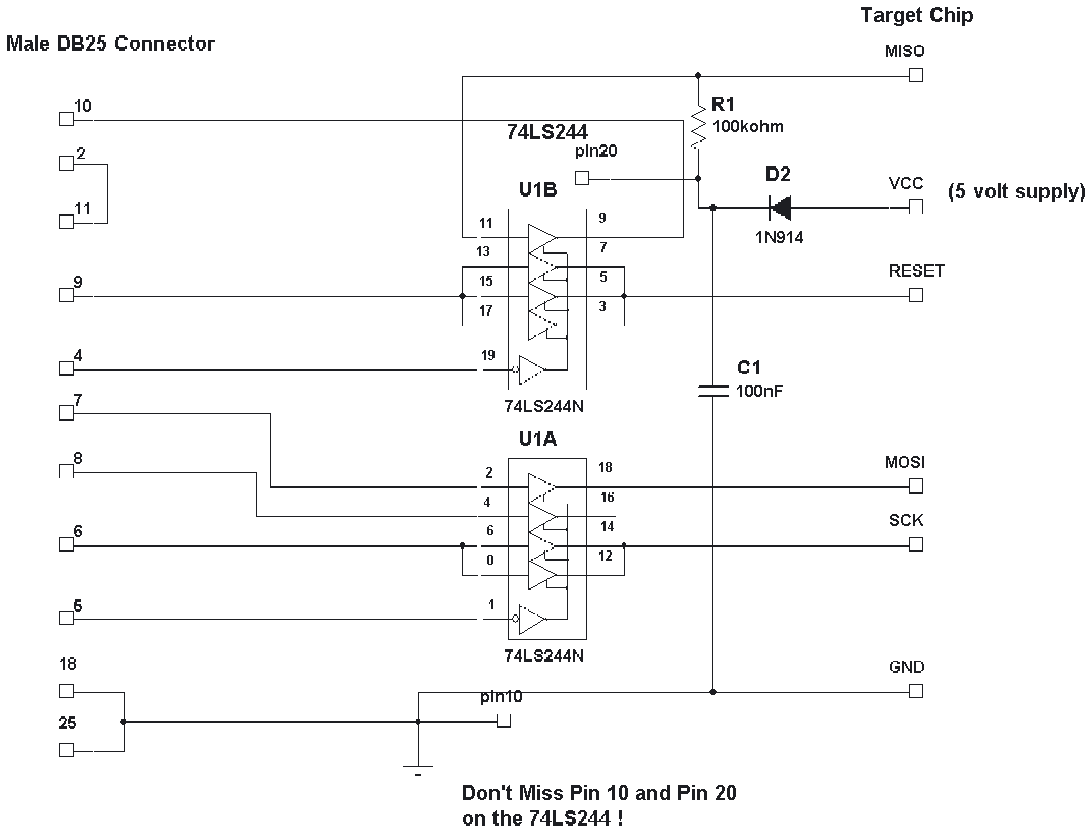
Once you assemble the circuit, test it out. Connect some voltage source (such as a wall wart) that is between 8 to 14 volts to the input of the regulator. It is important you keep

the diode in place, as eventually you will connect the power up backwards, and severe damage will be done without that diode. Check the output of the regulator with a voltmeter, to make sure it is 5 volts. If it isn't go back and check your wiring.

Programmer

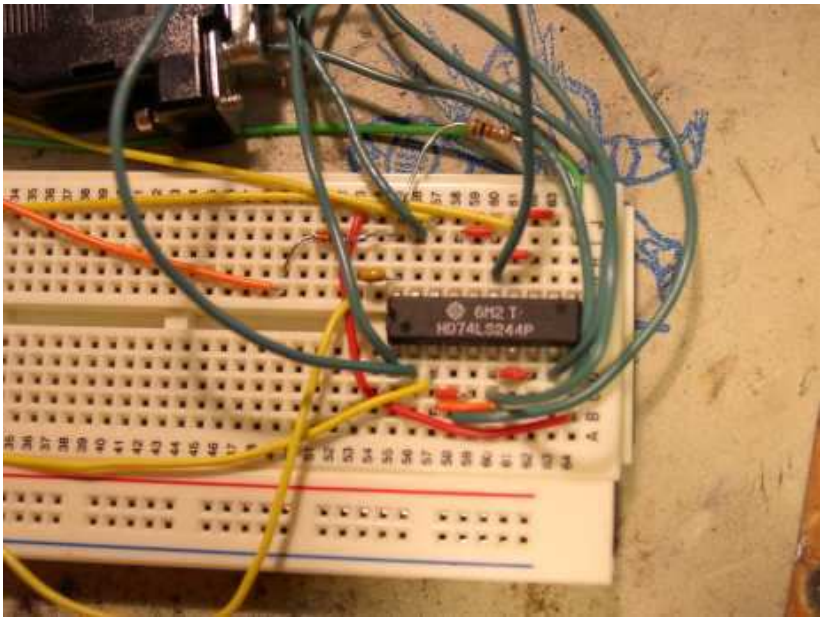
Note: The programmer involves messing around with your computer parallel port, and there is chance that damage could result to your computer. I will not be held responsible if anything happens to anything or anyone as a result of using this programmer, nor will Atmel or AVRFreaks.

The programmer is a very important part of using the Atmel AVR. The programmer lets you download your code to the AVR. Now lets take a look at how to make this special programmer. As it will be used quite a bit, I have to recommend shelling out a few extra dollars and using a buffer chip, as shown in the schematic below:

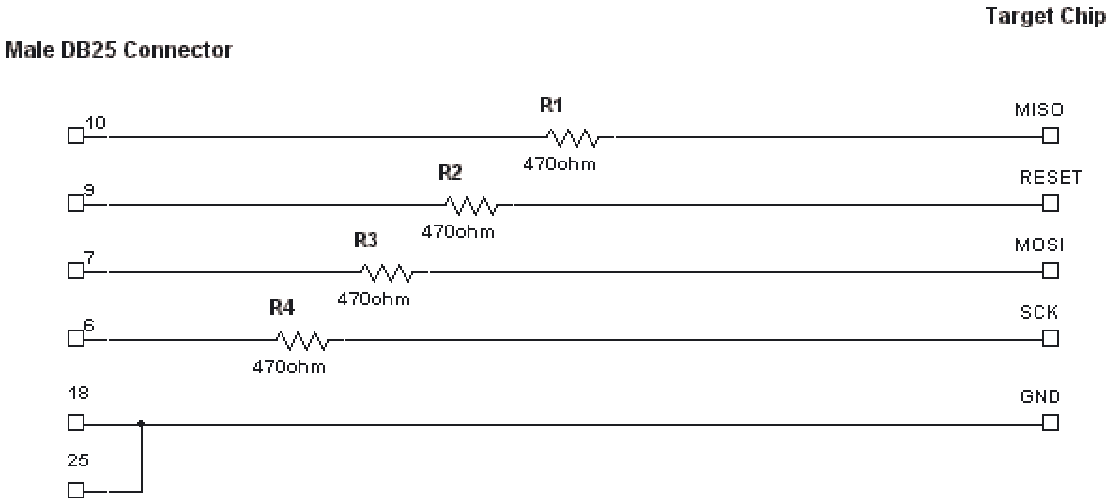


Note: if the pin numbers are too small to read, go to the end of the article, another copy of this diagram is presented with larger pin numbers.

You can build this on the breadboard, or just solder it onto some circuit board. However it will be cheaper to build it on the breadboard, as then you won't have to buy the circuit board. Here is a picture of it built on the breadboard.

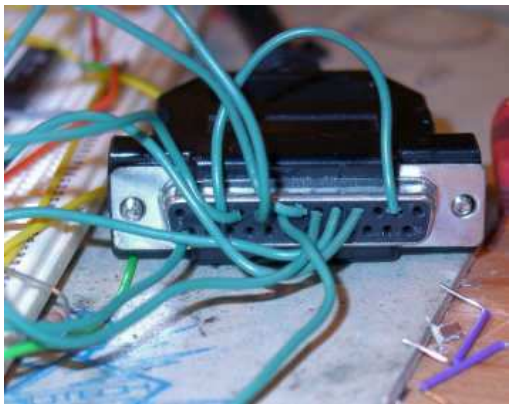


Now if you really don't want to buy the buffer chip, there is still hope. You can normally find these chips on the boards of old computers, search the expansion cards of old computers and you will run across a few of these, although you will have to unsolder the chip. If you don't want to do either of these, you can still do this:



This circuit doesn't offer any protection to your parallel port. However if you really need to you can use it. Be sure to use this circuit on a computer with a removable parallel port card, so that if you do damage something the extent of the damage is limited. This is a good idea even if you use the programmer with the buffer chip, as its possible that you could still mis-wire something.

For both of these circuits you will need a DB25 male connector and a parallel port extension cable, and this might end up being a bit expensive. If this is the case, you will need to find a spare printer cable or something with a DB25 male connector on the end. Cut off the one end (the non DB25 male end), and strip the wires. Then find the wires that connect to the pins the programmer uses with an ohmmeter. If even that is a bit too much to bother with, there is one extra cheap way to go. Though it needs a

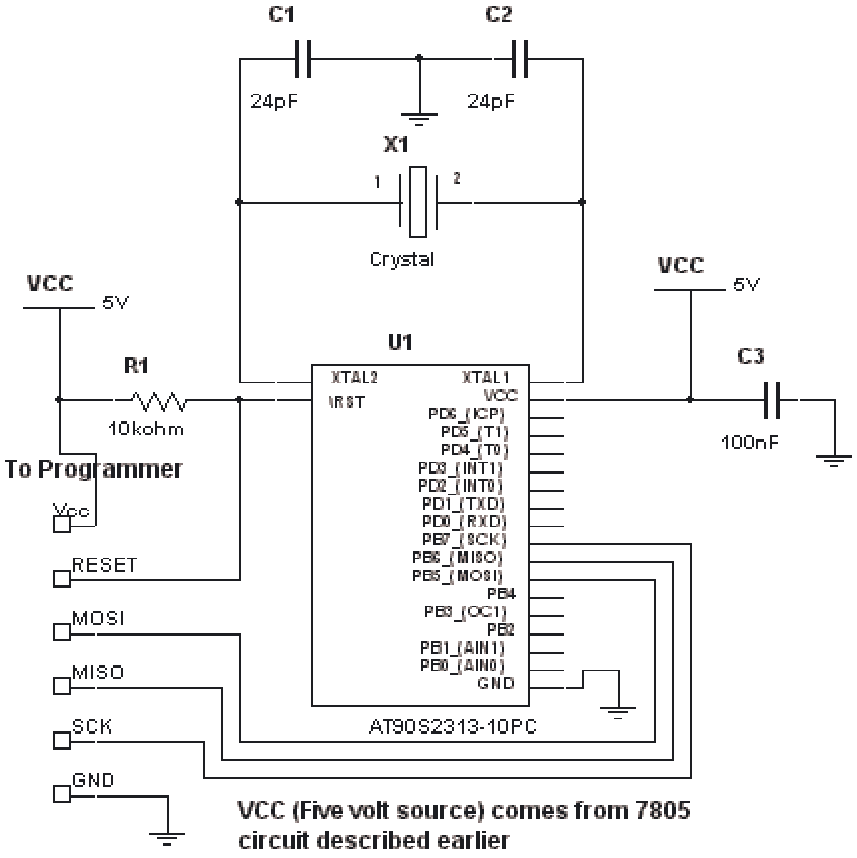


parallel port extension cable, and you might have one of these lying around. All you do is take the wires and put a little “wave” in them, so they look like the picture, and then just push them into the female end of the parallel port.

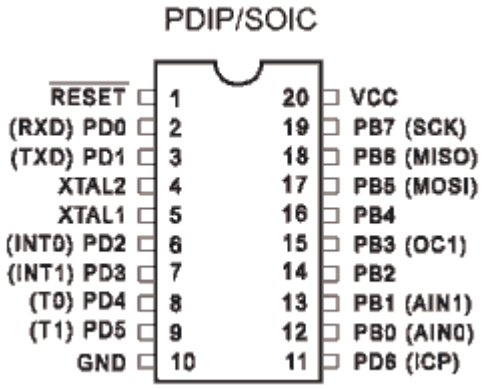
Once you’ve built your programmer of choice its time to test it out. The next step is building a circuit and downloading a few files to the AVR. Note that the programmer with the 74LS244 is compatible with the STK200 dongle, so if you use another programming software and it asks what type it is, it is an STK200. However the other schematic with just a few resistors is not really STK200 compliant. It is a bit, but not entirely. PonyProg works with it, but I’m not sure about other software. If you try it with another programming software, say it is an STK200 as well.

Building a Simple Circuit

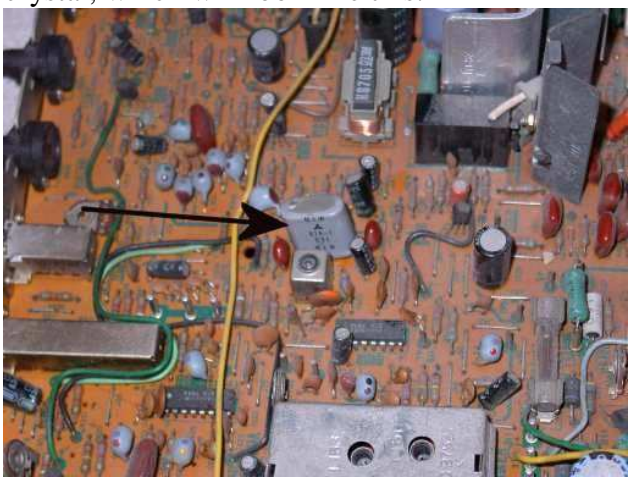
The next step is to build a circuit with your AVR microcontroller. Here is a schematic diagram of the circuit:



This schematic doesn't have pin numbers on the AVR, so here is the pinout of the chip:

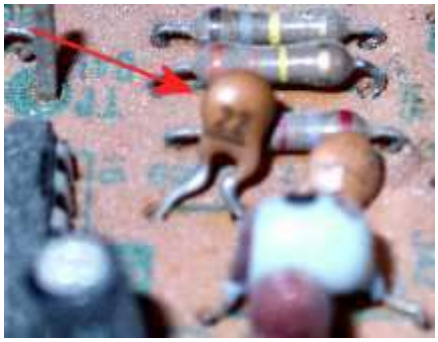


The crystal speed isn't too important, try to find something in the range of 1 MHz to the maximum speed of your chip. For example if you have an AT90S2313-10PC then it is 10 MHz chip, or if you have an AT90S8515-8PC it is an 8 MHz chip. Note that the PC part is just the temperature rating, so an 8PC and an 8PI are both 8 MHz chips. If you are using a breadboard, don't get too fast a crystal. A good source of crystals is old TV boards, just be VERY careful around old TV's as they have high voltages inside them that exist for a long time after power off. If you don't know about discharging the capacitors in them and the CRT, then you might not want to go this way. However just go to your local TV repair shop, and see if they will give you an old board that they have taken out of a TV that broke. They will likely just throw away some of the boards, so they should give you one for free. Look around the board until you find the colourburst crystal, which will look like this:



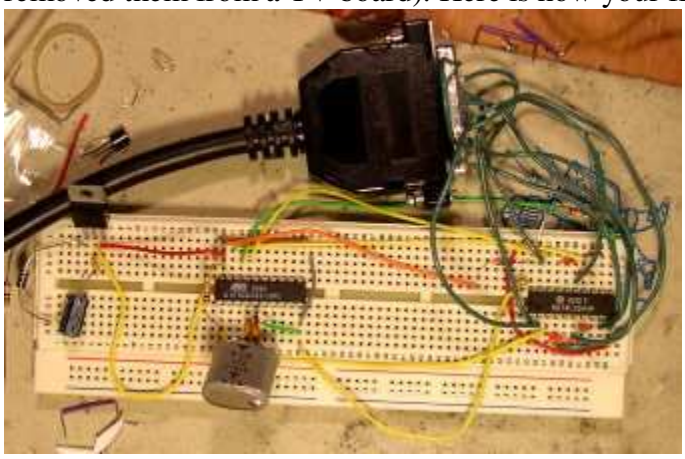
The crystal could look like either of the crystals pictured.

This is a special crystal used in TV's, and has the value of 3.579 MHz. Remove the crystal from the board; as well you will need some capacitors for the crystal. Look around the board for small capacitors marked with a number close to "24" and a line under that number. You will need two of these; they are used to help the crystal oscillate. Since the crystal came off an old TV, we can only really guess at the

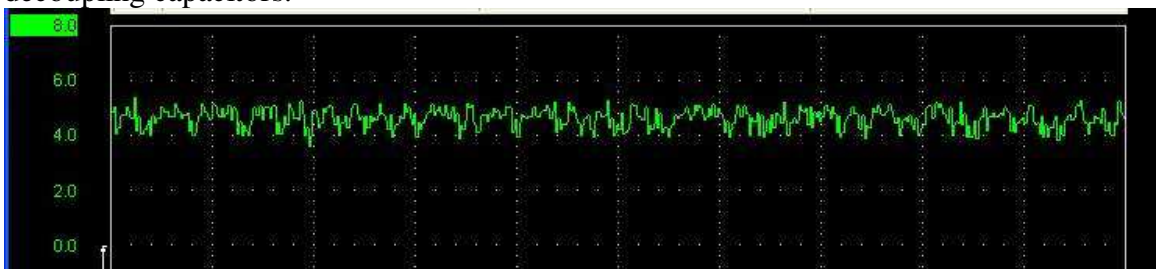


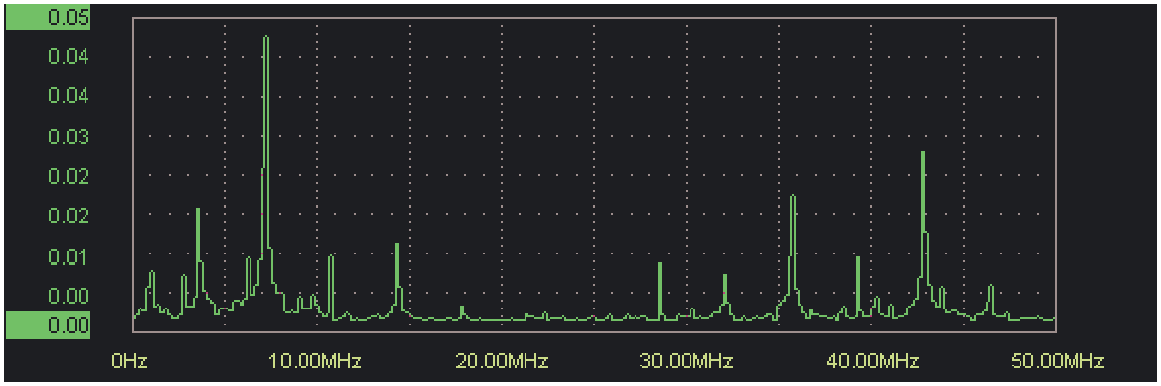
proper value for these crystals. There is ways of calculating the proper value (see the article on this in the AVRFreaks.net academy section). If you are just buying capacitors, get a few values: two 22pF and two 24pF. Also get a few capacitors marked “102”, “103” or “104” from the board, or just buy a few 0.001uF, 0.01uF or 0.1uF capacitors. See the end of the guide for an easy (and fun) way of removing large amounts of these parts… it involves a highly flammable gas.

Some of the leads might be too short to put in the breadboard, so make sure to add extensions on the leads of components with short leads (this will be most of them if you removed them from a TV board). Here is how your final board will look:



Don’t skip those small capacitors on the power leads of the AVR. Let me offer you some graphical proof. First, here is what the power looks like going into the AVR without any decoupling capacitors:

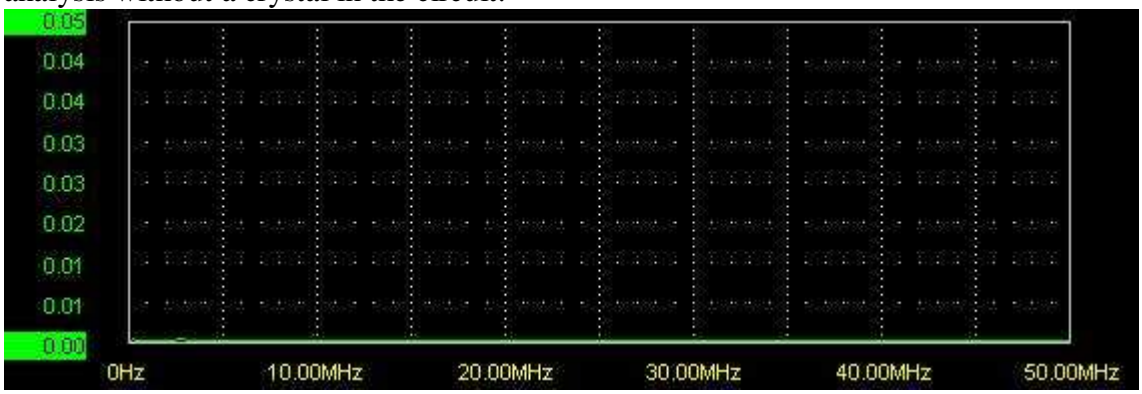




The top picture is an oscilloscope trace. That line that is fluxtuating around five volts should be a straight line right at five volts. The bottom picture is a spectrum analysis of that waveform. The spectrum analysis might fool you slightly, because the voltage levels

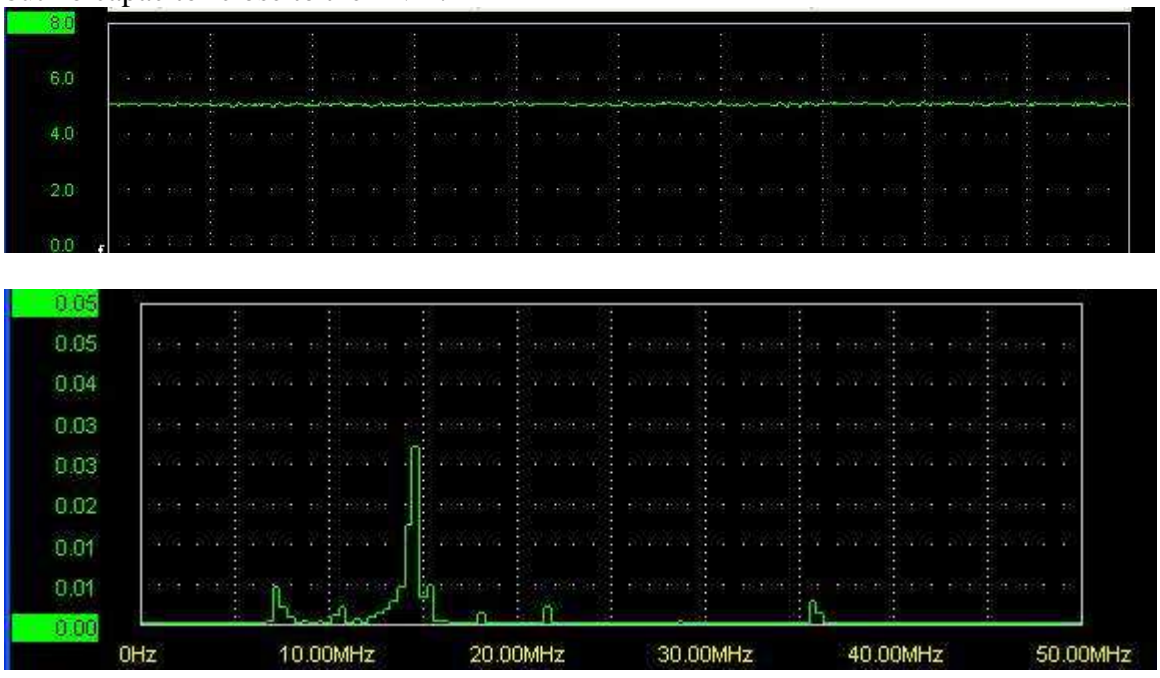
aren't that high on it. However the spectrum analysis spans a huge bandwidth (0 MHz to 50 MHz), and a lot of those "spikes" go very high (several hundred milivolts), but they are very thin as they span only a specific frequency range. Many start to appear so thin near the top that they don't show up on the screen. However it is useful to show you the huge amount of different frequencies.

The crystal oscillating generates most of this interference. Lets look at a spectrum analysis without a crystal in the circuit:

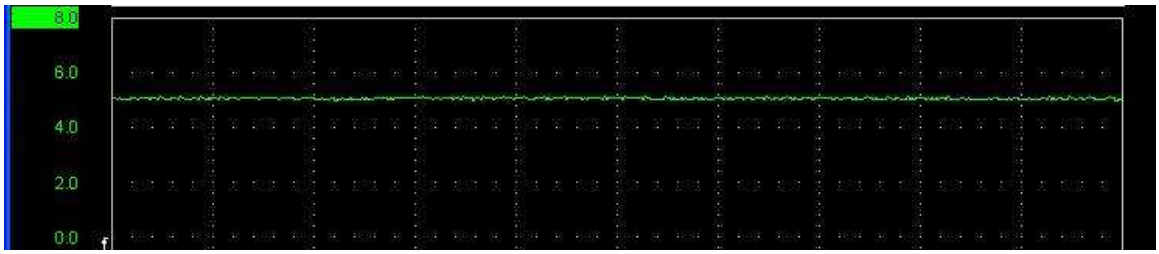


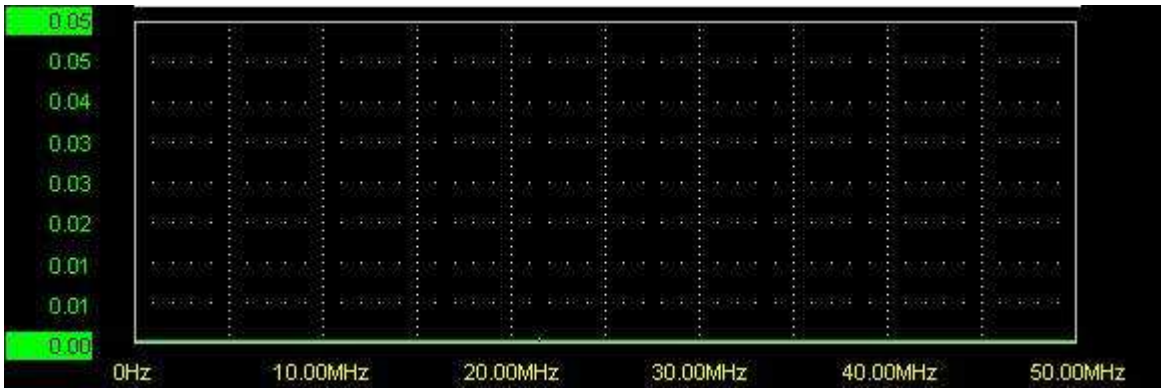
Most of the interference is gone! However again there are a lot of spikes that don't show up, as an oscilloscope view still shows a lot of noise on the line.

Now lets look at the voltage going into the AVR with just the capacitor on the LM7805, but no capacitor close to the AVR:



As you can see, just a capacitor on the output of the LM7805 won't cut it. You need a capacitor as close as you can get to the power pins of the AVR. Lets look at the voltage going into the AVR with a capacitor right on the power lines:





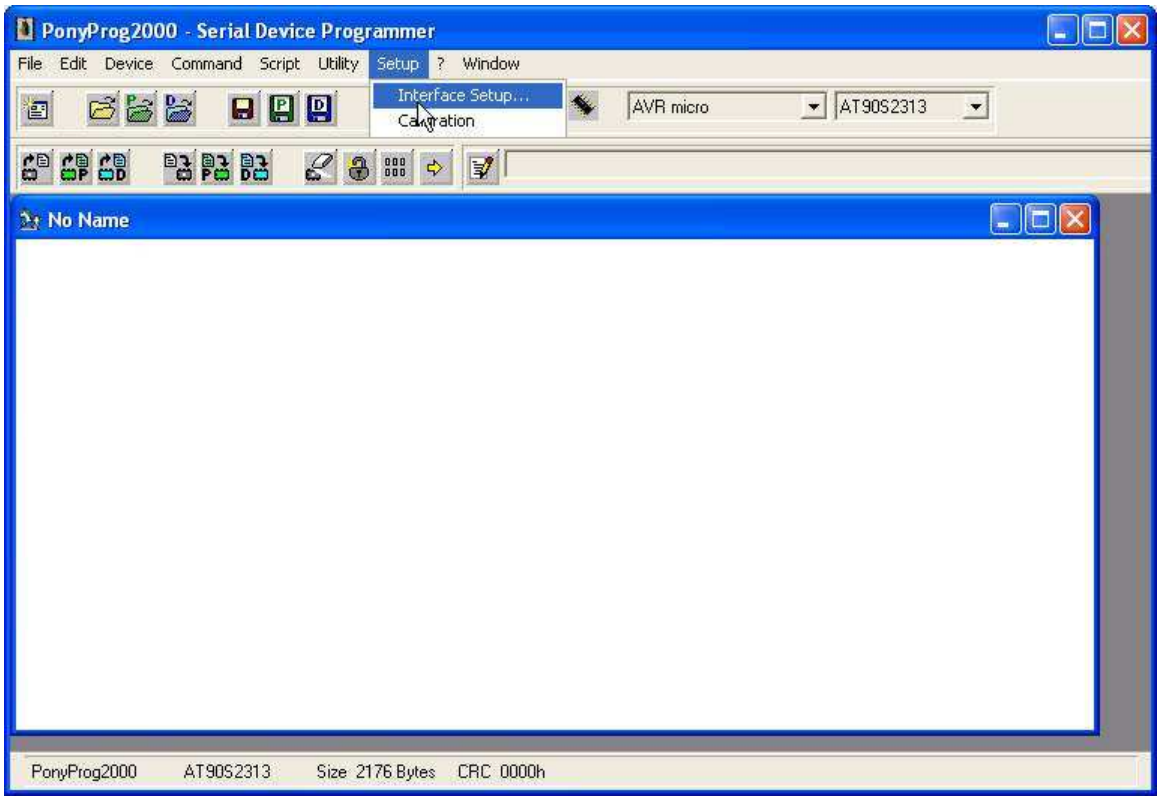
This should make it obvious to you: there is no substitute for a decoupling capacitor close to the AVR. You can still see some slight activity on the line, but you will have to live with that when you just use a 0.1uF or 0.01uF capacitor. If you wanted to completely eliminate the problem, you could calculate the proper value of a capacitor. But if you're getting your capacitors off an old TV, you really won't have a lot of choice.

Before powering up the circuit, check and double check your wiring. The diodes are in the circuit to prevent you from putting power backwards, but check the orientation of the diodes. As well when you first apply power keep a finger on the AVR to make sure it doesn't warm up. If all goes well, it won't. If something goes wrong and it starts to get hot, you have trouble. Take power off as soon as it starts to get warm, and check your wiring. If your lucky the chip will still be alive, so don't give up hope. I have personally had an AVR chip that released smoke… and still works.

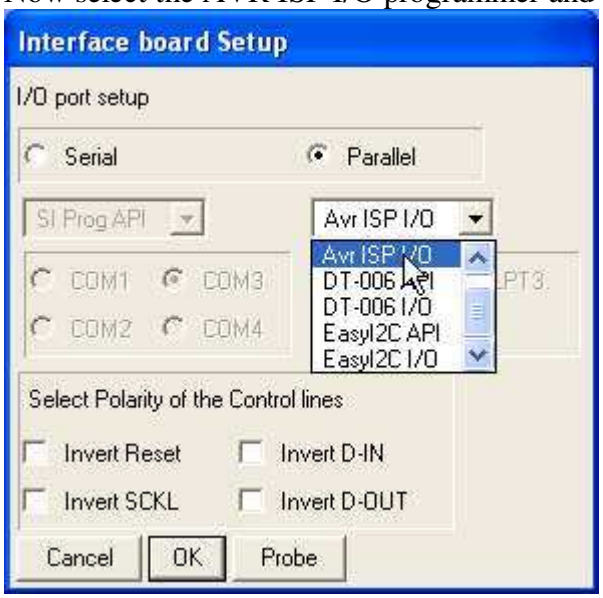
Finally connect up your programming cable to the proper lines. This is shown in the schematic diagram at the beginning of this section. Your programming cable is either the 74LS244 circuit or just the parallel port cable with four resistors.

Programming Software

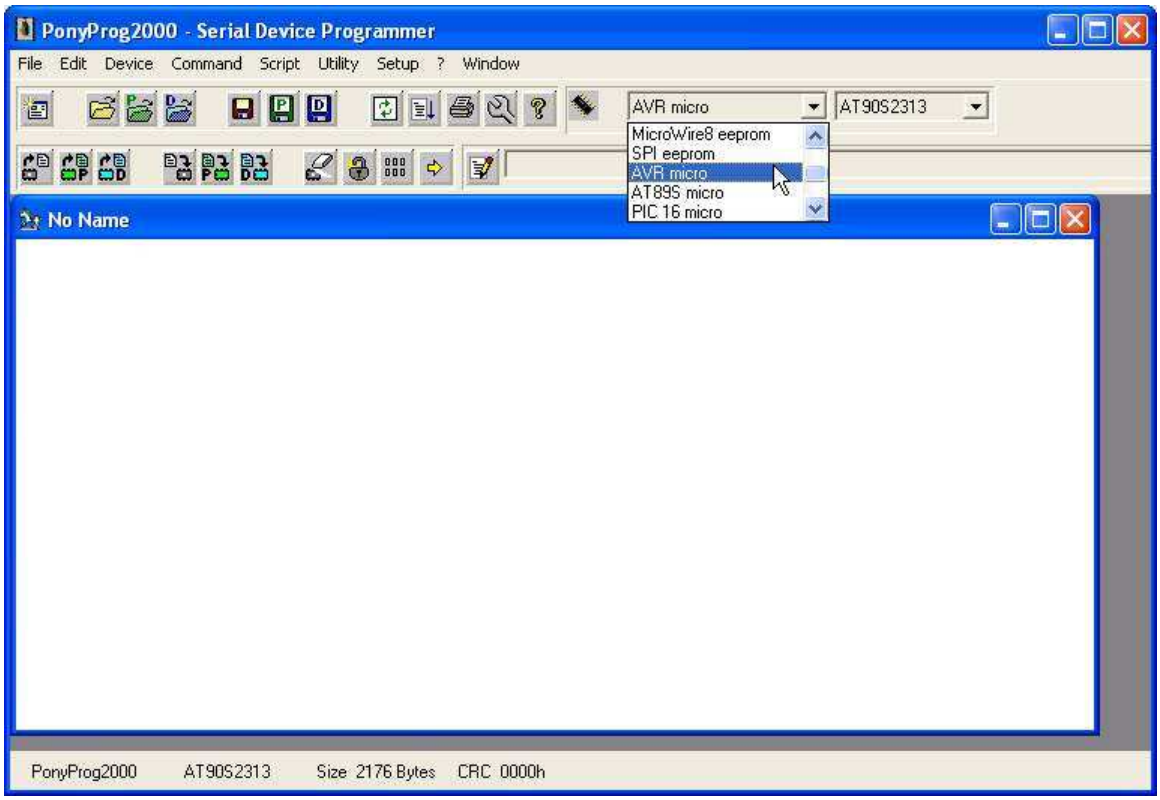
The programming software used here is PonyProg, available for download at their website which is www.lancos.com/ppwin95.html and download the latest version (right now I'm using v2.05a BETA). PonyProg is available for Linux and Windows, so no matter what operating system you're using you're covered (unless it's a Mac that is…). Open up the software and let's test out the circuit you built. First wait for the horse sound and click OK. Then go to the Setup menu and click Interface Setup.



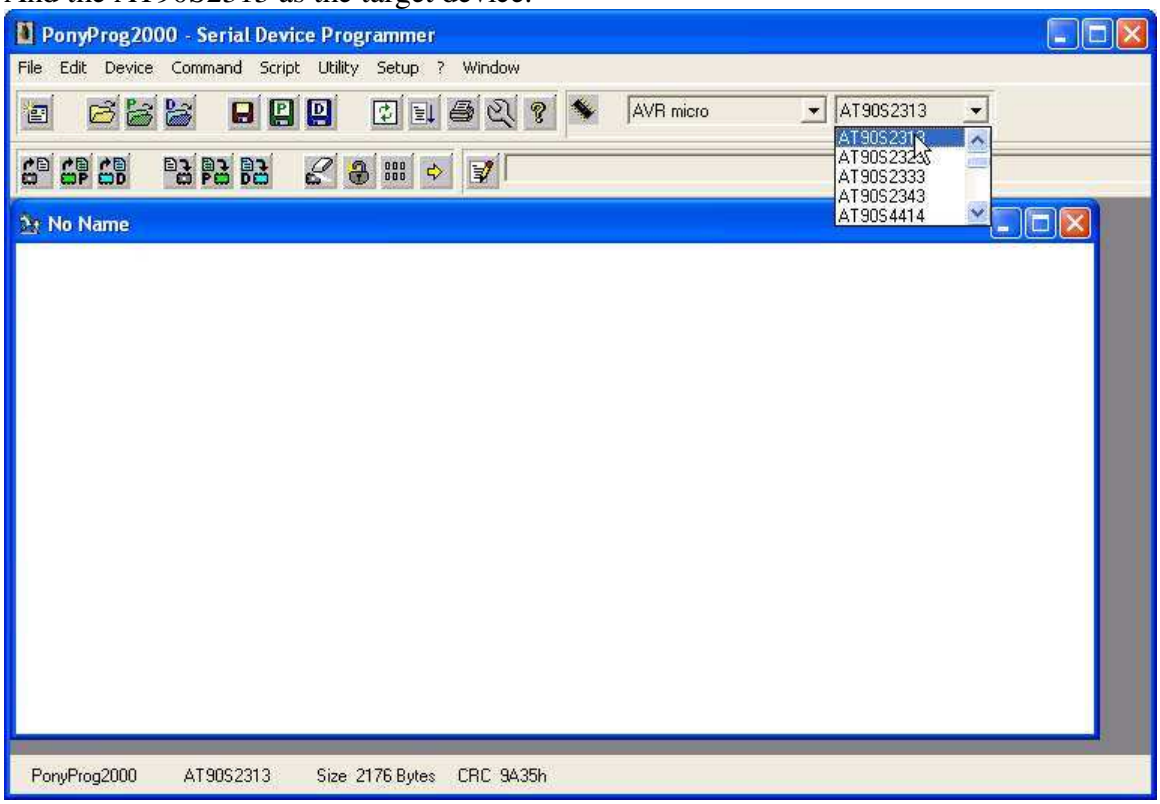
Now select the AVR ISP I/O programmer and click OK.



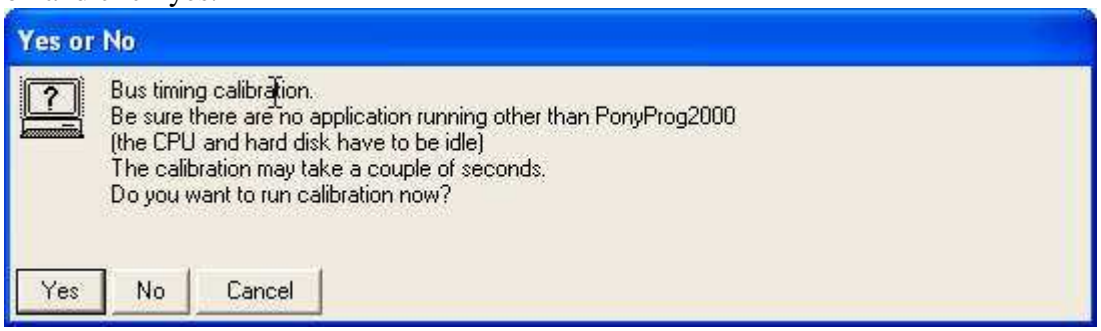
Next select the AVR Micro as the target family.



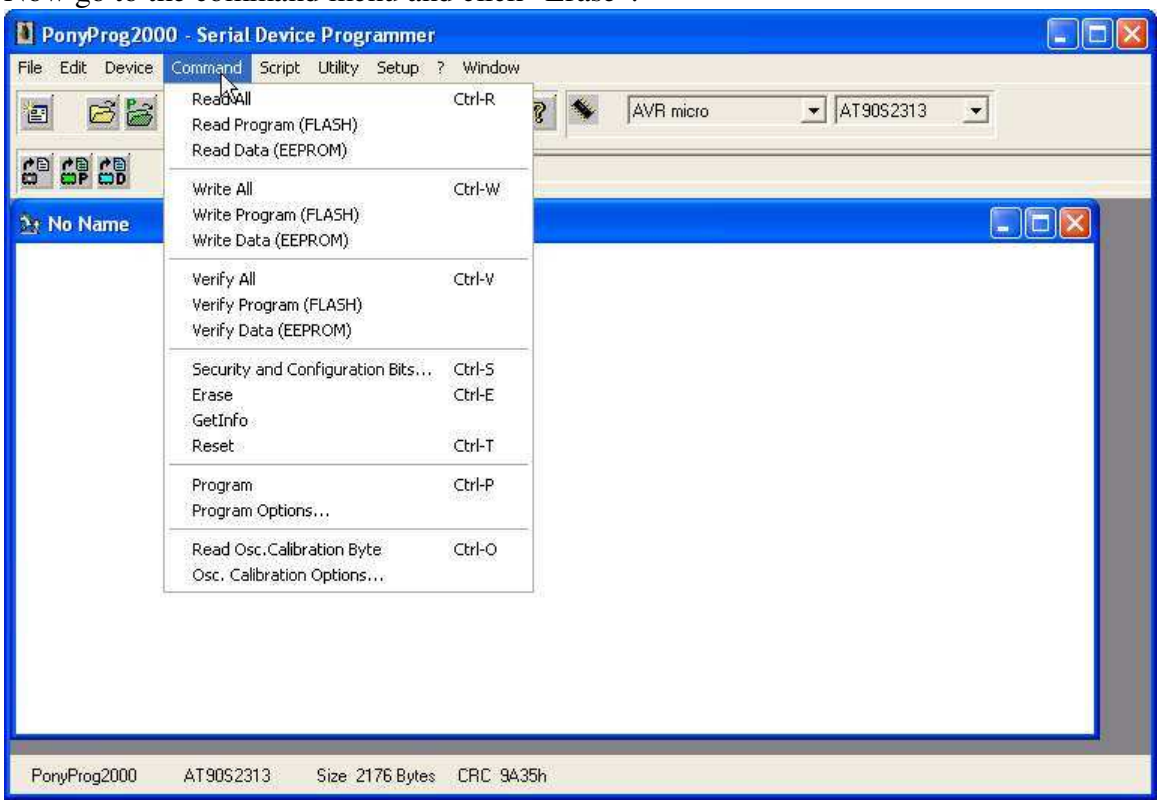
And the AT90S2313 as the target device.



Now go back to the Setup menu and go to “Calibration”. Make sure nothing else is going on and click yes.



Now comes the moment of triumph or defeat. Quickly check your wiring over and make sure that power is applied. Make sure your programmer is connected to the parallel port. Now go to the command menu and click “Erase”.



If all goes well, you will get a message saying the erase was successful. If it doesn’t work, you’ll get an error saying it didn’t work, and give you the option to retry. Check over your wiring again. I made about three wiring mistakes when I did this quickly. One of them was forgetting to connect the GND of the parallel port (pin 18 to 25) to the GND of the circuit. Also check the connections to the parallel port; make sure that it isn’t a bad connection (its been said that 80% of time a problem with an electronics circuit is in the connections).

If your wiring is perfect, it may be other things. Your parallel port may be set to the wrong type in BIOS, so try it on a different machine. If it works on that machine, it's a good chance that your BIOS is set up wrong. You want another parallel port mode; you can change that in the BIOS setup when you reboot your machine.

Also it's possible that your crystal isn't oscillating properly. The crystal can be a finicky thing, and is hard to get working at times. You can try a few things. First, change the value of the capacitor on either side of it. So if you are using two 22pF capacitor right now, try two 24pF or two 18pF capacitors instead. Also you can try touching the crystal lead with your finger and trying to erase the chip again. If the chip erases when you touch the crystal with your finger, than the problem is that the capacitors need to be a different value.

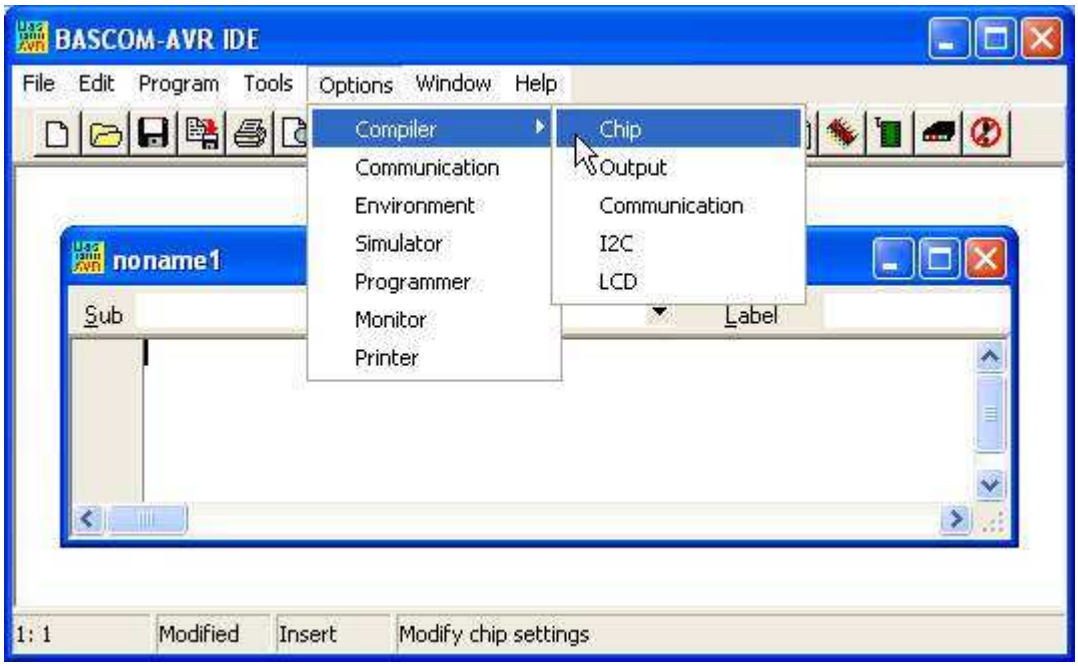
Also if you are using the programmer with the 74LS244 in it its possible the chip is fried, so try making the other programmer that just uses resistors.

Important note: when you finish programming the chip, you have to disconnect the programmer from it if you intend to use the pins the programmer is connected to. If you are using the programmer with just the four resistors, you may have to disconnect the programmer from the chip no matter what. Also, if you use the pins that the programmer connects to for inputs or outputs in your circuit, you may have to disconnect the circuit from those pins, and connect the programmer to the pins, then disconnect the programmer, and reconnect your original circuit. However this just depends on your circuit, it may not be needed.

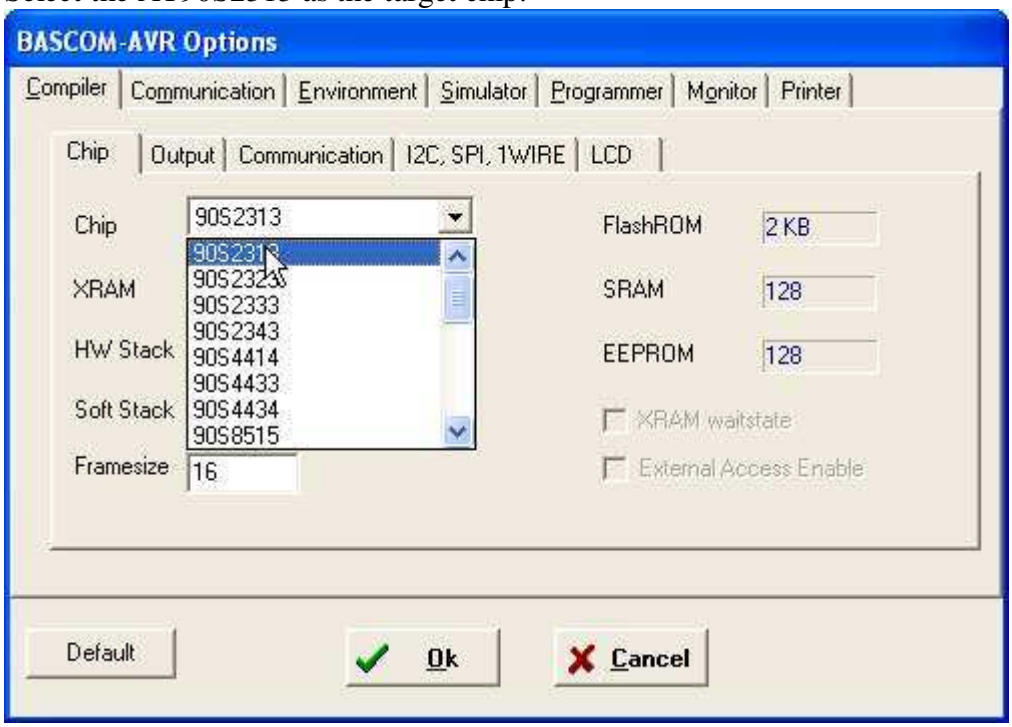
The Compiler

The compiler is the most important part. You can skip this section if you want though, because it's very general. There are lots of different compilers around, and lots of different languages. For this example I'll just use the BASCOM-AVR demo compiler with the circuit we made earlier.

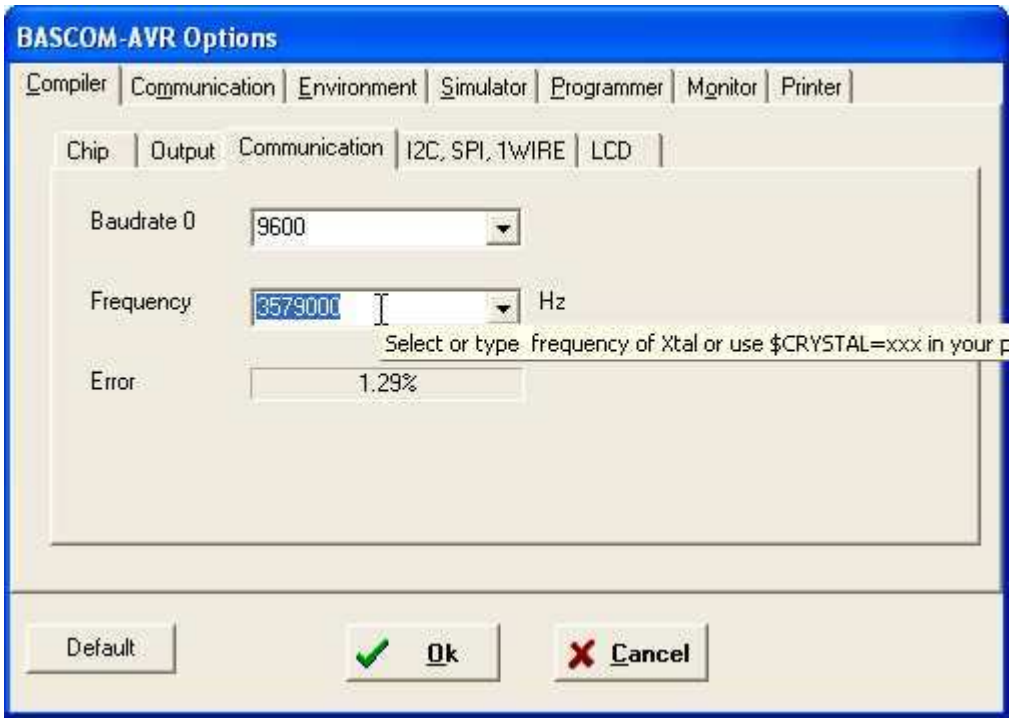
Install the BASCOM-AVR demo version, available from www.mcselec.com and opened it up. Start a new file, and lets setup BASCOM-AVR. Go to the Options-Compiler-Chip menu:



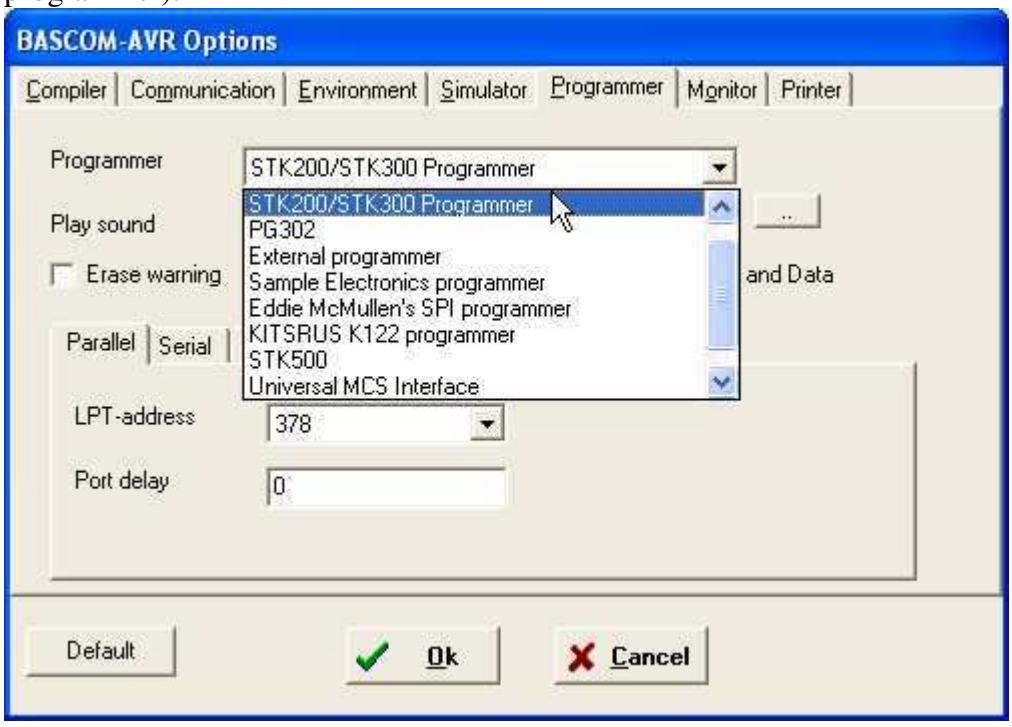
Select the AT90S2313 as the target chip:



Then goto the communications tab, and enter your crystal frequency on Hz. If you use a colourburst crystal, it is the one shown in the screenshot.



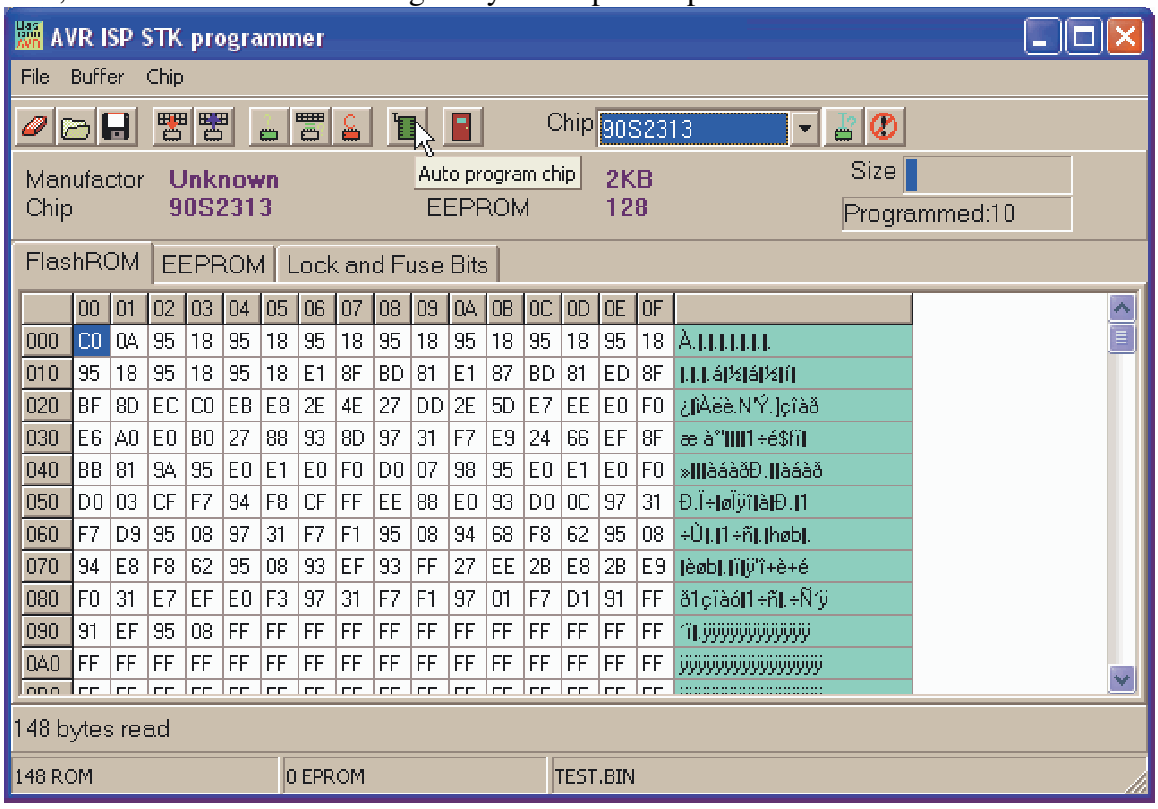
Finally if you want to use the integrated programmer in BASCOM-AVR, select the type. For me "STK200/STK300 Programmer" worked (tested with the four resistor programmer).



If you plan on using BASCOM-AVR, read the help file. I cannot stress this enough. The help file contains a lot of important information on BASCOM-AVR, as well as all the commands. Now type this in your BASCOM-AVR window:

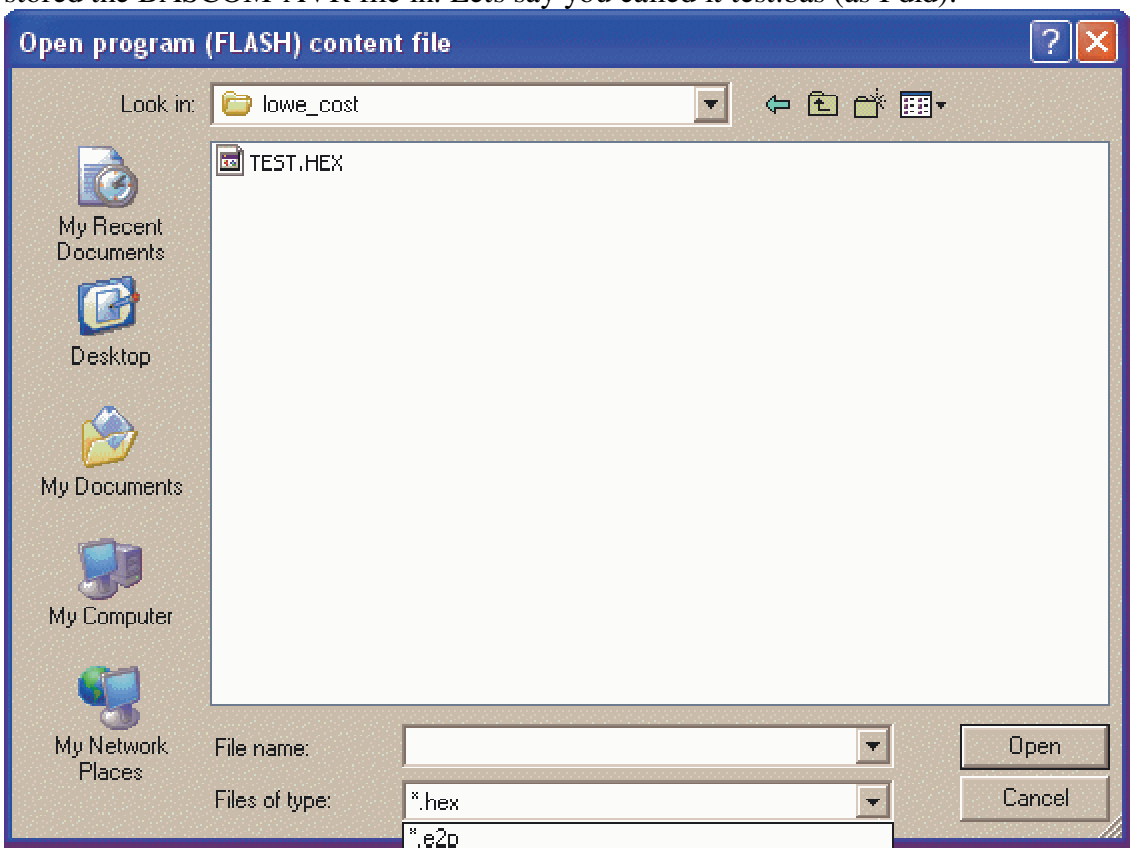
```
Config Portd = Output

Main:

Portd.5 = 1
Wait 1
Portd.5 = 0
Wait 1

Goto Main

End
```

And click on the “compile” button (little black chip).
Hopefully no errors are generated. Now hook up your programmer to your parallel port. Power on your circuit, and click the program button (little green ZIF socket). If all goes well, BASCOM-AVR will recognize your chip and open a window.

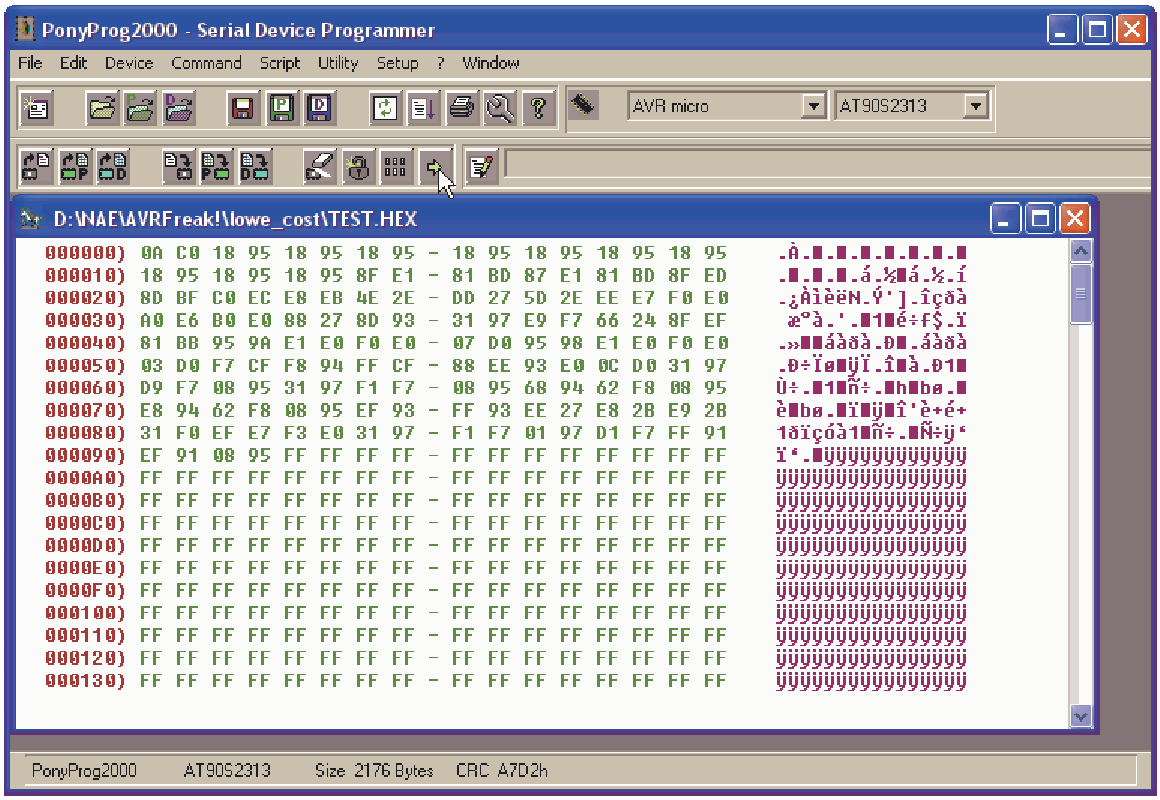


Then click on the auto-program button (little green ZIF socket). The chip should program and verify, if it doesn’t then check the wiring of your programmer. Also open up PonyProg and try to erase the chip using the procedure shown earlier. If PonyProg is successful at detecting the chip, BASCOM-AVR isn’t set up properly or it’s a problem with Windows.

If you need to use PonyProg, this isn't a problem. Simply open up PonyProg, and go to the "File---Open Program (FLASH) file…" menu. Then browse to the directory you stored the BASCOM-AVR file in. Lets say you called it test.bas (as I did):



Select the file type as *.HEX and click on the test.hex file. Click the "Launch Program Cycle Button" (the little yellow arrow).



This programs the *.HEX file that BASCOM-AVR or any other compiler generates into the chip. You may also have to select the chip type, as described earlier.

Now get a small LED and a 470-ohm resistor. Connect the negative of the LED to pin 9 on the AT90S2313. The positive of the LED goes to one side of a resistor, and the other side of the resistor (resistor value: around 470 ohms) goes to +5 volts. The LED should be flashing, if it isn't try reversing the LED, it might be in backwards. Also try disconnecting your programming cable from the chip. More specifically, disconnect the wire driving the reset of the AVR. The computer may accidentally be driving the AVR's reset pin the wrong way. This is more of a problem with the programmer with four resistors. In BASCOM-AVR the programmer with the four resistors did this, where after programming you had to disconnect the wire that goes from the parallel port to the reset pin (the reset pin is pin 1 on the AT90S2313).

Where to Go Next

Now that you know how to program the AVR, you need to know how it works specifically. The best thing to do now is to read. Read the datasheet for the AVR you are using, or at least read important parts of it. Read the FAQ's at AVRFreaks.net, probably many questions you have will be answered there. You will get a lot of knowledge by reading these information sources, as this article is really designed as a bare-bones overview of getting into AVR microcontrollers.

If you know C, then use a C compiler for the chips. An article on AVRFreaks.net will help you chose the right C compiler for you. If you want to use BASIC, then download the BASCOM-AVR compiler. BASCOM-AVR is a very easy language to learn, and if you have no idea where to go next then it might be a good choice for you. Also check out the "Tools" section of AVRFreaks.net, as it lists a lot of compilers. As well you can program in assembly language, but if you have no experience with microcontrollers this may be the hardest route.

Appendix A: How to Extract Components From Old Electronics

Note: This involves some danger, as such: Atmel, AVRFreaks or I will not be held responsible if any damage occurs to anything (including any person) through the use of this guide.

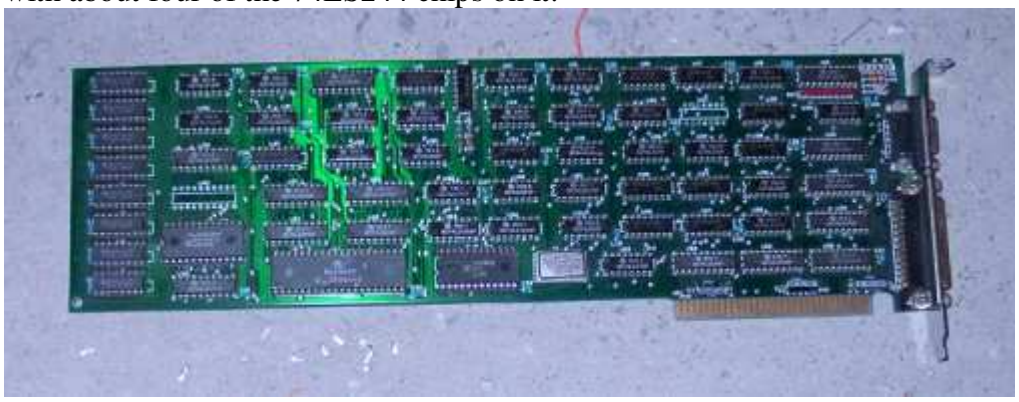
An old TV board (with the large capacitor properly discharged) will be a good source for capacitors, resistors and colourburst crystals. But how do you remove all those components? There is a lot of unsoldering to do, and you don't want to damage the components. You could carefully unsolder every component, using a soldering iron and solder wick. Or, if you're like me (that's a scary thought), you could just use a blowtorch. First step; get all these materials:

- Blow torch – I use a "Bernz-O-Matic TS4000". It is easy to start, and will work with propane as well as brazing fuel (brazing fuel is probably too hot for this application though, it goes to 1982 °C)
- Safety Glasses – use them
- Screwdrivers – used to pry components and hit the board
- The victim – whatever board you will be removing components from
- A fire extinguisher or hose – just in case something goes wrong
- A large outdoor driveway – you need somewhere to do this, a driveway works well
- A friend – not required, but it helps

Stand the board up on its side, or use bricks to help keep it upright. For the first example, we will remove the crystal. Get your friend to slip the screwdriver under the crystal, not too hard though. It just needs a light pressure so that when the solder melts the crystal will pop out. Then turn on the torch. Aim the torch in the general direction of where the crystal is. You want the torch to quickly melt the solder. So keep the torch close enough that a lot of heat is transferred to the board, but if it is too close the board will burn before the solder has time to melt. With my torch it only took about two seconds, your results will vary though. If you don't have a friend

there or want to get a lot of components off the board, there is other ways to do it. Mount the board on some bricks, with the component side facing down. The idea is that when the solder melts, gravity will pull the components down. You may have to tap the board with a screw driver to knock them lose though, so spend some time testing this!

Another source of good digital components is old computers. Here is a picture of a board with about four of the 74LS244 chips on it:



To use the blowtorch method of removing IC's, you need a friend. Prop the board up or hold the board with fire proof gloves, preferably you're holding the end as far away from where you will be heating as possible. Then get your friend to slip a small screwdriver under one side of the IC. Heat the board under the IC, and the solder will melt. Then the IC will come partially out. Your friend will have to then put the screwdriver under the other side of the IC, and pull out the rest of the pins that didn't come out. Remember that the IC can be sensitive to heat, so don't keep the blowtorch on for too long. But removing the IC is a lot messier than removing the small components. When I did this, the board

partially caught on fire and the copper traces under the IC partially melted. However, the IC was still fine.

This isn't an exact method of removing parts, so experiment a bit! You really just have to get a feel for your torch.

Appendix B: Reprint of the Programmer Diagram