Notes on Repairing a STK500

By: Colin O'Flynn

This guide is supposed to give you an idea of how the STK500 works, and hopefully how to fix it.

It started when I ended up breaking my STK500, and needed to fix it. It took a reasonable amount of work, so to spare anyone else the trouble of going through that I have made this guide. However it details how I did do some of this, as you may need to dig a bit deeper, in which case you can continue where I finished.

Also note that this guide is more concerned with problems where the STK500 isn't detected at all, not that maybe certain features don't work. However you should still find this useful if your STK500 is detected fine, but probably not as useful as you are hoping for. This guide is NOT a step-by-step guide to fixing your STK500, as there are too many possible problems. If you follow this guide from start to end you will not find out how exactly to get your STK500 working, instead you should read it through once to get an idea of what recommendations I make.

How the STK500 Works

The basic block diagram is shown in figure 1.

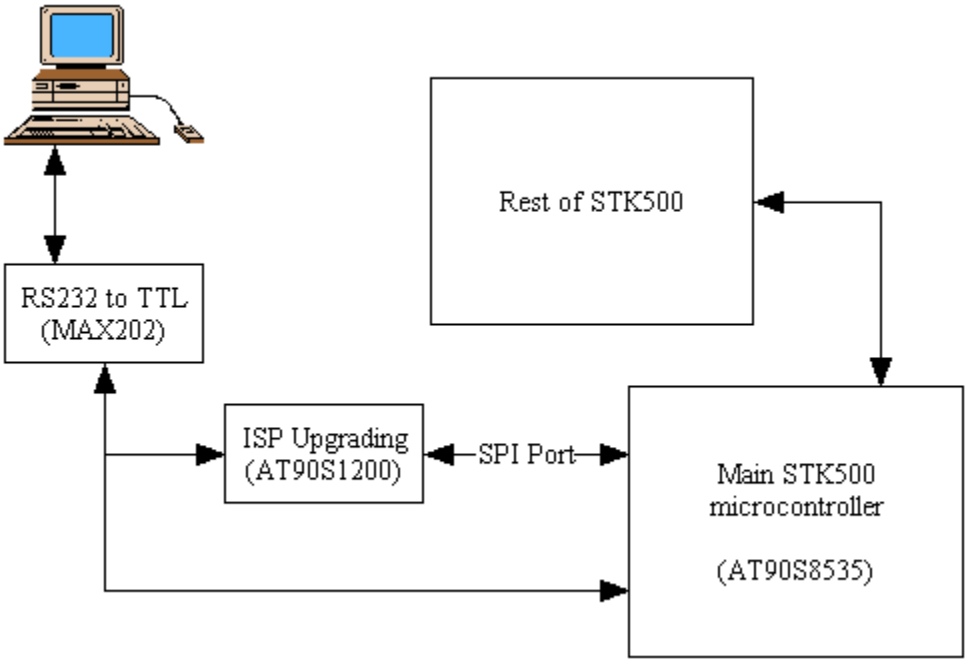

Figure 1 - STK500 command interface diagram

So the three areas of trouble are the MAX202, the AT90S1200, and the AT90S8535.

The way the STK500 works is by default the AT90S8535 is in charge, communicating on the "RS232 CTRL" port on the STK500. If you want to upgrade, you have to hold down the "PROGRAM" button while powering up the STK500. This buttons is connected to the AT90S1200, and when it detects the button being pushed down it holds the AT90S8535 in reset and listens on the "RS232 CTRL" port for a command. The code in the 1200 is essentially the code in AVR910, the appnote on building an in-system programmer. It is set to communicate at 19200 baud rate – I found this out using a great tool called PortMon from http://www.sysinternals.com which will tell you what an application is doing on the serial port.

Eventually it would seem you could eliminate the AT90S1200 by using the ability of the ATMega8535 to have a bootloader in it, as that is all the AT90S1200 does essentially.

Status Light Test

The status light is a very good indicator of what is wrong. If your status light does not come on at all (as was the case for me), then something lis likely wrong with the AT90S8535. It is possible that a glitch in the AT90S1200 is holding the AT90S8535 in reset (as it controls its reset line), but a quick logic probe check will figure that out. See pin 16 on the AT90S1200, that should be at a logic high level.

Also be sure to check the +5 volt regulator section, measure the voltage between pin 20 and 10 of the AT90S1200; it should be around 5.00 volts.

If your status light comes on and stays green, probably the MAX202 or serial port section is

to blame.

Testing Serial Communication

So the first step to troubleshooting your STK500 is to see if the AT90S1200 can be reached. There is a few ways to do this. First, connect up your STK500 to the serial port. Next hold down the “PROGRAM” button and power up your STK500. An even better solution is to get a jumper and put it on the two-pin header just above the “PROGRAM” button. This way you don't have to keep holding the button down, as you are probably going to be doing a lot of power-cycling.

The STK500 is now set-up as a AVR910. If you have AVR Studio installed, open it up. Go to the “Tools” menu and select “AVR Prog”. If you get a message “No supported board found! AVRProg version 1.37” than there is another problem somewhere else. If however you get a dialogue, than that is very good news! The AT90S1200 can be communicated with, so it is fine. Likely as well your MAX202 and possibly even AT90S8535 are all working as well.

With this guide you should have gotten a few files as well (see AVRFreaks site). One is called STK500_AT90S8535_old.hex, so select that in the AVRProg dialogue, and try to program it into the chip. If it works, try power-cycling the STK500. Hopefully that was all that was needed!

If this doesn't work (either you get programming errors or are unable to communicate with the STK500 using AVRProg) than its time to do some more searching. First you will want to see if the MAX202 is working OK. You will need a logic probe and a voltmeter for this experiment.

The pin-out of the MAX202 is shown in figure 2, you will probably want the entire data-sheet at http://www.maxim-ic.com.

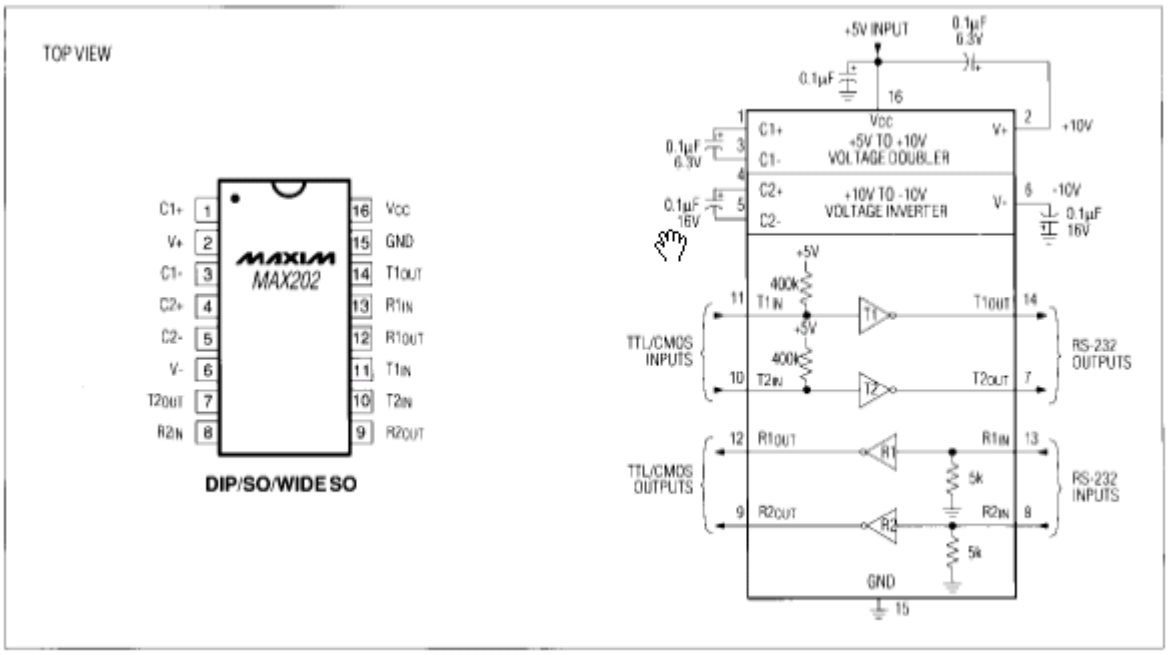

Figure 8. MAX202 Pin Configuration and Typical Operating Circuit

Figure 2 - MAX202 Pin-Out

Measure the voltage first between pin 2 and GND (pin 15). This should be around +10 volts (of course the STK500 should be powered on as well), although it will probably be lower. As long as this is above +5 volts you are OK. Next measure pin 6 and GND, this should be around -10 volts. Again it might be closer -5, but that is still OK. This is the voltage used for RS232 communications, so if the MAX202 is broken it might not be generating this voltage. Or the capacitors connected to it could be leaky – you would have to test them. Or remove them and replace them with other capacitors – in fact the MAX202 should work with as low as 0.1 uF capacitors.

If the voltages are OK, than the MAX202 is likely working OK. Using a logic probe see if pin 14 and 12 generate activity when you try to communicate using AVRProg. Or power up the STK500 normally (no “Program” button held down or jumper mounted) and see if they pulse when you try to communicate using the AVR Studio STK500 tool.

If it looks like the MAX202 is OK, then its possible that there is a short on the RXD or TXD line. They are shared by both the AT90S1200 and AT90S8535, so a problem in either chip could result in a short of this line.

To do this you will want to hold both chips in reset so they should tri-state both lines. They may not though – as in my STK500 the AT90S8535 was damage so severely that it held the TXD line at a logic low state, even during a reset. If you look at the STK500 schematic (which is an essential part of the troubleshooting) you will see the AT90S1200's reset is pin 1 (on the AT90S1200), and the reset of the AT90S8535 is connected to pin 16 of the AT90S1200.

Figure 3 - Connecting all the reset pins to GND

So with the STK500 off, solder a fly-wire from pin 1 to pin 16 to GND, as shown in figure 3.

Also short pins 2 and 3 of the AT90S1200, either with a wire soldered on or just use the tip of a screwdriver. This will connect the RXD to the TXD pins together, which means that everything will be echoed back to you that you send out the serial port.

So now start up a terminal emulator (do NOT use Hyperterminal though, it is very unreliable) such as “Tera Term Pro” or “Bray's Terminal”. Set it to 19200 baud rate (it doesn't actually matter a huge amount), and unplug your STK500 from the serial port. Then send a few random characters – you are trying to see if local echo is on. If local echo is on the keys you type will come back on the display, and you don't want them to. If they do, find the option to turn local echo off.

Now plug the STK500 into your serial port using the “RS232 CTRL” port, power it up, and send some characters again. You should see whatever you type echoed back to you. If you do, then it means that the MAX202 is good, and the AVR chips may be good as well (hard to say for sure yet though).

If you do NOT see the data echoed back it may mean that the AVR chips are damaged and shorting the lines, the MAX202 is damaged, or your serial cable is damaged. If you have another serial cable (should be straight-through Female to Male type) try it to see if it corrects the problem.

Testing the AT90S1200

At this point you will have to figure out if the AT90S1200 is working. Remove all the previous fly-wires, you don't need them anymore.

Now mount the 6-pin header beside the AT90S1200, and get another AVR programmer. You can use a normal parallel port mode programmer, I had an old STK200 dongle that I used. You may have to convert the 6-pin header into the 10-pin header most older dongle's come with, which can be done using the 6-pin cable that comes with the STK200 and a few extra wires (see figure 4). The pin-out for the 6-pin ISP socket is giving in figure 5, this is the same pin-out that is silk-screened on the bottom of the STK500.

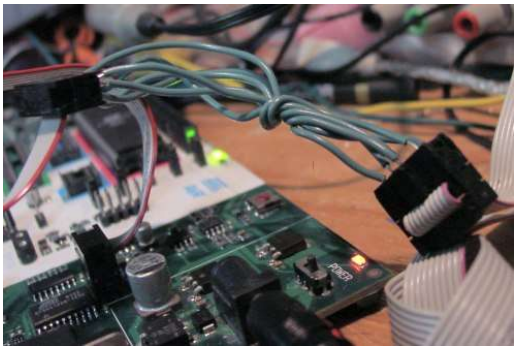

Figure 4 - Converting 6 pin to 10 pin

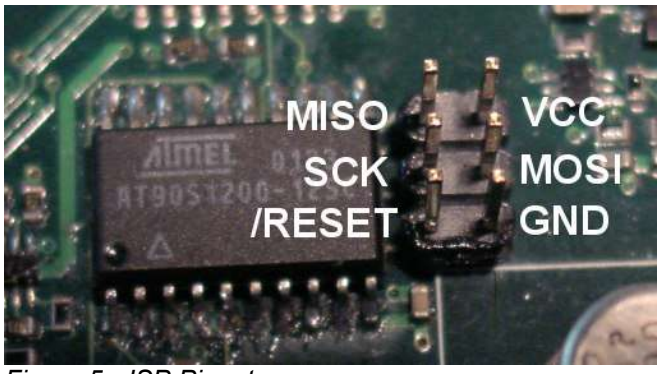

Figure 5 - ISP Pinout

Now run a programmer, and try reading the chips signature bytes. For example PonyProg or avrdude could both be used, to use avrdude type at the command line (avrdude comes with

WinAVR):

```
avrdude -c stk200 -p 1200
```

And avrdude should respond with a message reading the signature bytes. If you do not even get signature bytes, then double-check your parallel port programmer is working OK. If you have another AVR around, use it to check that the paralell port programmers is working.

If you still can't read the signature bytes, use a logic probe on pin 5, which is the XTAL1 pin. They way the clock works in the STK500 is the AT90S8535 is connected to a crystal, and then the XTAL2 output of it connects to the AT90S1200. However if the AT90S8535 is damaged, you will get no oscillation, and neither the AT90S1200 or AT90S8535 will work.

With a logic probe on pin 5 you should see some sort of activity, it probably won't be anything you can visibly see (ie: logic 1, then logic 0, then logic 1) as it will be too fast. But if you have a “pulse” output that might blink, and likely the logic high and logic low LEDs will light up.

If it looks like there is no oscillation, jumper pin 4 of the AT90S1200 to the other side of the crystal – this way the AT90S1200 should make the crystal oscillate (see figure XX). However likely even if the AT90S8535 is damaged, the oscillator section of it will not be, so you shouldn't need to do this. And also it could make the two oscillators fight if in fact they are both still running.

Try programming the part again – if you still get nothing, its also possible that in fact the AT90S8535 is so damaged that it is shorting the XTAL1 or XTAL2 pins, or also the ISP pins (which the AT90S1200 and AT90S8535 share). Or the AT90S1200 could also be damaged – to be sure you will have to remove the AT90S1200 from the circuit.

If you are able to get the AT90S1200 communicating, try loading the code called STK500_AT90S1200.hex into the AT90S1200, and see if you can get AVRProg to communicate with the STK500 as described earlier.

Testing the AT90S8535

The AT90S8535 is the most likely chip to be damaged, as many of its leads connect to the AVR target area, such as the programming, voltage control, and so on. These connections are often handled by the user, and a prime target for an ESD shock, or just plain old miswiring.

If you can get the AT90S1200 working to communicate with AVRProg, then this becomes very easy to test. Go to the “Advanced” tab in AVR Prog, and erase the chip. Then read its signature bytes (the read button), see if they match the AT90S8535 correctly as shown in figure 6.

If you can get this far, then try reprogramming the AT90S8535 with the file STK500_AT90S8535_356.hex, which is the version of the STK500 that works with AVR Studio 3.56. Then verify the file, and power up the STK500 normally and try to use it from AVR Studio.

However if you cannot communicate with the AT90S8535 you should get more information on what is the problem. If when opening AVRProg you get a message that says “No Supported Board Found”, it means you cannot use the application at all. However if instead you get a message that says it is unable to enter programming mode, the AT90S8535 is probably dead.

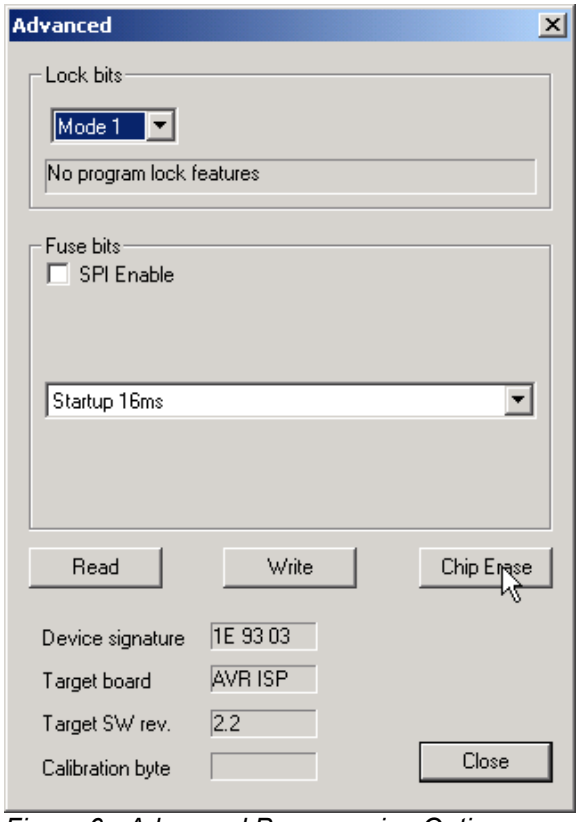

Figure 6 - Advanced Programming Options

Either way, you should use an alternative method to check. This uses the parallel port programmer that was used in the section on testing the AT90S1200, but this time it is used for the AT90S8535.

To do this you have to do a few things:

1. Erase the contents of the AT90S1200 (as described earlier)
2. Carefully lift pin 1 of the AT90S1200 so it is not touching the solder pad anymore
3. Connect a fly-wire from pin 1 of the AT90S1200 (not the pad, the physical pin) to pin 20, which is VCC
4. Connect a fly-wire from the pad of pin 1 on the AT90S1200 to pin 16 on the AT90S1200.

As you can see by looking at the schematic, both the AT90S1200 and AT90S8535 are connected to the same SCK, MISO, and MOSI bus. However the reset connection on the 6-pin header beside the AT90S1200 goes to the reset pin on the AT90S1200. So instead we clear the contents of the AT90S1200 to tri-state its pins, connect its reset pin to a logic high (so it will not respond to programming requests), and patch the reset pin on the 6-pin header to the reset pin on the AT90S8535.

Now try using PonyProg or avrdude to see if the device is present, and erase and program it. Program it with on of the hex files for the AT90S8535, such as STK500_AT90S8535_356.hex

If this doesn't work, there is a good chance you need to replace your AT90S8535. Be sure to check that there is no shorts anywhere causing the problem, as you would hate to replace the AT90S8535 only to find its not the problem! If you are not familiar with SMD soldering, there is a guide on the AVRFreaks.net site on how to solder SMD parts.

However you should seriously consider what your STK500 is worth at this point. If you have not soldered any SMD parts before, there is a chance you will damage the replacement you try to solder in. As well you still have an STK500 that is dead, and also a damaged part, and also probably some supplies you had to buy for SMD soldering.

If you continue though, there is one catch: you cannot buy the AT90S8535 anymore. Instead you have to get the ATMega8535 in TQFP-44 package. However it has a compatibility mode

fuse you can problem to solve this problem. The Mega8535 in compatibility mode is slightly different from an actual AT90S8535, but there are ways around this problem.

After replacing the AT90S8535 with an ATMega8535, you should follow these steps exactly, as using the method described you only get one shot at this. If it doesn't work you can do it again, but you will have to lift the pin of the AT90S1200 as described above to connect the ATMega8535 to the ISP system.

1. Make sure the STK500 is connected normally – remove all fly-wires and hardware changes to the system.
2. Fly-wire pin 1 to pin 16 on the AT90S1200
3. Connect your ISP to the 6-pin header beside the AT90S1200
4. Read the signature bytes, verify you can communicate with the ATMega8535
5. Erase the device (ATMega8535)
6. Program the proper hex file in (this isn't too important if you can communicate with the board using AVRProg), likely the file will be STK500_ATMega8535_hack.hex
7. Set the low fuse byte to 0xFF (Crystal Oscillator, Slowly Rising Power)
8. Set the high fuse byte to 0x49 (CKOPT Programmed, 8535 Compatibly mode)
9. Remove the fly-wire on the AT90S1200
10. Read the device signature, verify you can communicate with the AT90S1200
11. Erase the device (AT90S1200)
12. Program the AT90S1200 with STK500_AT90S1200_Mega8535.hex

This system works because until you program the CKOPT fuse, the oscillator isn't at a high-enough level for the AT90S1200 to pick up on it, so it can't respond to programming commands. But once you program that fuse, you can't only communicate with the ATMega8535 (in theory anyway) anymore as both chips will respond.

At this point you should have a fully functioning STK500! However there is a important note: first of all, the AT90S8535 and ATMega8535 have a different programming mode. This is what the STK500_AT90S1200_Mega8535.hex file does – it is a very serious hack of the AVR910 app-note designed to make the ATMega8535 look like an AT90S8535 to the AVRProg software. In doing this, it makes the programming process very very slow. You can view the source in STK500_AT90S1200_Mega8535.asm to see what it does.

Also AVR Studio may ask you to update the STK500 – try it, and see if it works. However I found that for example AVR Studio 4.07's code broke the STK500, I had to resort to using AVRProg to reload the working code on the ATMega8535.

I then made a hack of the last known code to work – the code that comes with AVR Studio 3.56. This file is called STK500_ATMega8535_hack.hex, and the assembly code is STK500_ATMega8535_hack.asm. The assembly is a result of reassembling the hex file from the AT90S8535 in a STK500, and figuring out where the version information was stored in the program. Then the version information was placed as a declaration at the beginning of the file, and changed so that AVR Studio 4.07 thought my STK500 had the latest firmware, and no update was needed.

In later editions of Studio you may need to change this file, assemble it, and download the hex to your ATMega8535 to work. For example for Studio 4.08, you will likely need to change the “Software Minor” to 0x01. The only way to find out is to try new values until AVR Studio stops complaining about needing to update.

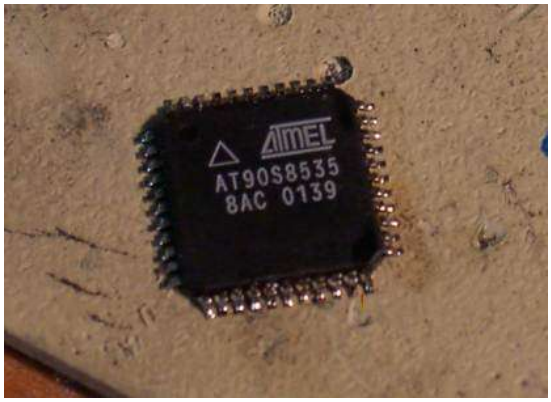

Figure 7 - Suspect: AT90S8535, Charged with failing to respond to ISP commands. Verdict: Guilty

A quick hint – when doing this, first turn your STK500 off. Then open up the STK500 programming dialogue, then after it complains about a missing STK500 turn it on. Then goto the “Advanced” tab and hit “Read Signature”, so it will try again to communicate with the STK500. This way you can see the current version number of the STK500 that is being reported, so you know that the changes are making it into the STK500. If the reported version doesn't reflect the ASM changes you made, check that you have selected iHex as an output type.

File Descriptions

This document should have come with a number of support files, this section tells you what they all mean.

AT90S1200 Code

STK500_AT90S1200.hex
: This is the default code for loading into the AT90S1200 in the STK500, and works with the AT90S8535 chip.

STK500_AT90S1200_Mega8535.hex
: This is a hack of AVR910 designed to work with the Mega8535. It overrides the signature bits so they are reported as a AT90S8535's, and uses page mode programming. But it is very very slow, as it uses page mode in a way it was never supposed to be used (write a page every time) so that it is pretty hard to break.

STK500_AT90S1200_Mega8535.asm
: This is the source code for the above hex file.

AT90S8535 / Mega8535 Code

STK500_AT90S8535_Old.hex
: This is the STK500 code for the AT90S8535 (works on Mega8535) from some old version of AVR Studio. The version is unknown, it is before Studio 3.56 though.

STK500_AT90S8535_356.hex
: This is the STK500 code for the AT90S8535 (works on Mega8535) from AVR Studio 3.56.

STK500_AT90S8535_407.hex

This is the STK500 code for the AT90S8535 (does NOT work on Mega8535) from AVR Studio 4.07.

STK500_ATMega8535_hack.hex

This is a hack of STK500_AT90S8535_356.hex, it is the same code, but the version number has been changed so it is reported as the code that comes with Studio 4.07. I have no idea what repercussions this causes, but it seems to work.

STK500_ATMega8535_hack.asm

This is the source code for the above hex file. To support new versions of Studio if their code doesn't work on the Mega8535, open this up and change the software version number. Then assemble it, and load the new hex file.