

Power Analysis for Cheapskates

By Colin O’Flynn – coflynn@newae.com

Contents

What is Power Analysis	3
Capturing Traces: Basics & Hardware Requirements	4
Typical Capture Environment.....	4
Using Low-Cost Hardware.....	4
Capturing Traces: An Interlude	6
Introduction	6
Getting Familiar.....	6
Capturing Traces: Advanced Topics	13
Considering Noise	13
Differential Probe.....	16
Electromagnetic Probes	21
Building Amplifiers	22
Target Practice - Hardware	25
Simple AVR Target.....	25
Arduino Target	27
Simple XMega Target	27
SmartCard Target: A Normal Reader	30
Smartcard Target – Cheapskate.....	32
Target Practice – Firmware	35
How to Build.....	35
SimpleSerial Firmware	35
Description	35
Programming AVR & XMega	36
SmartCard Firmware.....	36
Description	36
Programming	36

ChipWhisperer-Analyzer	39
The Example Attack	39
Further Attacks	43
CD Resources	43
A Buyers Guide.....	44
Scope.....	44
Magnetic Field Probe	46
Low Noise Amplifier	47
AVR Stuff	47
SmartCard Stuff.....	48
The SASEBO Project	48
Where to Go From Here	49
References	50
Revision History	51

What is Power Analysis

This white paper isn't really going to cover the theory behind Side Channel Analysis (SCA). There are lots of great references, so instead I will just point you to this.

First, you could read the original paper in this field, which even if you aren't too mathematical should make sense [1]. There is also a nice book [2], all of which are discussed in more detail in the 'Where to Go From Here' section of this White Paper.

You may want to check for updates to this White Paper at www.newae.com/blackhat , or otherwise see if there are other interesting things on my website.

This White Paper contains only a few of the many great references in the area of Side Channel Analysis. Spend some time on Google Scholar to start exploring other interesting articles.

Capturing Traces: Basics & Hardware Requirements

Typical Capture Environment

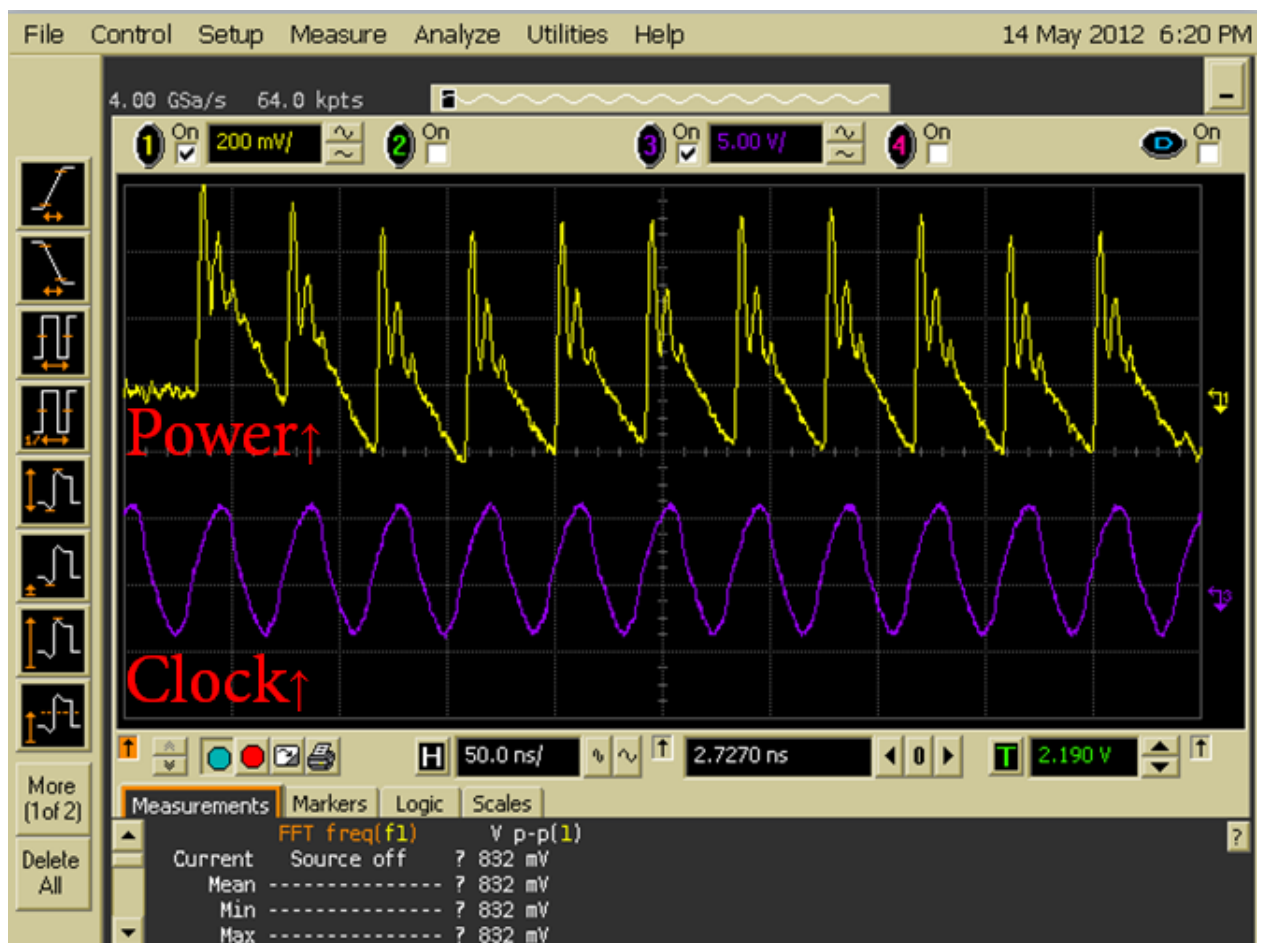
Looking in papers in this field, most people are using normal oscilloscopes. When attacking hardware implementations, this often means one requires a high-speed oscilloscope. When attacking software implementations one can often 'get away' with low-cost oscilloscopes, such as 100 MSPS units.

I have a separate paper which goes into more details around the requirements of this oscilloscope. You can see that paper at [3].

Using Low-Cost Hardware

The phrase "low cost hardware" normally means a cheap oscilloscope. This cheap oscilloscope has a limited sample rate, which will work for attacks at low clock frequencies. But what if you want to attack real implementations running at high speeds?

You can still use low-cost hardware, provided you carefully consider the sample clock. The following figure shows the capture of a power trace from a SASEBO-GII (FPGA) target. In addition the clock which is driving the SASEBO-GII is provided, which is running at 24 MHz:



If we could sample on the clock edges perfectly, we could collect exactly the number of samples needed. Previous work has used this idea for compressing captured traces to only keep one sample per clock cycle. This means you can capture thousands of traces without wasting a lot of hard drive space. But you still need a good oscilloscope to perform the original capture.

I designed the OpenADC, which takes this even further. It uses the device clock to decide when to sample the power consumption. This means that you don't need post-processing to keep the samples of interest. In addition it drastically reduces the sample rate requirements, meaning very low-cost hardware. To give you an example on the SASEBO-GII running at 24 MHz, sampling rates lower than 250 MSPS had been shown to be useless for SCA. With the OpenADC one can sample at 24 MSPS and still achieve a successful attack, as described in my paper at[3].

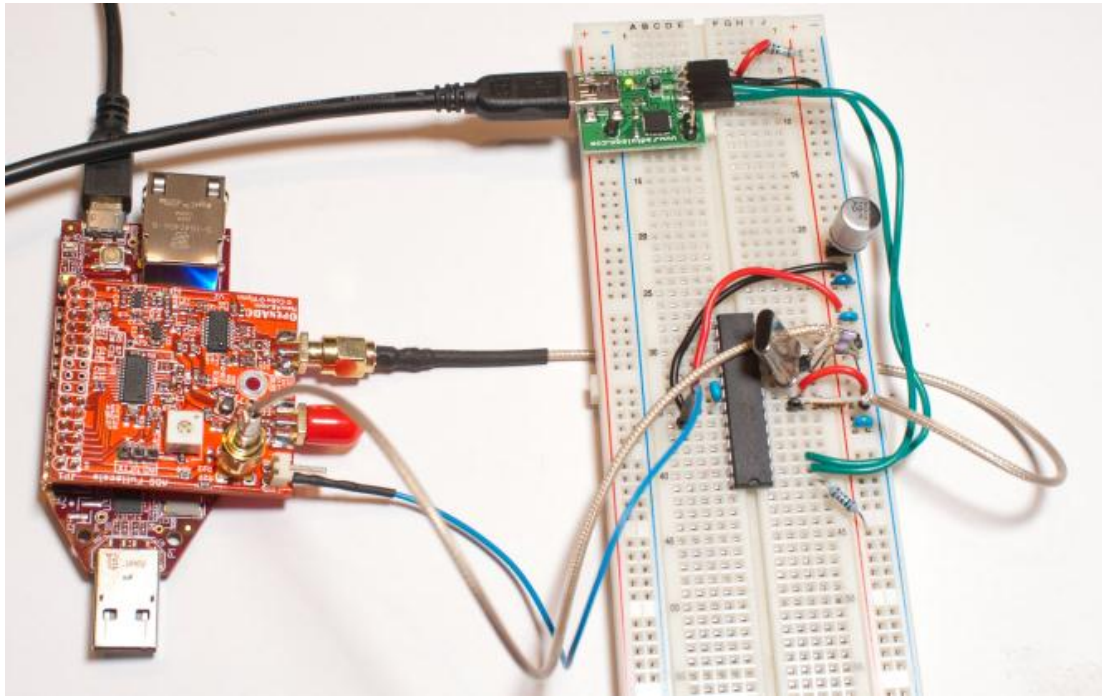
Capturing Traces: An Interlude

Sandwiched in-between the basic & advanced topics, I’m going to give a little bit of a tutorial. If you wish this would let you ‘follow along’ which may help you learn on your own. But even just reading through this may help you understand

Introduction

First, you’ll need to setup/install the capture software. You’ll also need a target.

Your target will probably be an AVR microcontroller, which means your bench looks like this:



Note the following:

1. The AVR is powered from 3.3V, which is the same voltage level the FPGA I’m using with the OpenADC is running at. The clock & trigger lines of the FPGA are normally **not** 5V tolerant. If using a 5V Arduino you need to convert these voltages down, see details later.
2. I’m passing the clock from the target into the external clock input on the OpenADC.
3. The trigger line is connected to the ‘POS’ trigger pin on the OpenADC.
4. I’m measuring the voltage across a 75-ohm resistor in the ground pin of the AVR.

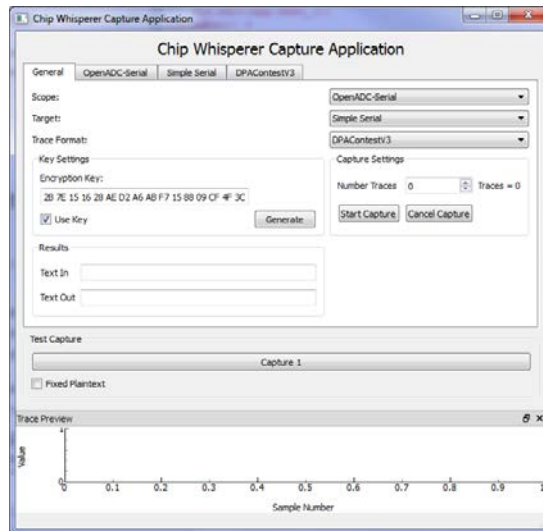
Getting Familiar

Start the chipwhisperer-capture application, and your window looks something like this:

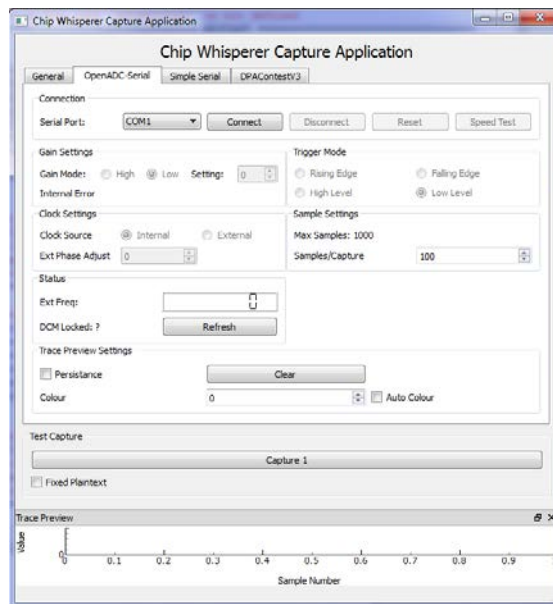
The ‘scope’ should be set to ‘OpenADC’. The ‘target’ should be set to ‘Simple Serial’, assuming you are using the targets described here. Finally you have the choice of how traces are stored

– the default ‘DPAContestV3’ is a format defined for the DPA Contest V3 (see dpacontest.org). The DPAContestV3 is very straight-forward, and consists of text files with the saved waveforms & other required data.

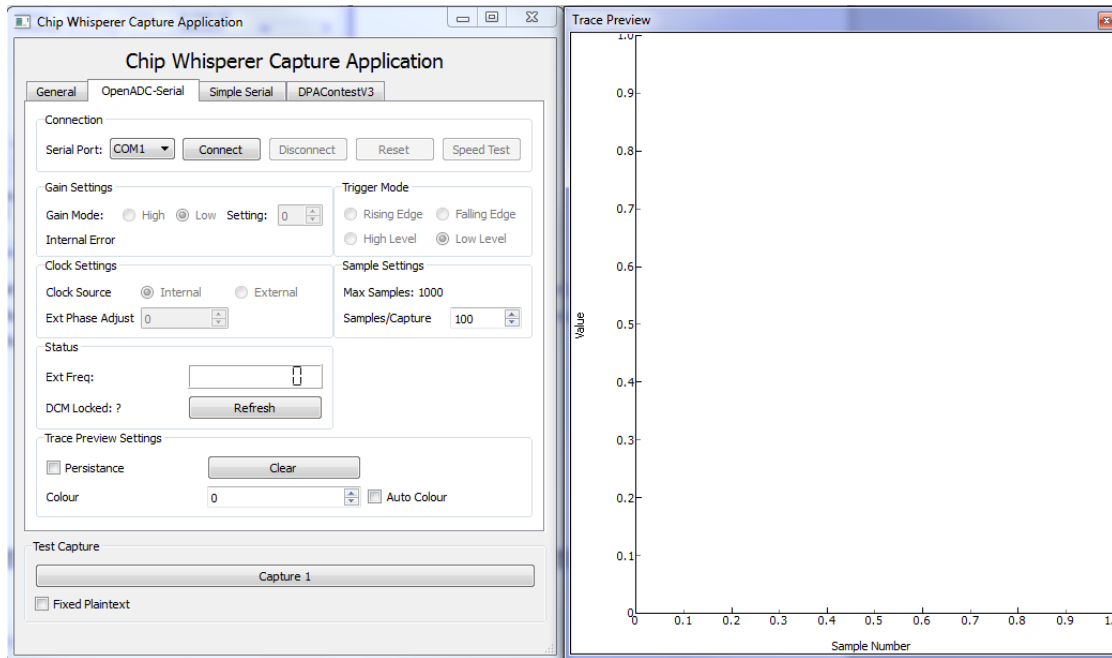
Go to the ‘Simple Serial’ tab, select the proper COM port and hit ‘Connect’. If your serial target is attached with a USB-Serial converter, you will need to figure out which one it is.



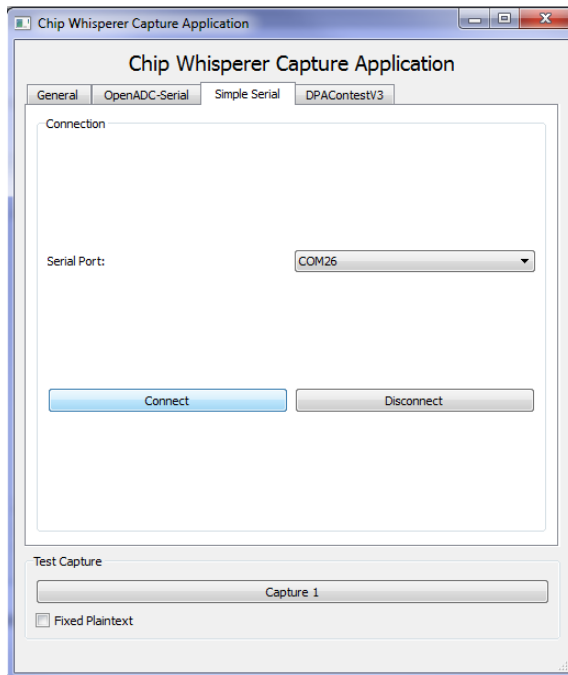
Go to the OpenADC tab, which looks like this:



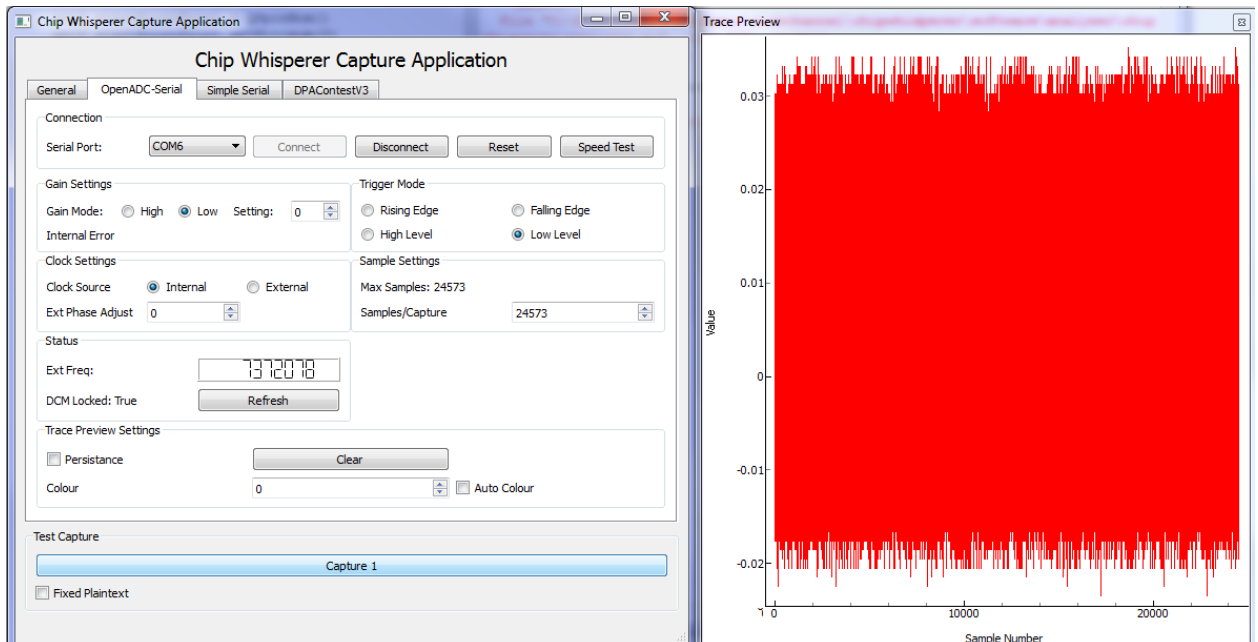
Select the proper COM port and hit ‘Connect’, assuming you are using the LX9 microboard with the OpenADC. You can move the ‘trace preview’ window off or otherwise scale it to make things easier to see:



Move to the target tab, in this case the ‘Simple Serial’ tab. Select the proper COM port and hit ‘Connect’:



You can hit 'Capture 1' now to try and get a trace, you should see something appear in the 'Results Preview' graph:

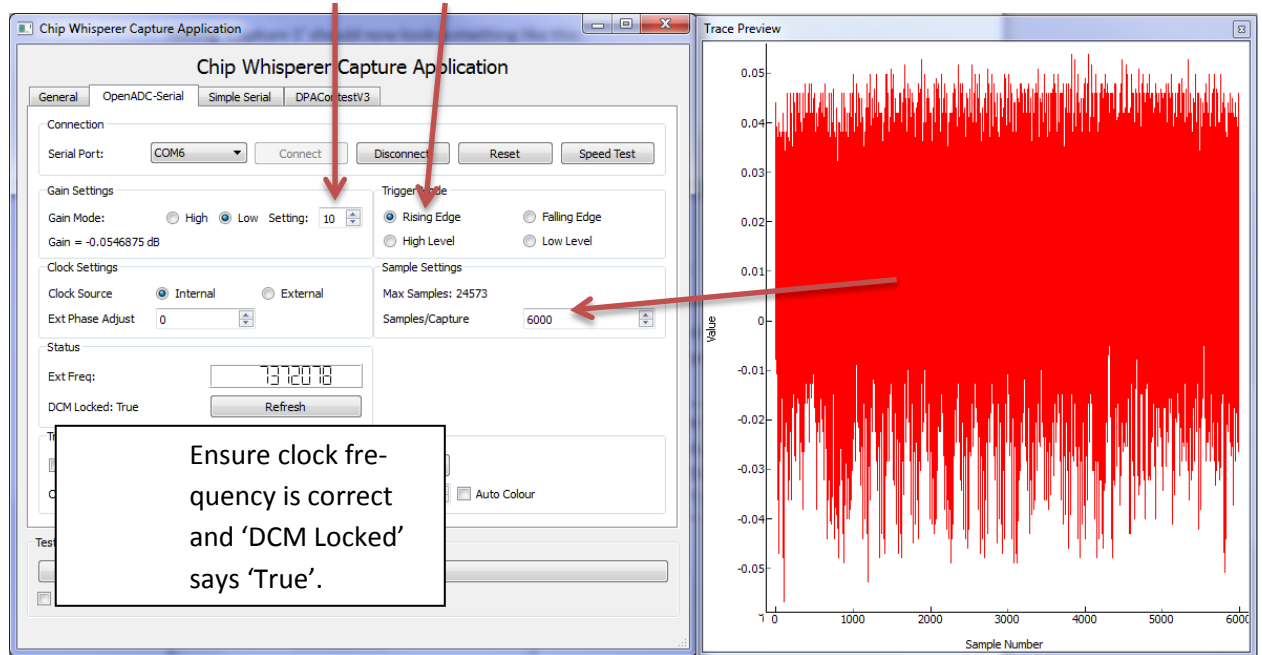


In the 'Status' field you can hit 'Refresh' and see if the external clock is working. You should see the correct frequency here should be around 7372800. If you get 0 here you need to check the clock path, or maybe the crystal isn't oscillating.

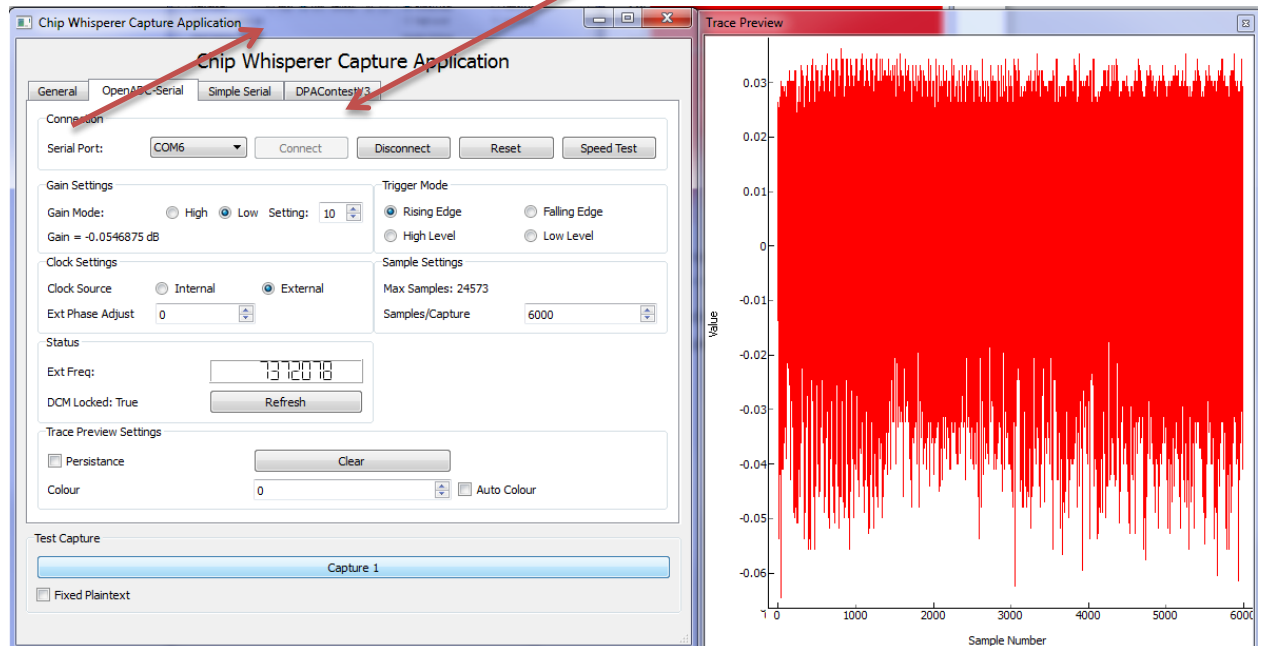
At this point the system isn't properly setup though. You should adjust the following:

- Set Trigger Mode to 'Rising'
- Turn gain up slightly (say to '10')
- Set Samples/Capture to 6000

Hitting 'Capture 1' should now look something like this:

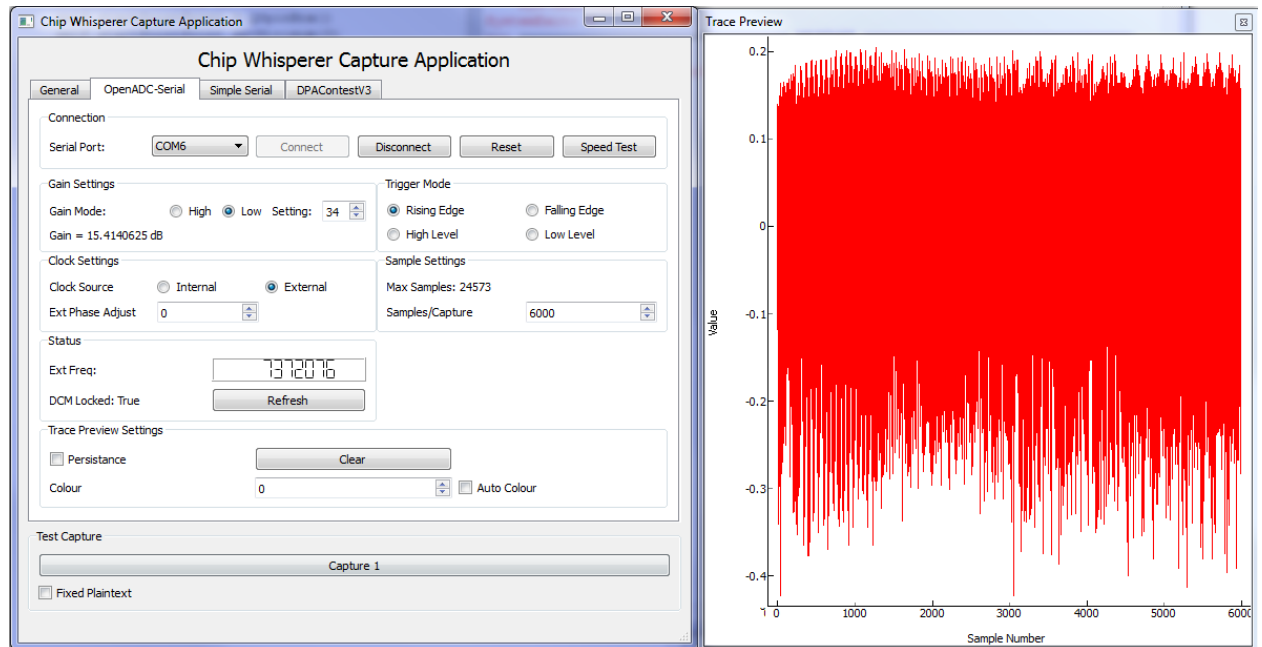


The next thing to check is what happens when you set the sample clock to 'external'. Again see if 'Capture 1' works. Now that the sample clock is running slower and perfectly synchronized, you will see the signal change:



Adjust the gain upwards a bit. Note the total range is +/- 0.5. Thus the previous signals weren't hitting the upper limit yet, but you don't want to clip! You can use the mouse button to select

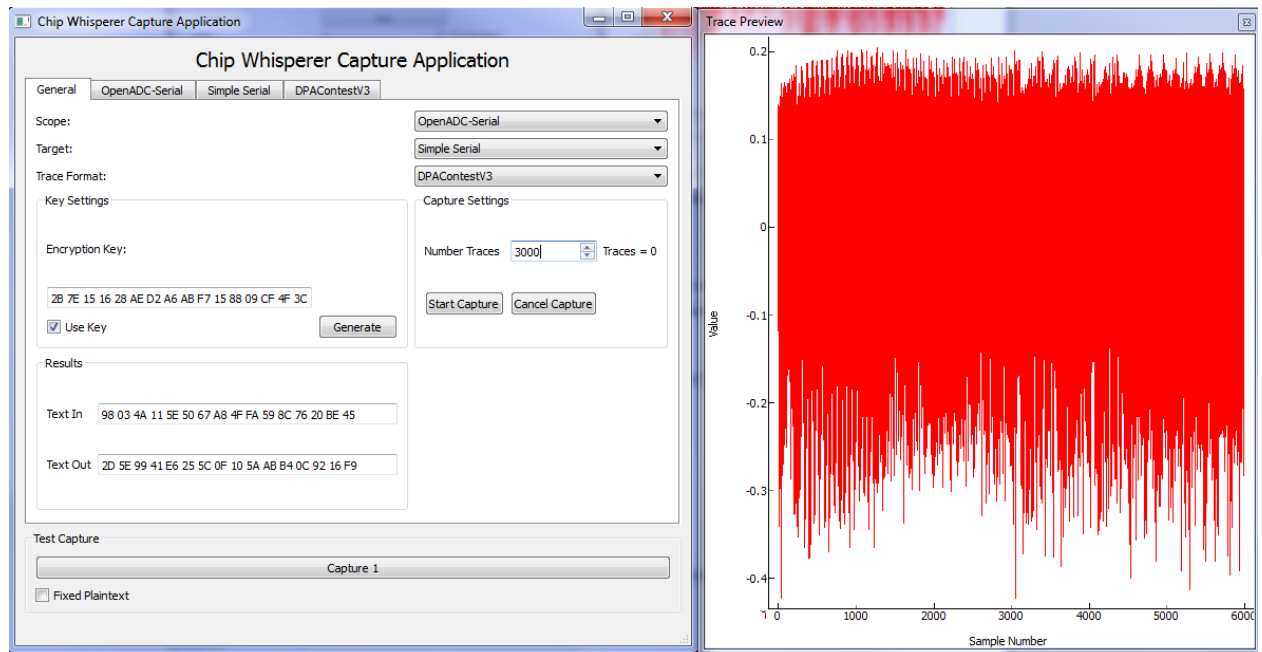
an area to zoom in on, and right-click to get the menu to unzoom.



You can also adjust the ‘Phase Adjust’. This is highly dependent on your specific target. Some targets, such as the MEGA163 card, are very particular to the phase adjust.

You can play around with them a little to see visually what they do.

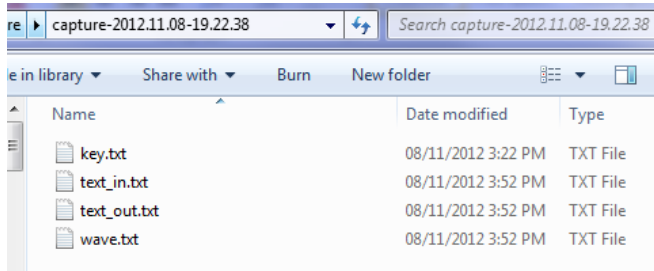
Now all you need to do is capture several traces. On the ‘General’ tab set the number of traces to 3000 for example. Then hit ‘Start Capture’, you will see the Trace Number increase, and each time the new trace will be written to the preview window.



It will take some time for this to complete, so go have a coffee (or beer). Once it completes the files will be written to a directory in the same file:

Name	Date modified	Type
capture-2012.11.08-19.22.38	08/11/2012 3:22 PM	File fo
aesexplorer-capture.py	08/11/2012 2:36 PM	Pythoi
openadc.py	29/09/2012 4:15 PM	Pythoi
openadc.pyc	08/11/2012 10:00 ...	Comp
openadc_qt.py	29/09/2012 4:15 PM	Pythoi

You can see the files in this directory contain all the data required for the attack:



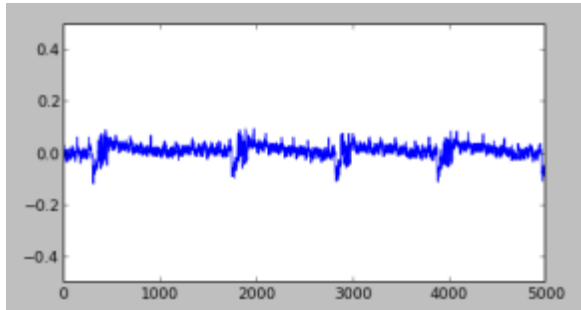
Name	Date modified	Type
key.txt	08/11/2012 3:22 PM	TXT File
text_in.txt	08/11/2012 3:52 PM	TXT File
text_out.txt	08/11/2012 3:52 PM	TXT File
wave.txt	08/11/2012 3:52 PM	TXT File

For now we won't be doing the attack though, just the capture part. Now that you've got an idea of what is required, let's look at more details of the capture.

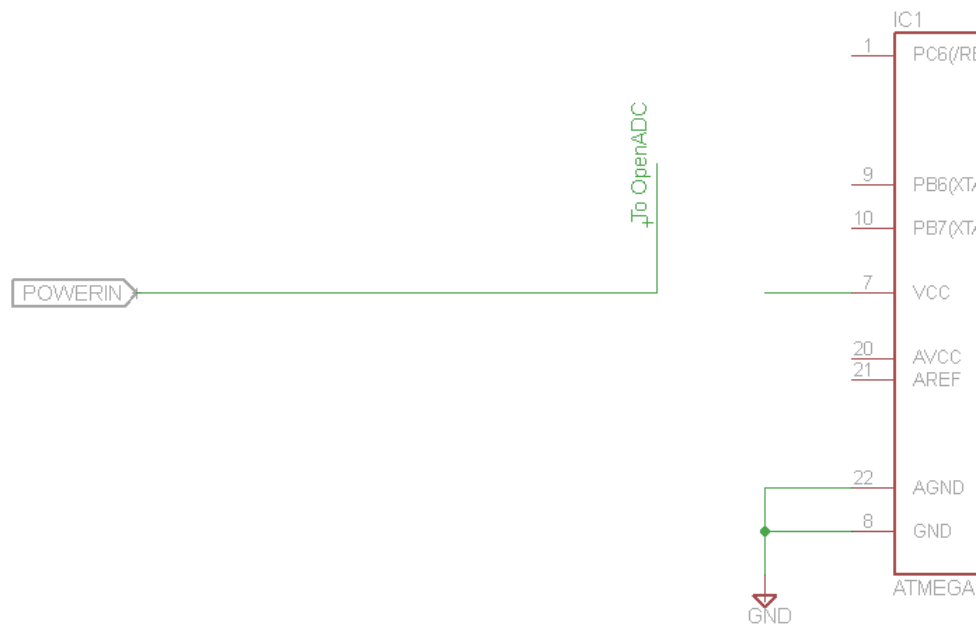
Capturing Traces: Advanced Topics

Considering Noise

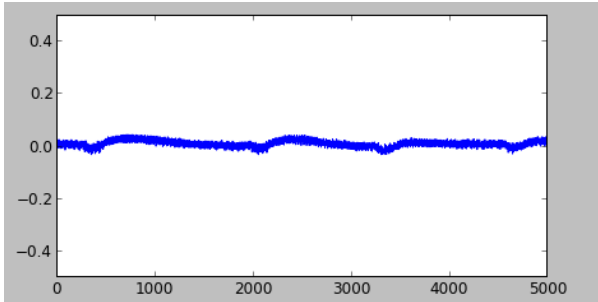
Noise can come from many sources, and you need to minimize it in your measurement system. Consider the following capture. This was done by measuring just the power rail of a target, without the microcontroller itself present:



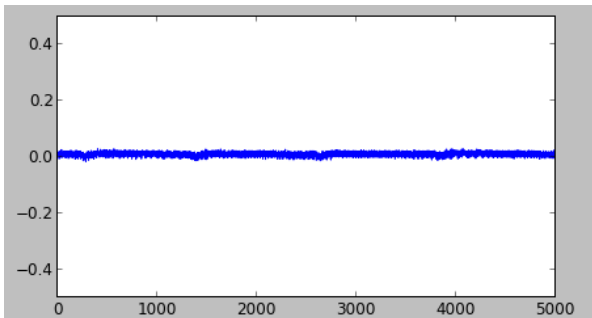
This is bad! There is far too much noise in this measurement! The voltage regulator for this example comes from a USB chip (CP2102), which has a fairly poor regulator! This is where all the noise comes from, so your systems will probably have less noise. Schematically, this looks like the following:



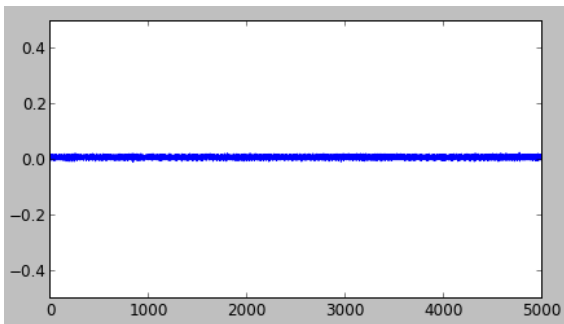
Now, we add a 2.2uF ceramic capacitor at the power rail. The trace now looks like this:



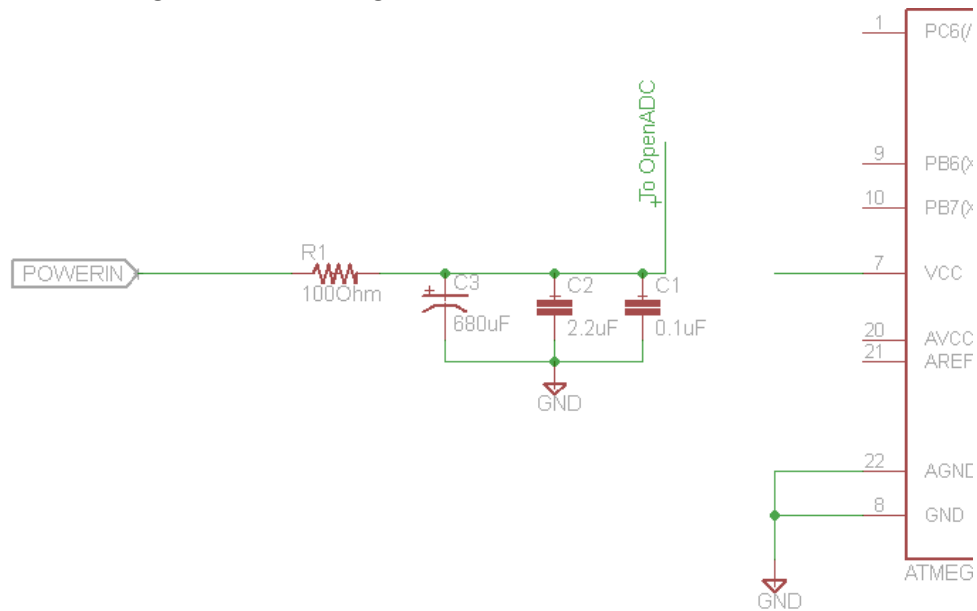
A slight improvement. Next we add a 680uF capacitor to try and even out the lumpy bits:



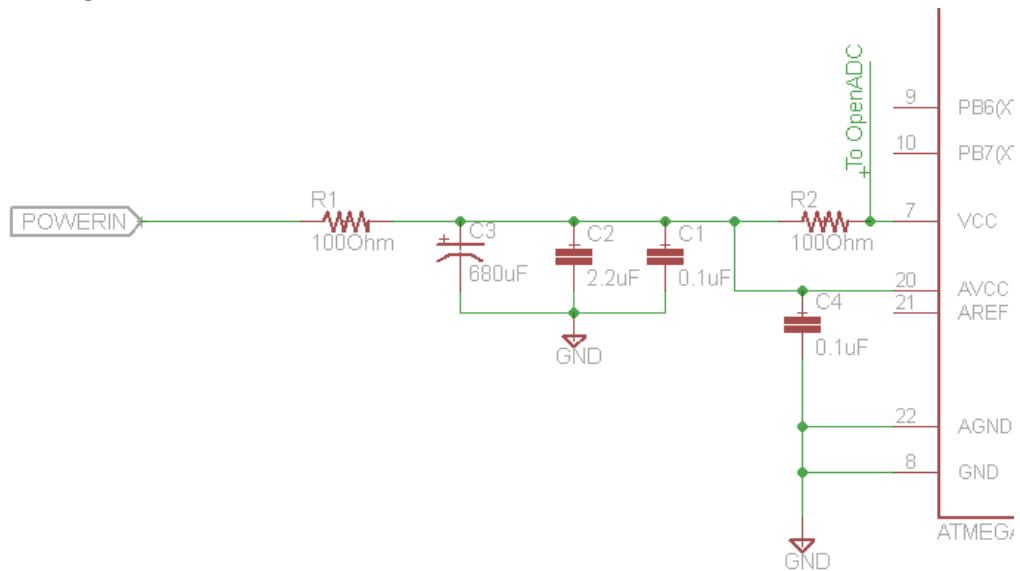
Finally we add a resistor in-line. This helps decouple us from the regulator to give us the final clean signal:



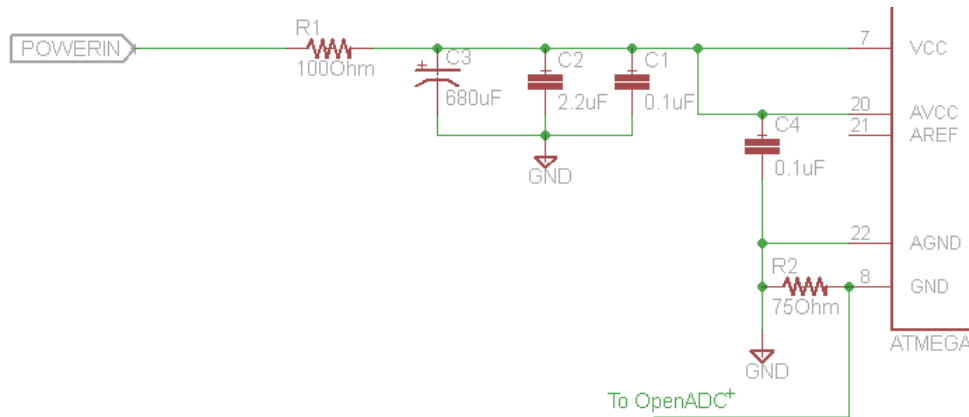
With the resulting schematic looking like:



So to do the actual measurement we insert a shunt (and also connect up AVCC), and measure the voltage at the microcontroller side of the shunt:



Note this example is using the shunt in the VCC line. You may get better results with the shunt in the GND line instead, which would look like:

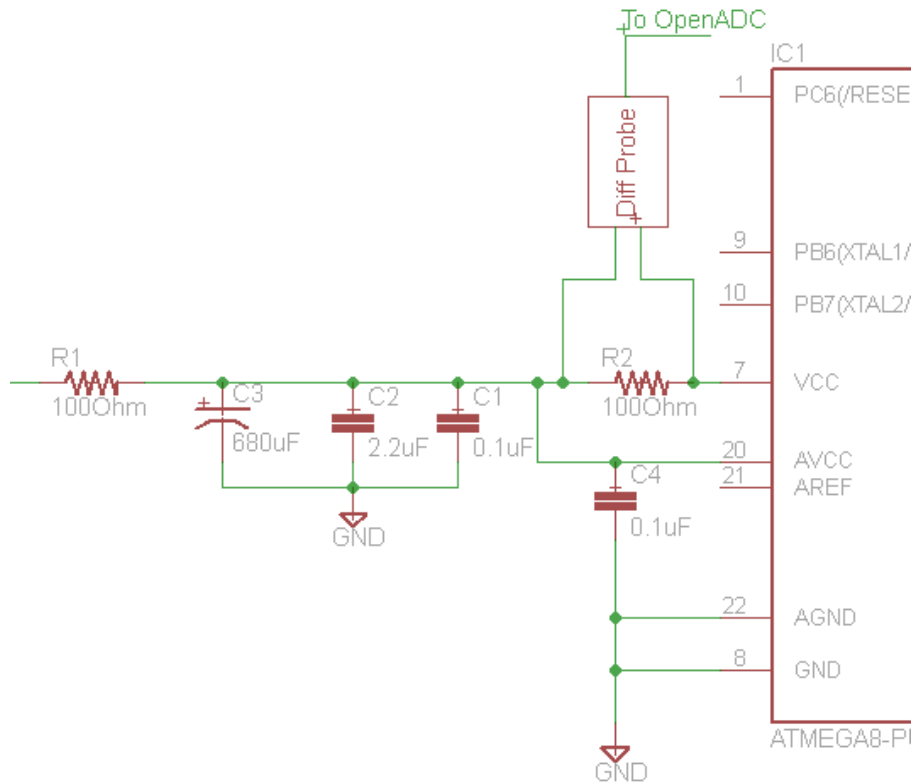


Differential Probe

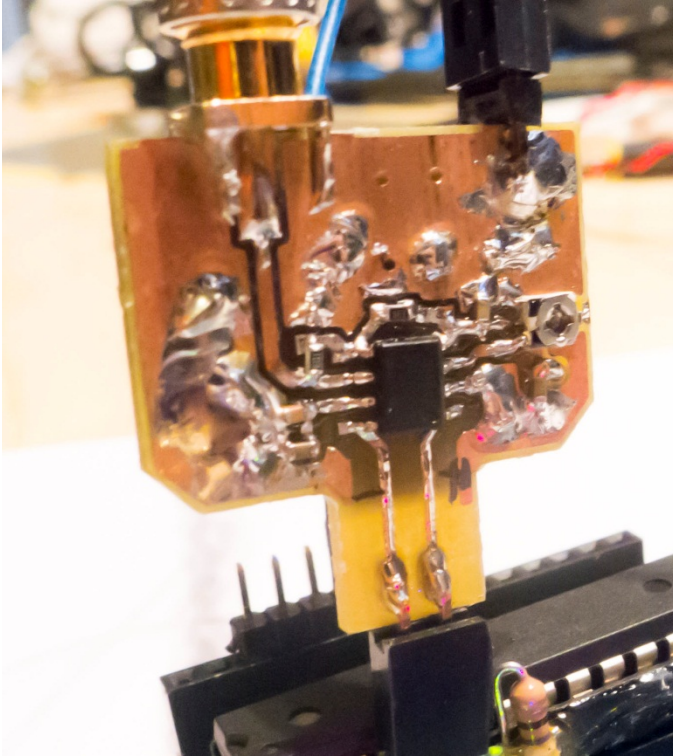
The previous diagram showed measuring the voltage across a resistor. This was measured 'single-ended'. Besides just measuring the drop across the resistor, you will also measure any variation in voltage. This could be due to the regulator, environmental noise, or voltage variations from other areas of the circuit switching.

Rather than use a single-ended probe, a differential probe ignores that variation in voltage that is *common* to both sides of the resistor, also called 'common mode' voltage. Based on the

same schematic as before, here is what the differential probe would look like:



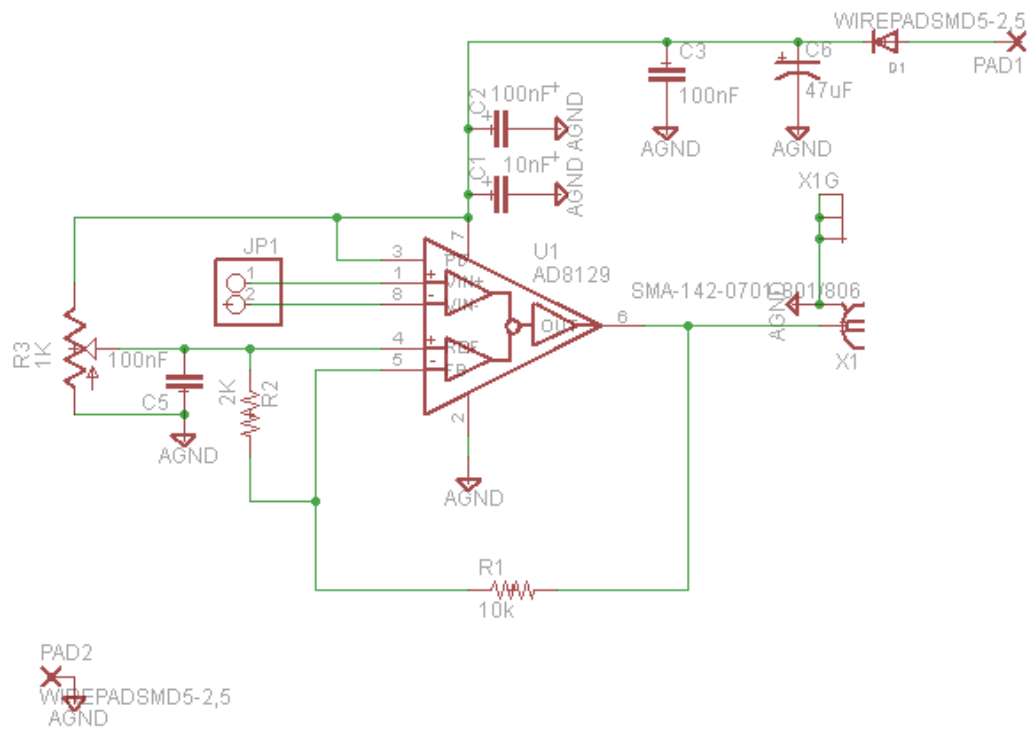
You can buy differential voltage probes, but they are pretty expensive. So here I will show you how to build one for around \$20 (depending what you've got on-hand). The end result might look something like this:



The critical part is the AD8129/AD8130 differential amplifier chip. You will typically add some gain right here, as that will reduce the ability of noise to affect the signal. How much gain to add is up to you – with the OpenADC you don't want too much gain, as the maximum input signal swing is limited.

The AD8130 part will work if you want gain of < 10 (recommended for your first probe with the OpenADC). If you want higher gain use the AD8129 part, which is only stable for gains > 10 . If you are building a differential probe for a generic scope, you'll probably want gains of more than 10.

The schematic is shown in the following image:

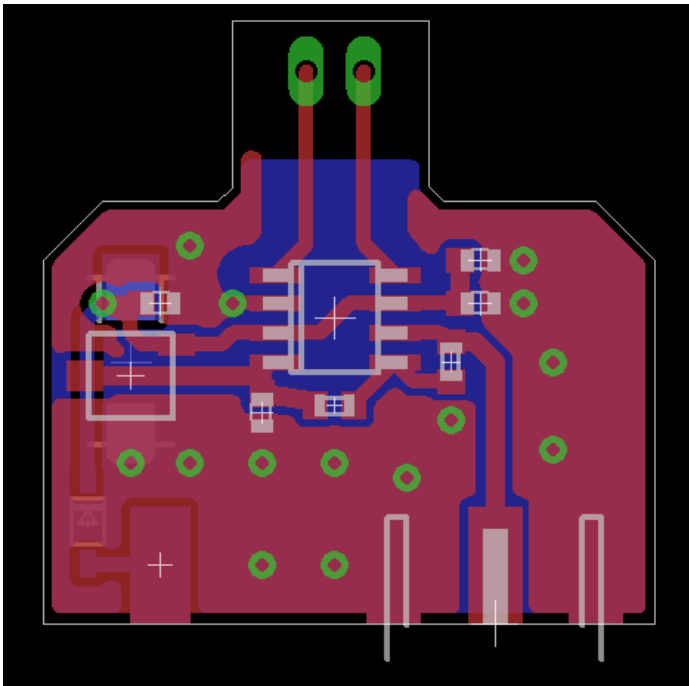


Note that R3 should be a multi-turn trimmer pot (unlike the single-turn one shown in the images). The gain is set with:

$$G = 1 + (R1 / R2)$$

Note the gain for the example schematic would thus be about 6. This should actually be built with an AD8130 part, not the AD8129. This schematic is based on my actual design, and if you are starting again should modify it.

A simple PCB layout is shown below, which the CADSOFT Eagle source files are included for:



The AD8129 must have supply voltages with at least about 1.3V of headroom. In the given design $-VS$ is connected to ground. This means you COULD NOT use the provided design on a shunt in the ground path, since the lowest common mode voltage it could measure is around 1.3V.

You must also thus power it at a higher voltage than the operating voltage of the system you are measuring. If you are measuring a 3.3V system, set VCC of the probe to 5V. If measuring a 5V system, power the probe from 7V.

If you want to use the probe on a ground shunt, you will need to generate a negative supply of at least -2V and connect that to $-VS$.

Usage Instructions

The following is a brief guide to using the probe.

1. Determine the device power supply, and power the differential probe by at least 1.3V more than this
2. Determine where you will plug it in – this means normally across a shunt inserted into the VCC line of the device under attack
3. Connect the positive differential input to the higher voltage side of the shunt (e.g.: side of shunt connecting to target VCC)
4. Connect the negative differential input to the lower voltage side of the shunt (e.g.: side of the shunt connecting to the device under attack)
5. Connect your oscilloscope to the output if you have one. If you are using the OpenADC, instead connect a multimeter.

6. Adjust resistor R3. If you start with the resistor at one extreme, you will see the output start at some fixed limited voltage. This is the op-amp trying to drive the output beyond what it is capable of. Typically this will be either around 1V or $V_{CC}-1V$ (e.g.: if powering from 5V, you'll see around 4V at the output). You want to adjust resistor R3 until the output is half-way between the two voltage supplies of the differential amplifier chip.
 - a. If +VS is 5V and -VS is 0V (GND), this means you want the output to be around 2.5V
 - b. If +VS is 7V and -VS is 0V (GND), this means you want the output to be around 3.5V
 - c. If +VS is 5V and -VS is -2V, this means you want the output to be around 1.5V
7. Note the resistor will be very finicky! If you didn't use a multiturn resistor here you will be kicking yourself, as the tinniest turn of the screwdriver may cause the output to go to the other rail
8. If you cannot get the previous steps to work, the differential voltage across the input may be too high. Short the inputs together as a test, or substitute a smaller shunt resistor.
9. Now that you've set resistor R3, connect the differential probe to the OpenADC/scope (if not already done) and capture as normal

Electromagnetic Probes

Rather than measuring the current across a shunt, you can also measure the electromagnetic field from a chip. You can build an electromagnetic probe with a length of semi-rigid coax:



Then form it into a loop and solder the shield as shown. Cut a little bit of the shield away:



You can see more details of constructing such probes at [4] and [http://www.compliance-club.com/archive/old_archive/030718.htm]. Specific to side-channel analysis, you might want to read [5] as well. These probes can be connected to the OpenADC, you will need to set gain to maximum.

Building Amplifiers

For the electromagnetic probes mentioned above, you may ultimately need an external amplifier. The signals coming from those probes is very faint, and the OpenADC may not provide quite enough gain.

The simple solution is to purchase one of these ‘Block’ RF amplifiers. They normally look something like this:



The following are some example of such amplifiers:

- MiniCircuits ZFL-1000LN
- MiniCircuits ZFL-500LN

Alternatively, you can build one for even cheaper! MiniCircuits (and others) sell nice little LNA amplifier chips, such as the MAR-8SM+:

Surface Mount

Monolithic Amplifier

DC-1 GHz

Product Features

- Wideband, DC to 1 GHz
- Exact footprint substitute for Avago's MSA-0886
- Internally Matched to 50 Ohms
- Very high gain, 32.5 dB at 0.1GHz



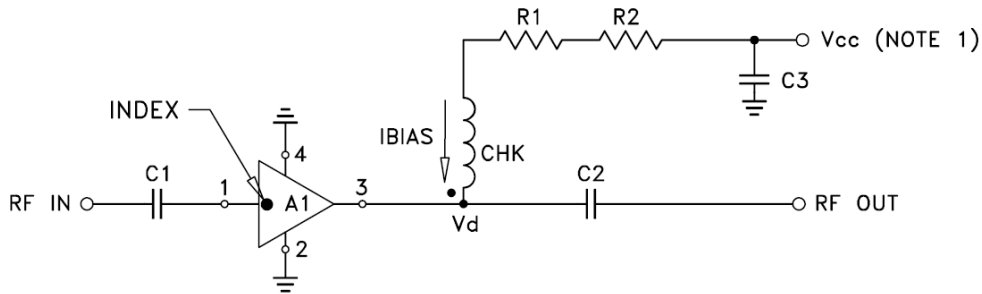
MAR-8SM+

CASE STYLE: WW107-1
PRICE: \$1.37 ea. QTY. (20)

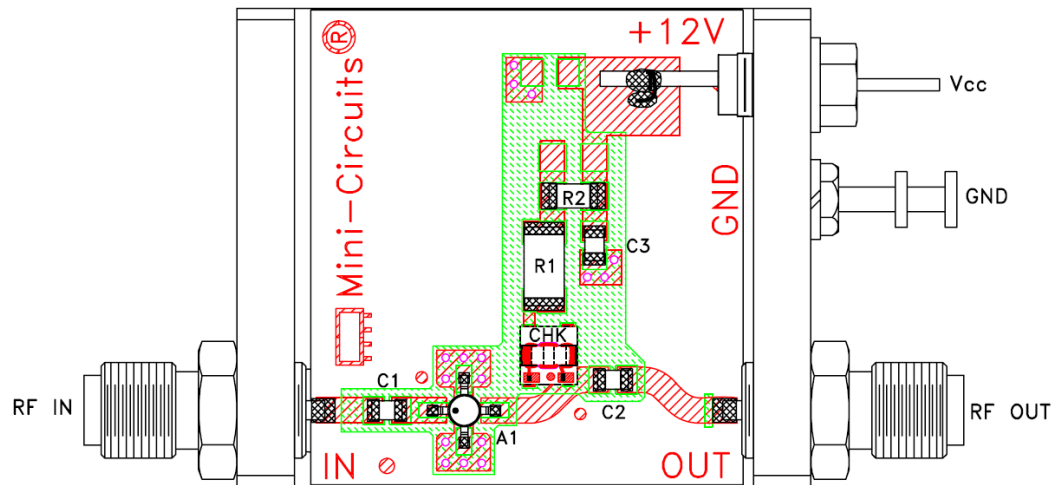
Typical Applications

If you can't find the MiniCircuits version, use the Avago MSA-0886 which as mentioned by MiniCircuits is equivalent. The MSA-0866 is available from Digikey etc for <\$2! Easy!

MiniCircuits provides the following schematic & layout:

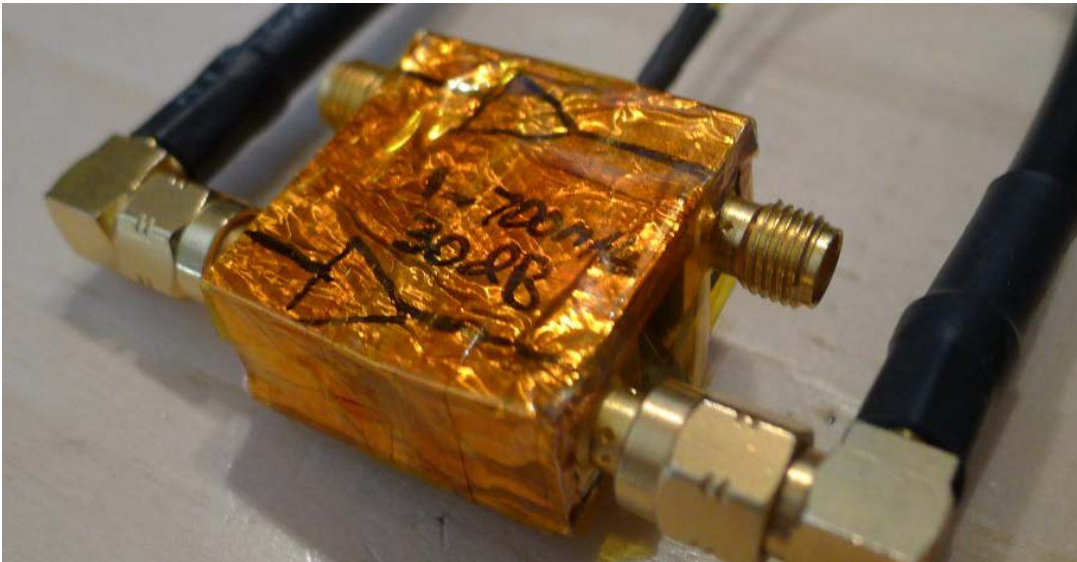
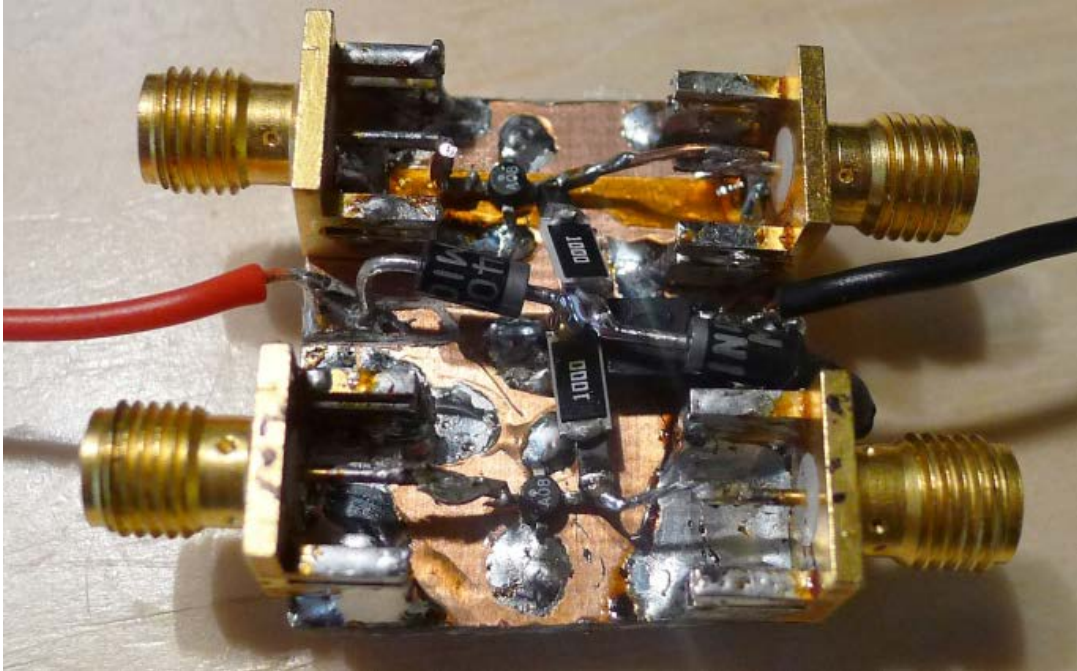


Evaluation Board and Circuit



<http://www.minicircuits.com/pcb/WTB-411-8+ P02.pdf>

You can base a simple amplifier on this! Be sure to add some diodes to prevent reverse polarity, or even a voltage regulator to make your life easier. The results could look like this:



WARNING: I tried to be too economical and built two amplifiers side-by-side. The cross-talk is very high, so you really shouldn't do this.

Target Practice - Hardware

In order to practice all this, you will need some targets. Basically any microcontroller can be used – I'm a fan of the AVR microcontrollers for my real engineering projects, so will use them here. Any microcontroller should work, but if you want to follow along I suggest trying and duplicating these results as closely as possible.

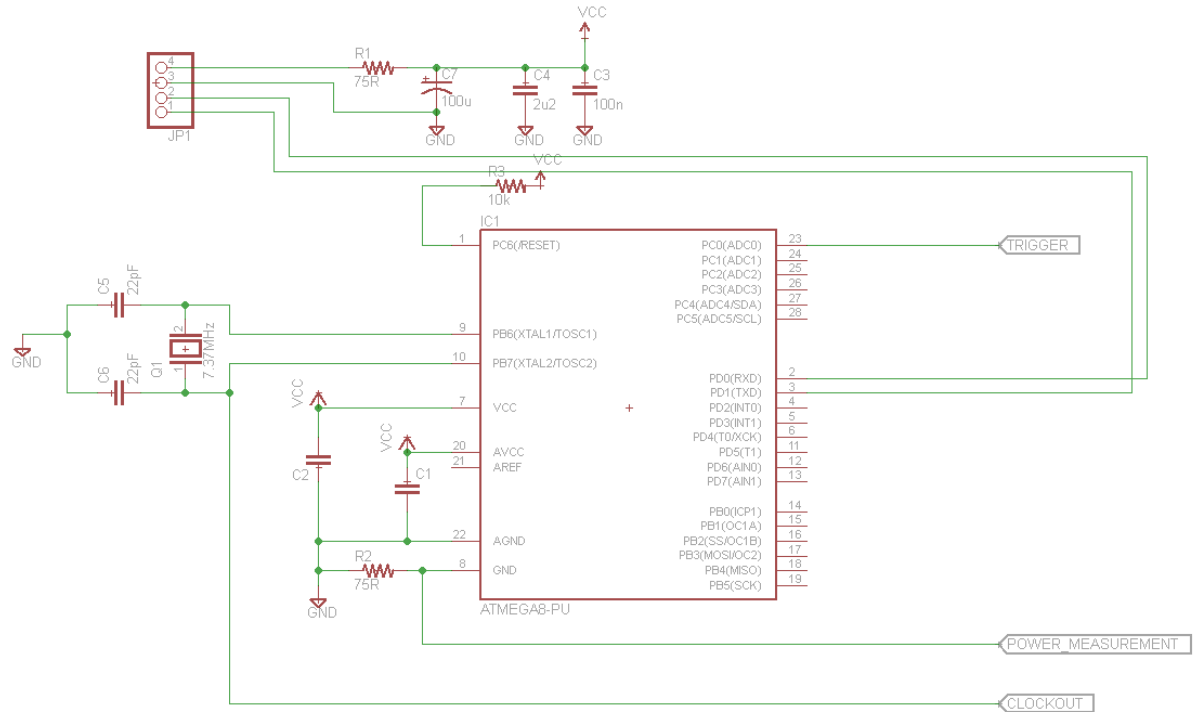
NOTE: I haven't included complete step-by-step instructions for most of these, especially for the firmware building & programming. You should already be familiar with AVRs and know how to write C code for an AVR and download it to the AVR. If not, you should first work on getting some simple C programs working on an AVR you wire up.

A few notes specific to the AVR targets:

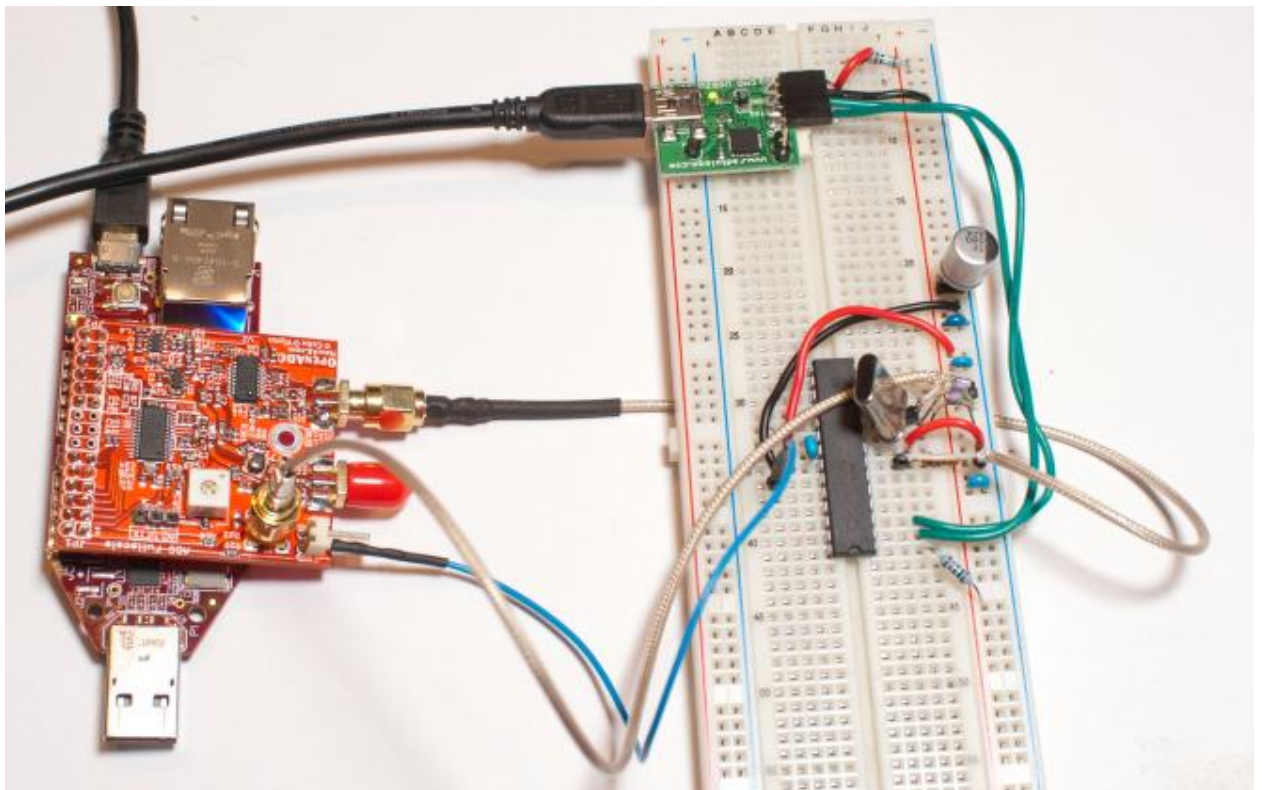
- We will use a port output as the 'trigger'. When that port switches it tends to contribute a lot of noise on the power trace, which is bad. To avoid this I recommend using one of the GPIO pins shared with the ADC inputs (PortA). The digital driver lines for PORTA are powered from the AVCC (Analog VCC) for most AVRs. This means you can add lots of decoupling on the AVCC power pins. This decoupling will not affect the power traces measured on the VCC pins, but will provide a good source of power for the trigger current.
- Most AVRs can be setup to output the system clock on a certain pin (CLKOUT). Some AVRs don't have this feature. Those that don't, use the XTAL2 pin which is the output of the crystal oscillator driver. To use this properly you may need to adjust the fuses on your AVR, in particular programming the fuse that causes the signal swing to be 'full-scale'.

Simple AVR Target

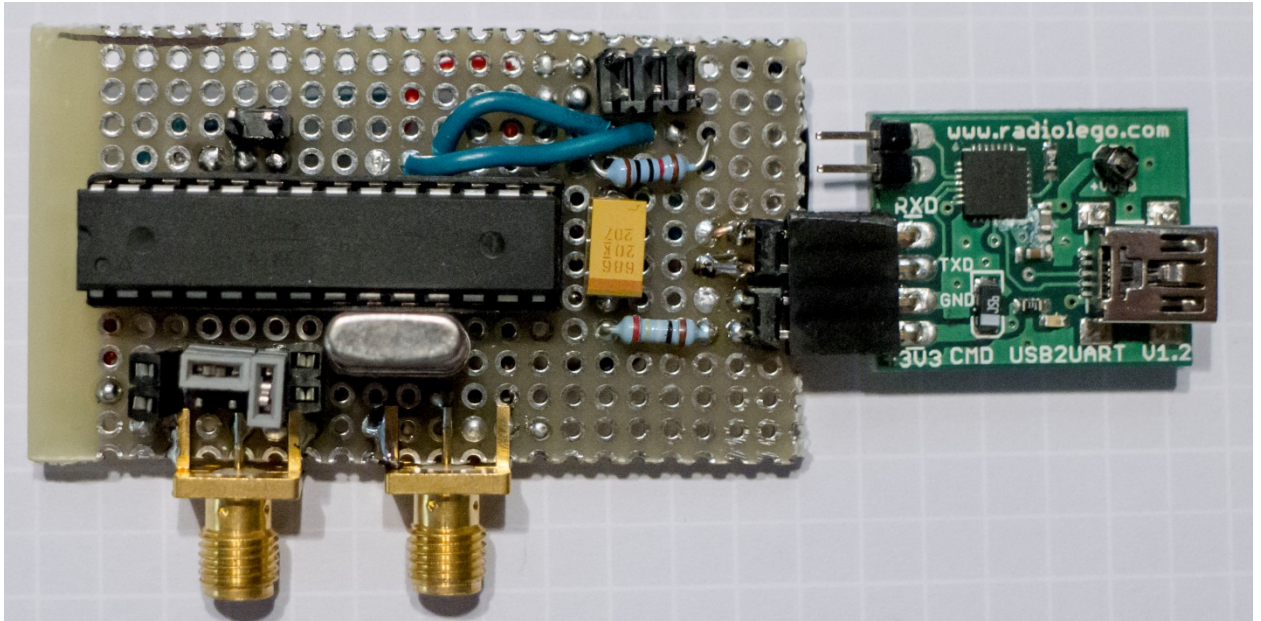
The first target to demonstrate is based on the Atmel AtMega8-16PU target. You will need a programmer tool for this to work (see Buying Guide). The basic schematic is shown below:



You can build this on a breadboard if you wish, as the following figure shows:



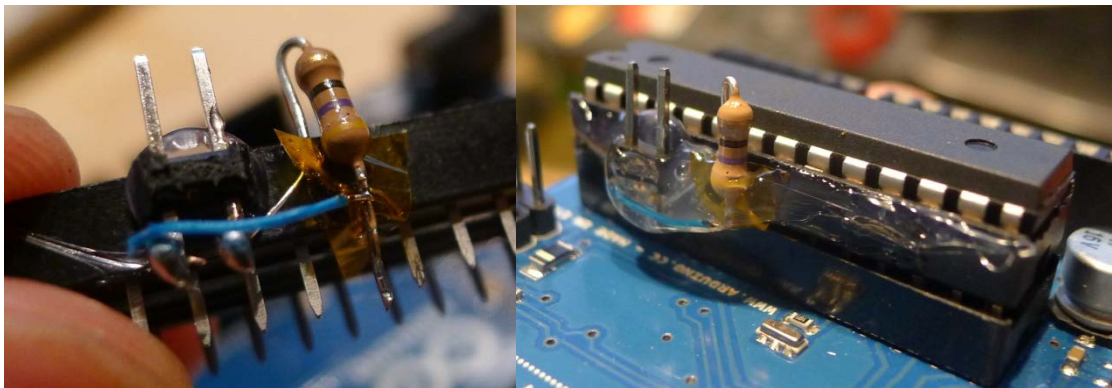
Alternatively you may wish to build this on a piece of PCB material so you can transport it easily:



To use this target, we connect the clock, trigger, and analog line to the OpenADC.

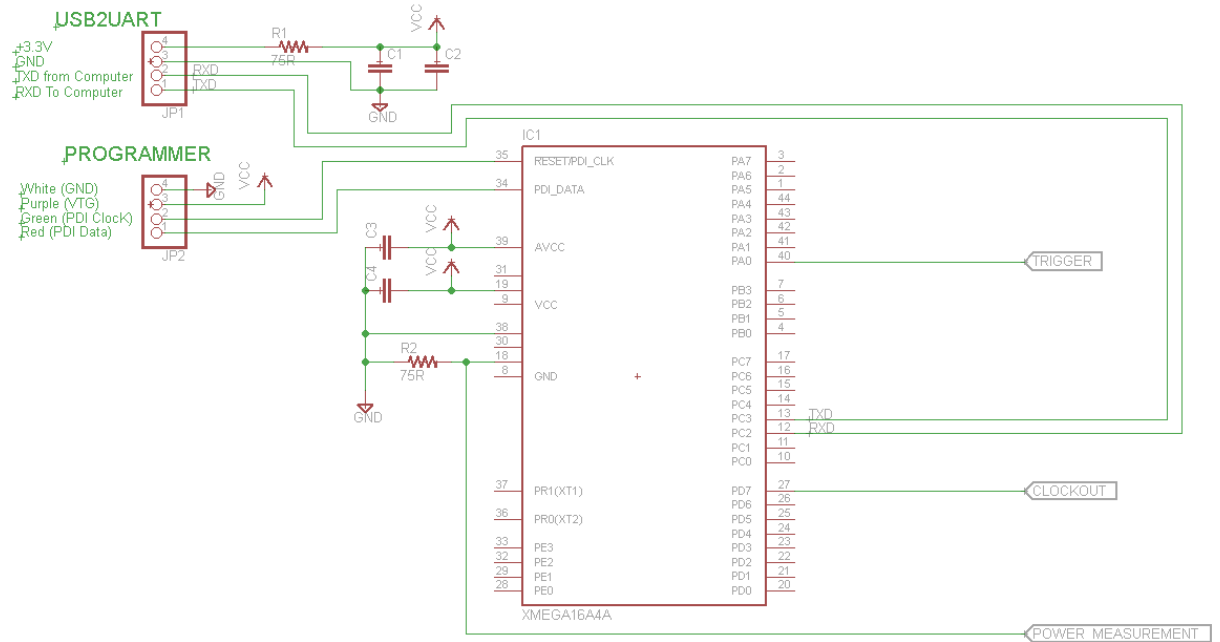
Arduino Target

The Arduino is a popular embedded device. It contains an AtMega328P device with a crystal oscillator – so we can use it to do some side channel analysis. You can either lift the pins of the DIP package IC, or build a little adapter socket that inserts a resistor in GND (preferred) or VCC (also works):

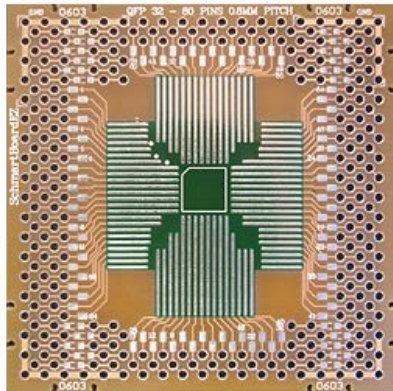


Simple XMega Target

The XMega device contains AES hardware, which has proven susceptible to Side Channel Analysis[6]. At this point we will only be demonstrating an attack of software AES on this platform, but it should be possible to expand this to attacking the hardware AES engine.



SchmartBoards are a very useful platform for building attack platforms. They allow you too easily solder SMD devices, and look something like this:

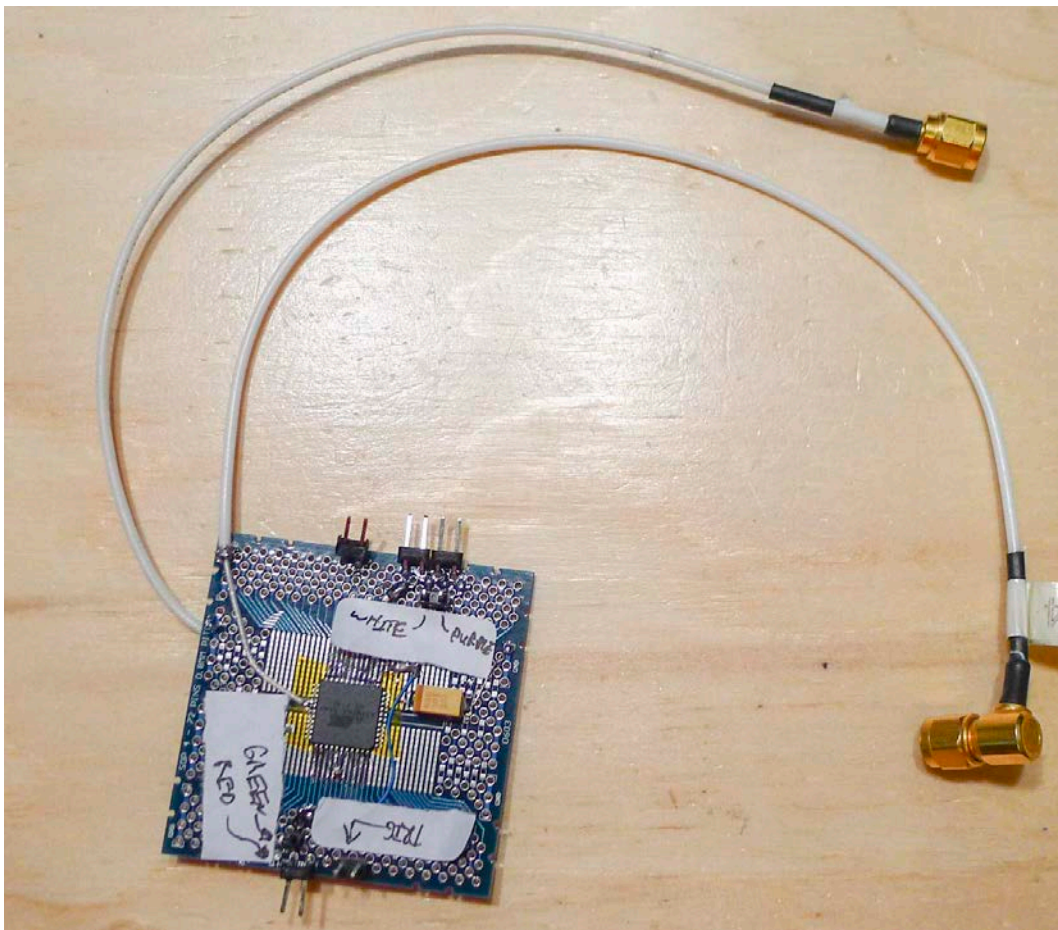


They have a very clever layout making it easy to add decoupling capacitors to ground on the backside. In addition we can use that to add a resistor in one line going to ground, like the fol-

lowing shows:



So the top-side of this target looks like this:



The White/Purple/Green/Red labels are indicating connections to the JTAG MK2 / JTAG3 / ISPMK2 programmer. The 'TRIG' label is the trigger. The 4-pin header on the upper right side connects to the USB2UART for serial communications and power.

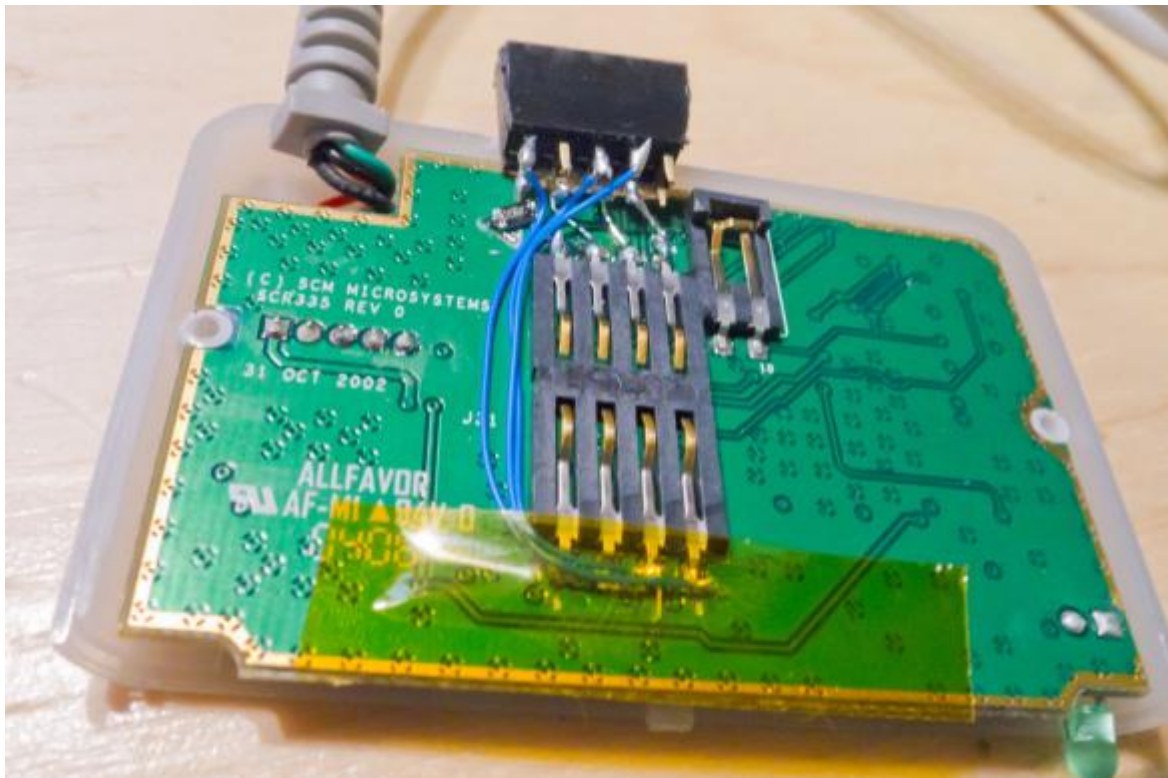
SmartCard Target: A Normal Reader

Smart cards have a certain degree of 'sexyness' when it comes to demonstrating attacks. A popular card for this demo is ones based on an AVR microcontroller. It's critical to understand the following:

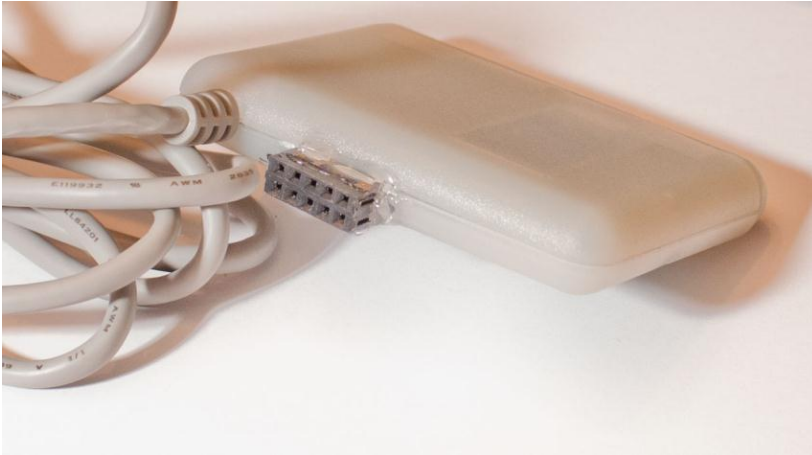


There is no difference between demonstrating an attack on one of these Smart Cards and demonstrating an attack on a discrete AVR. It's much easier to get a hold of discrete AVRs, so I don't even recommend bothering with the smart card attacks. Certain microcontrollers only come in the Smart Card form factor, so with those you **do** need to attack the smart card.

To attack the smart card, you simply insert a resistor in the power lines. Also bring out the clock pin:



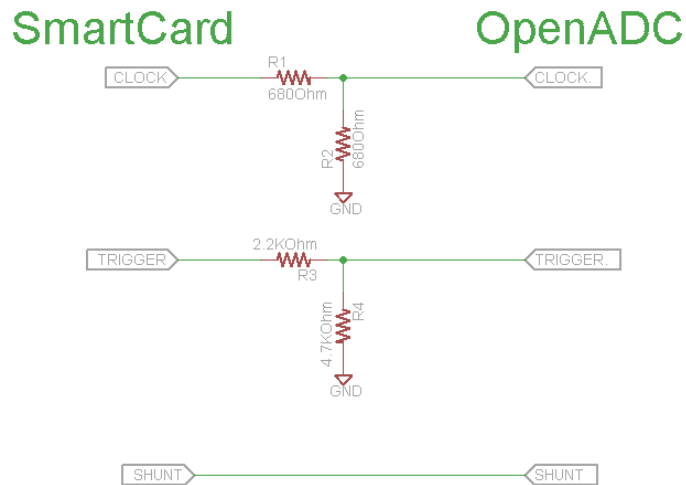
Close up the Smart Card reader, I added a connector on the backside:



WARNING: Your Smart Card will probably run at 5V. The FPGA connected to the OpenADC will be destroyed by 5V, it only accepts a 3.3V input signal. You need to add some resistive dividers on the Clock and Trigger line most likely. The analog input is fine at 5V due to being AC coupled.



My example resistor board on the backside has this basic schematic:

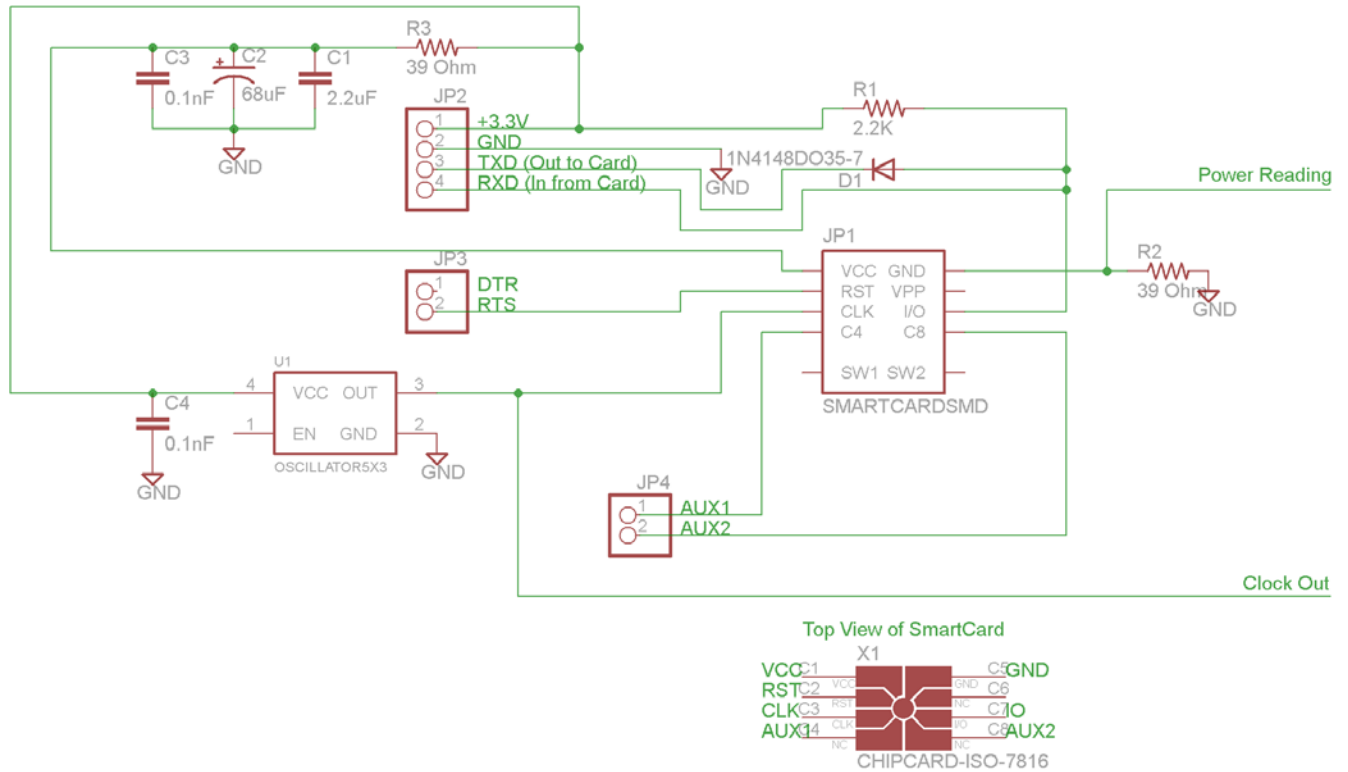


You MUST ensure the grounds of the two systems are connected for this to work! Note because the OpenADC as an AC-coupled analog input, the shunt line does not require any conditioning. Thus even though the shunt line is sitting at 5V, that will be fine.

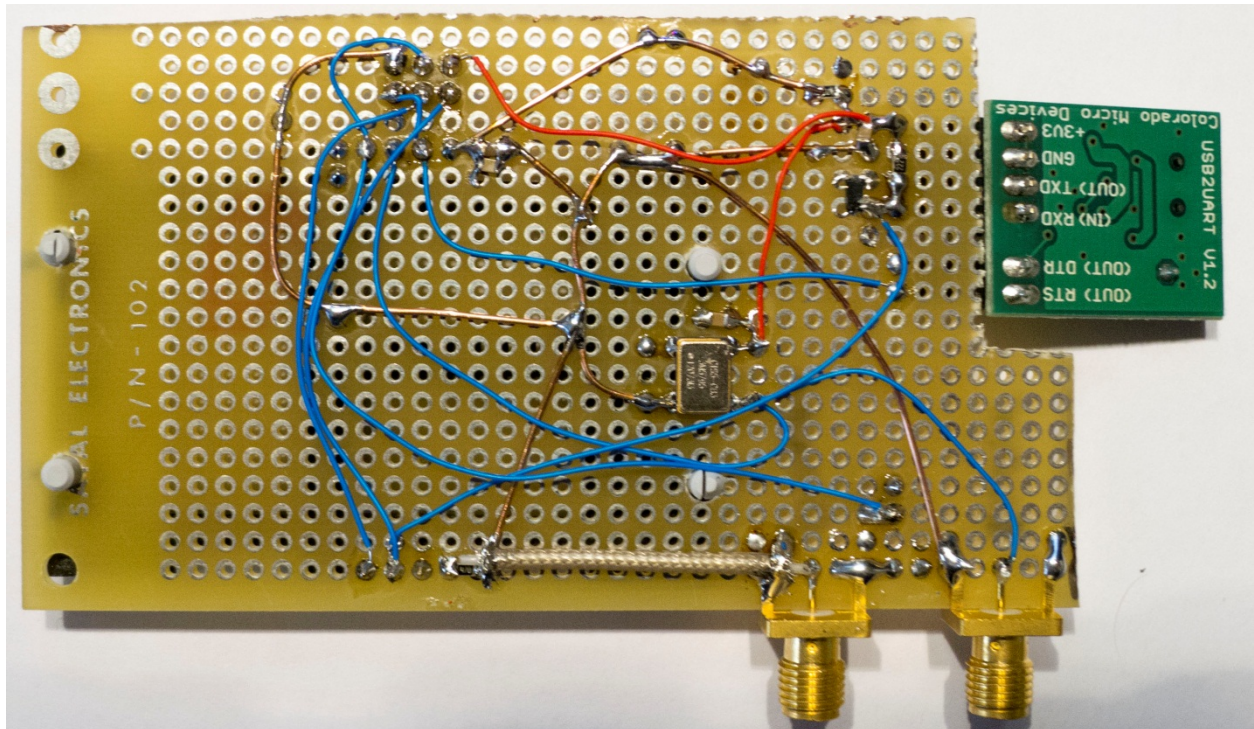
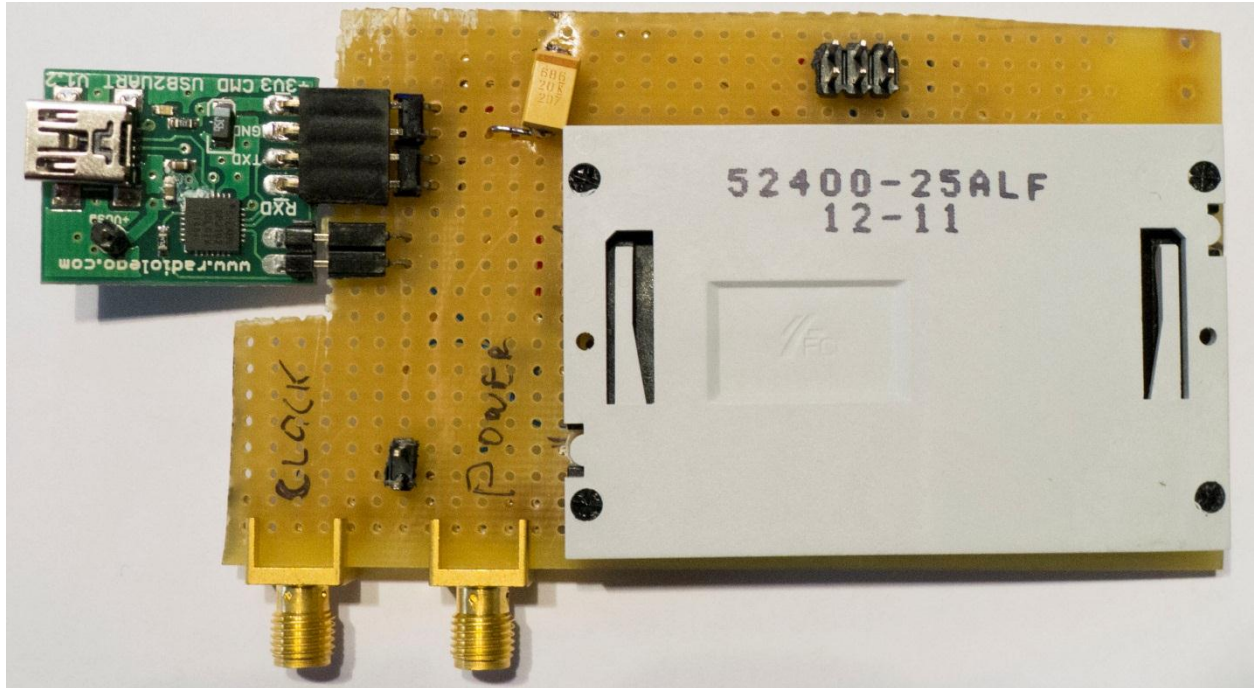
Smartcard Target – Cheapskate

A cheap reader can be built with a logic-level serial interface, smartcard connector, and a few discrete components. This smartcard reader can be run at 3.3V, meaning it directly interfaces to the OpenADC.

The reader uses a diode and pullup resistor to convert the unidirectional serial I/O lines from the computer to the single bidirectional I/O line of the smartcard. With this method any data sent on the TX pin will also be echoed to the RX pin, so the software on the computer must account for this.



- 3.579545 MHz 3.3V Oscillator (e.g. Digikey Part No. CTX929LVCT-ND)
- SmartCard Socket (e.g. Digikey Part No. 609-1414-ND)
- Small Signal Diode (e.g. 1N914 or similar)
- SMA Connectors (for OpenADC, not needed for scope)
- Resistors, Capacitors



Target Practice – Firmware

The firmware examples provide the example source code you can attack. Note the most recent source will always be available from the Chip Whisperer project

(<http://www.chipwhisperer.com>). This open-source project means you can contribute ports to other architectures or new attacks!

How to Build

All the current targets require an avr-gcc compiler. On Windows the easiest method is to download WinAVR 20100110. Then you can cd to the appropriate directory and simply type ‘make’. If you’ve done this correctly everything will build.

```

Assembling: ../crypto/avr-crypto-lib/aes/gf256mul.S
avr-gcc -c -mmcu=atmega8 -I. -x assembler-with-cpp -DF_CPU=7372800L -Wa,-gstabs,-adhlns=objdir/gf256mul.lst ../crypto/avr-
crypto-lib/aes/gf256mul.S -o objdir/gf256mul.o

Linking: simpleserial.elf
avr-gcc -mmcu=atmega8 -I. -gdwarf-2 -DAURCRYPTOLIB -DF_CPU=7372800L -Oe -funsigned-char -funsigned-bitfields -fpack-str
uct -fshort-enums -Wall -Wstrict-prototypes -Wa,-adhlns=objdir/simpleserial.o -I../crypto/avr-crypto-lib/aes -I../crypto
-std-gnu99 -MMD -MP -MF .dep/simpleserial.elf.d objdir/simpleserial.o objdir/uart.o objdir/aes-independent.o objdir/aes
-enc.o objdir/aes_keyschedule.o objdir/aes_sbox.o objdir/aes128_enc.o objdir/gf256mul.o --output simpleserial.elf -Wl,-M
ap=simpleserial.map,--cref -ln

Creating load file for Flash: simpleserial.hex
avr-objcopy -O ihex -R .eeprom -R .fuse -R .lock -R .signature simpleserial.elf simpleserial.hex

Creating load file for EEPROM: simpleserial.eep
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
--change-section-lma .eeprom=0 --no-change-warnings -O ihex simpleserial.elf simpleserial.eep !! exit 0

Creating Extended Listing: simpleserial.lss
avr-objdump -h -S -z simpleserial.elf > simpleserial.lss

Creating Symbol Table: simpleserial.sym
avr-nm -n simpleserial.elf > simpleserial.sym

Size after:
AVR Memory Usage
-----
Device: atmega8

Program: 2134 bytes (26.0% Full)
(.text + .data + .bootloader)

Data: 352 bytes (34.4% Full)
(.data + .bss + .noinit)

----- end -----

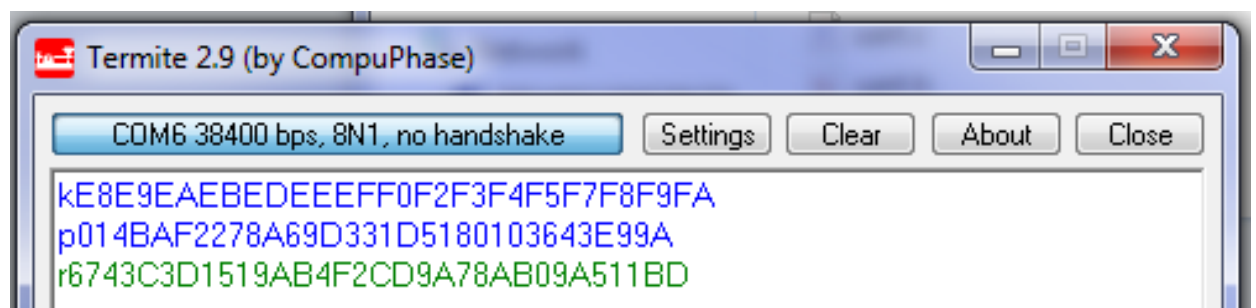
c:\N\Documents\academic\sidechannel\aesexplorer\software\targets\avr-serial>

```

SimpleSerial Firmware

Description

Currently there are two target folders: avr-serial and xmega-serial. Both these implement the ‘Simple Serial’ protocol running at 38400 baud rate. In this protocol sending in ASCII ‘kE8E9...F9FA’ would load the encryption key E8:E9:...:F9:FA. There will be no response by the target device. Then sending ‘p01BA...9A’ would cause the device to encrypt with the input plaintext of 01:BA:...:9A, and send the result with an ‘r’ in front of it. See the following diagram for an example:



For the avr-serial firmware, you can change the target in the Makefile and clock frequency the AVR is running at.

Programming AVR & XMega

To program the AVR target, you will need an AVR programmer capable of 'ISP' programming such as the AVRISP-MK2. You can use AVR Studio 4 for programming, or the newer Atmel Studio 6. Note certain AVRs may be removed from AVR/Atmel studio – for example to program the AtMega8, which does not appear on the list of valid AVR devices, select the 'AtMega88' instead. You may have to use AVR Studio 4 in this specific example to 'override' the device signature check. If programming succeeds it will still verify the flash OK.

The XMega device can also be programmed through the AVRISP-MK2. See the help file for details of the pinout, as this uses the 'PDI' mode .

SmartCard Firmware

Description

The smart card firmware is currently not part of this project. You can download the firmware & source from <http://www.morita-tech.co.jp/SAKURA/en/hardware/SASEBO-W.html> (see bottom of page – “Software for ATMega163 card”).

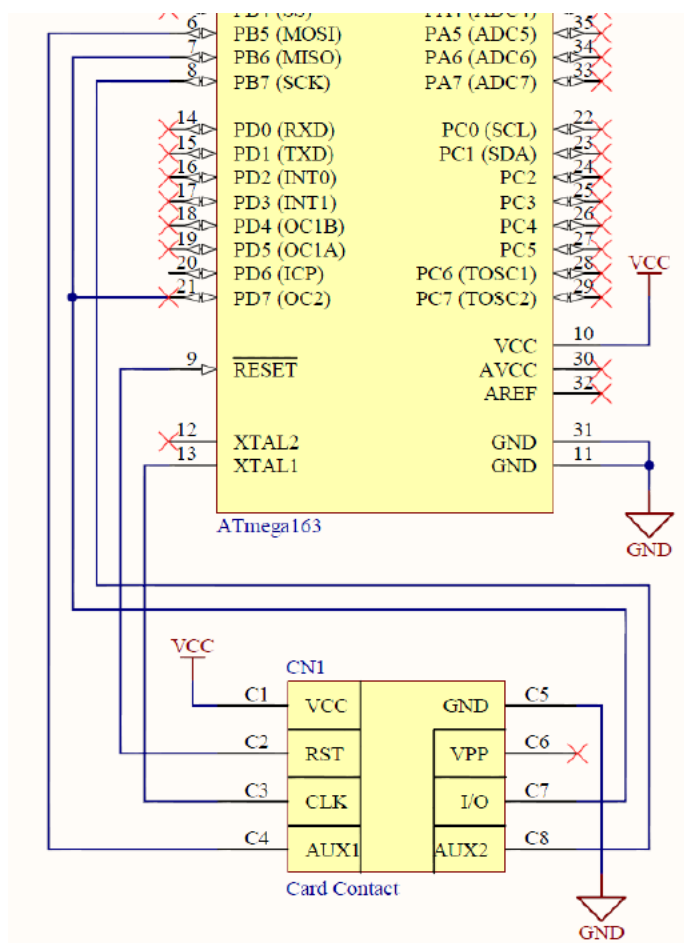
Programming

If you have a SASEBOW-W, it has already been designed with programming capabilities, see the QuickStart guide.

If you are buying a normal smartcard programmer, see the user guide for that device.

If you are building the cheapskate smartcard reader, you can add a 6-pin ISP header. Note the cheapskate reader provides a suitable clock already, all you need to do is wire out the RESET, SCK, MISO, and MOSI pins from the smartcard.

When programming, use AVR Studio 4. It will not support the Mega163 device, so instead select the 'Mega363'. Again tell it to ignore the issues of incorrect signature bytes, the programming should still succeed.



The cryptography implementation being used in these targets is selectable – it’s all based on open-source AES implementations. Many real devices are made by an engineering typing into Google ‘AES AVR’ and looking at the results. In this case you have three options for the crypto on the targets, although more are being added:

1. avr-crypto-lib C implementation from <http://avrcryptolib.das-labor.org> . This is the default library used, and is a crypto library written in C. Only the AES part is included here. This is the recommended first target, as it is fairly easy to crack, but still a ‘real’ cryptographic library probably used in products.
2. ‘Straight-forward’ C implementation from http://cs.ucsb.edu/~koc/cs178/projects/JT/avr_aes.html . The version as written is subject to timing attacks, so has been quickly modified to make it timing-independent. Note this modification is not done in a safe way, so if you use a different compiler version it may fail. This implementation
3. avr-crypto-lib ASM implementation from <http://avrcryptolib.das-labor.org> . This assembly version of AES is much faster than the C implementation, and requires more traces to completely break. It is not protected however, and can still be broken with DPA.

The crypto library to use is selected in the Makefile. Look for the following line:

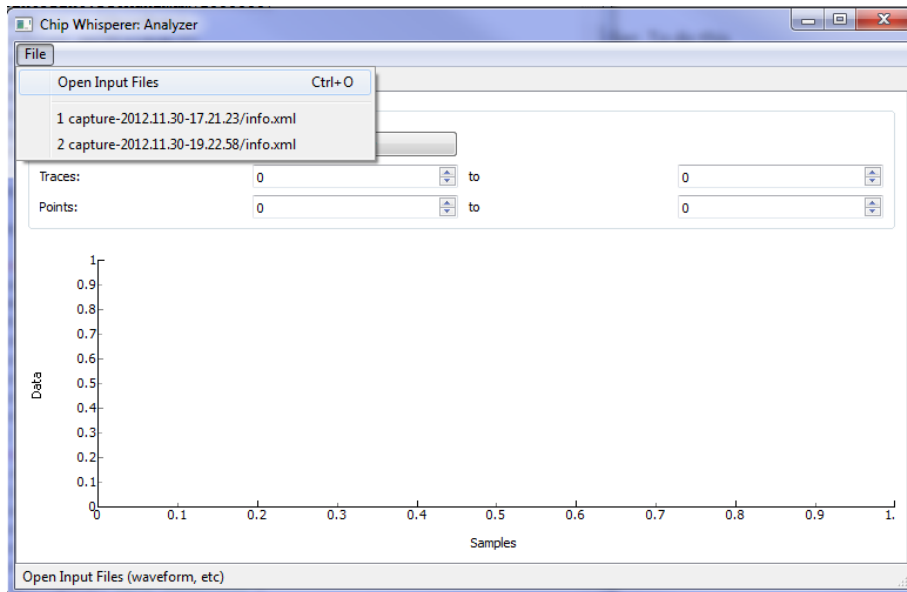
```
#####  
##### CHOOSE CRYPTO LIB #####  
#Uncomment the lines below for whichever crypto lib you'd like to attack  
  
#####  
#From http://avrcryptolib.das-labor.org, C Version  
#*This is the recommended first version to test*  
CRYPTO_LIB = avr-crypto-lib/aes  
SRC += aes_enc.c aes_keyschedule.c aes_sbox.c aes128_enc.c  
ASRC += gf256mul.S  
CDEFS += -DAVRCRYPTOLIB
```

Simply comment out the old crypto implementation and uncomment the crypto implementation you wish to use.

ChipWhisperer-Analyzer

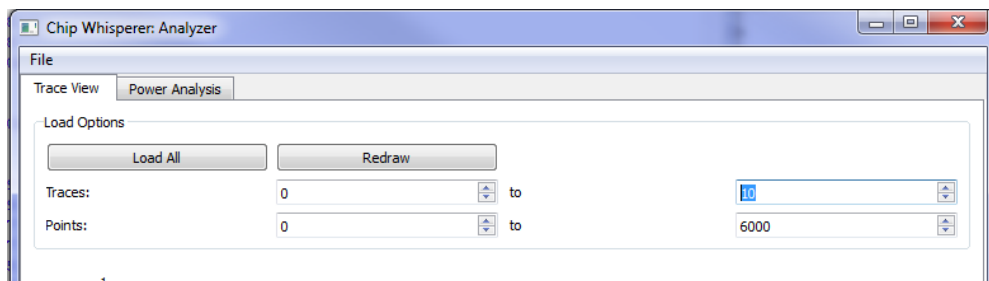
The Example Attack

The ChipWhisperer Analyzer program can load traces you created earlier. To do this run the program, and select the ‘Open Input Files’ option:

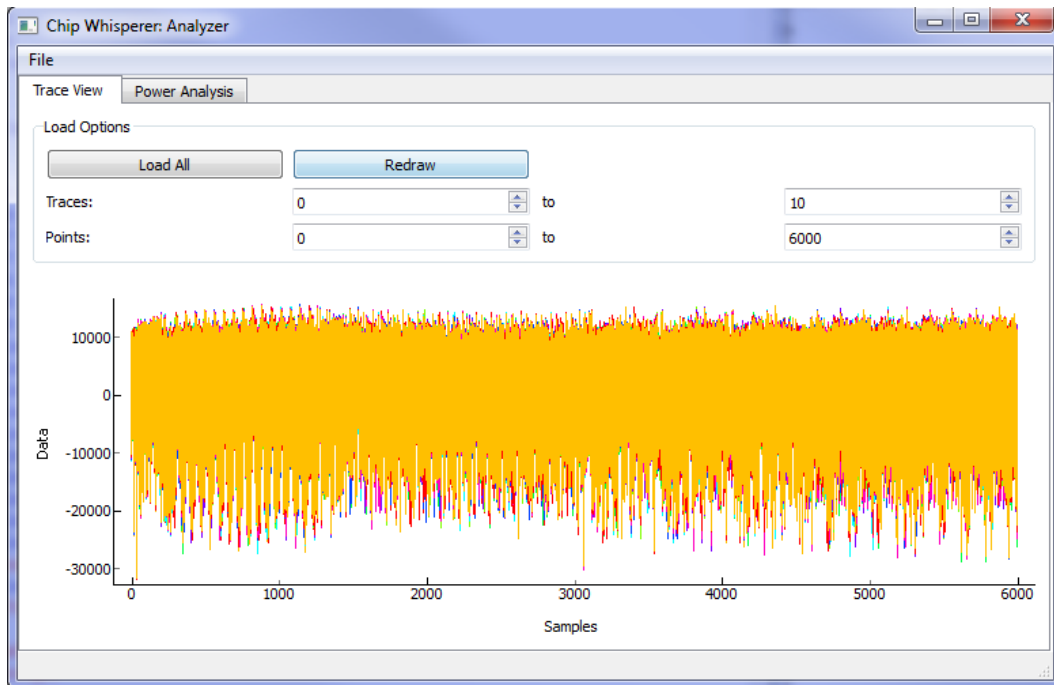


Find the info.xml file corresponding to the capture you just did, or any capture you want to analyze. When it opens it **does not** load the trace data into memory yet. To do this hit ‘Load All’ – the program will take a little bit (~30 seconds) to load everything into memory.

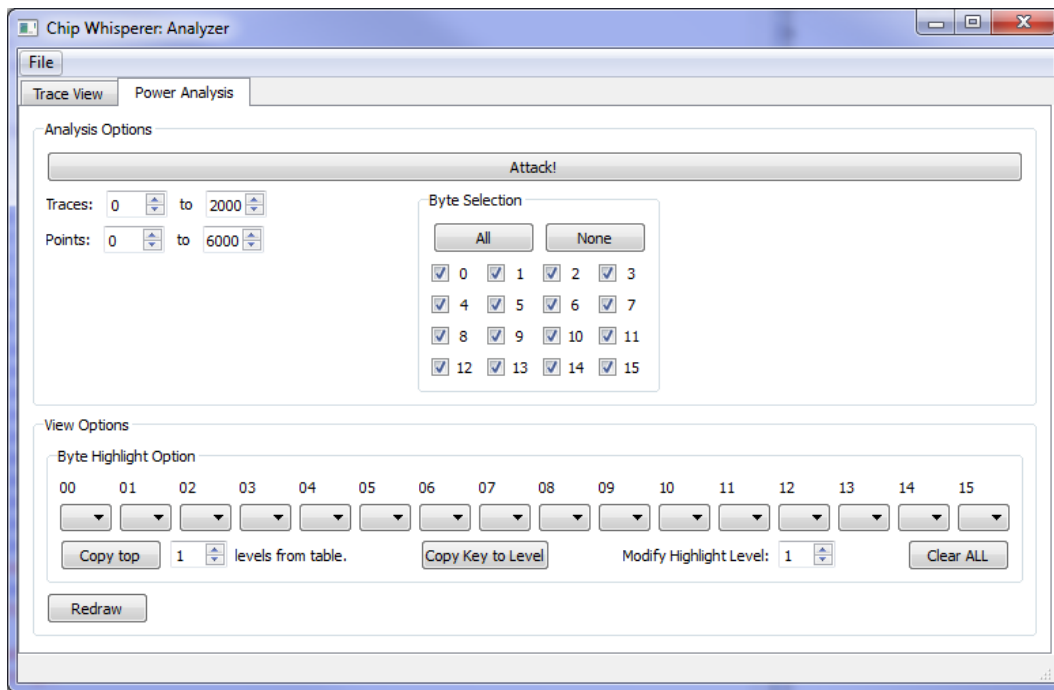
Then set the ‘Traces’ to say ‘0 to 10’ and hit ‘Redraw’.



This will plot the first 10 traces. You can zoom in on a section of it if you wish to look around.

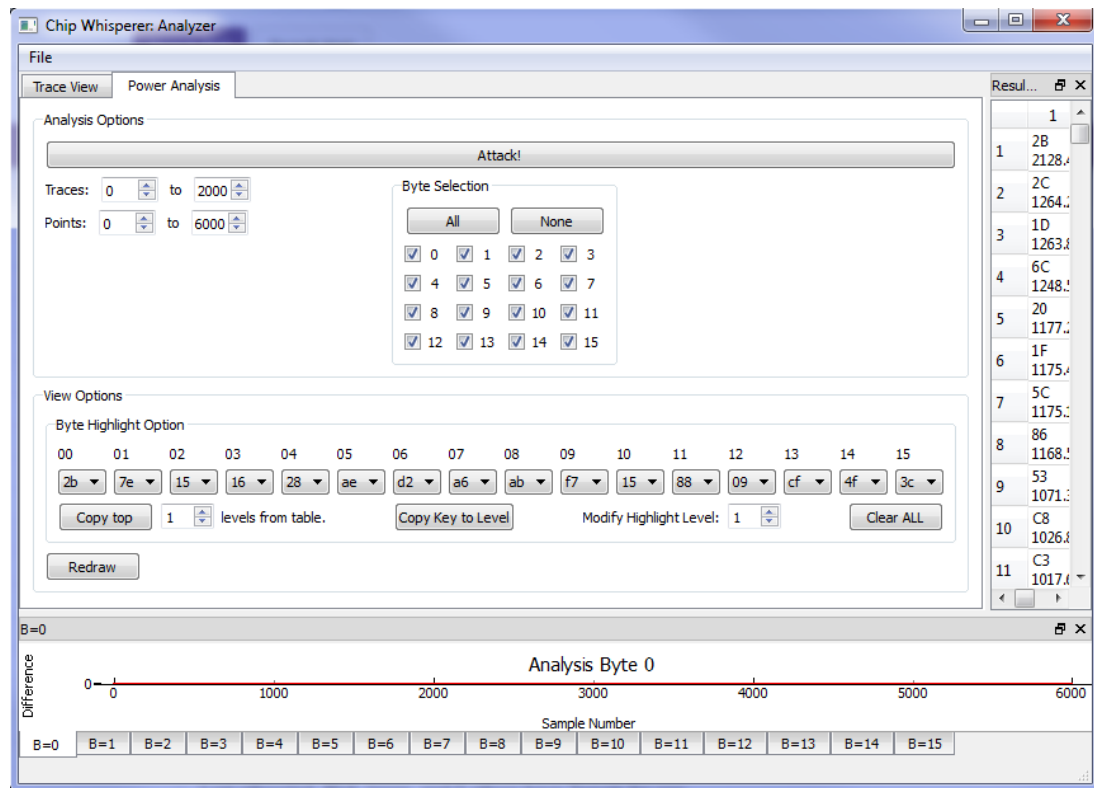


Now go over to the ‘Power Analysis’ tab.

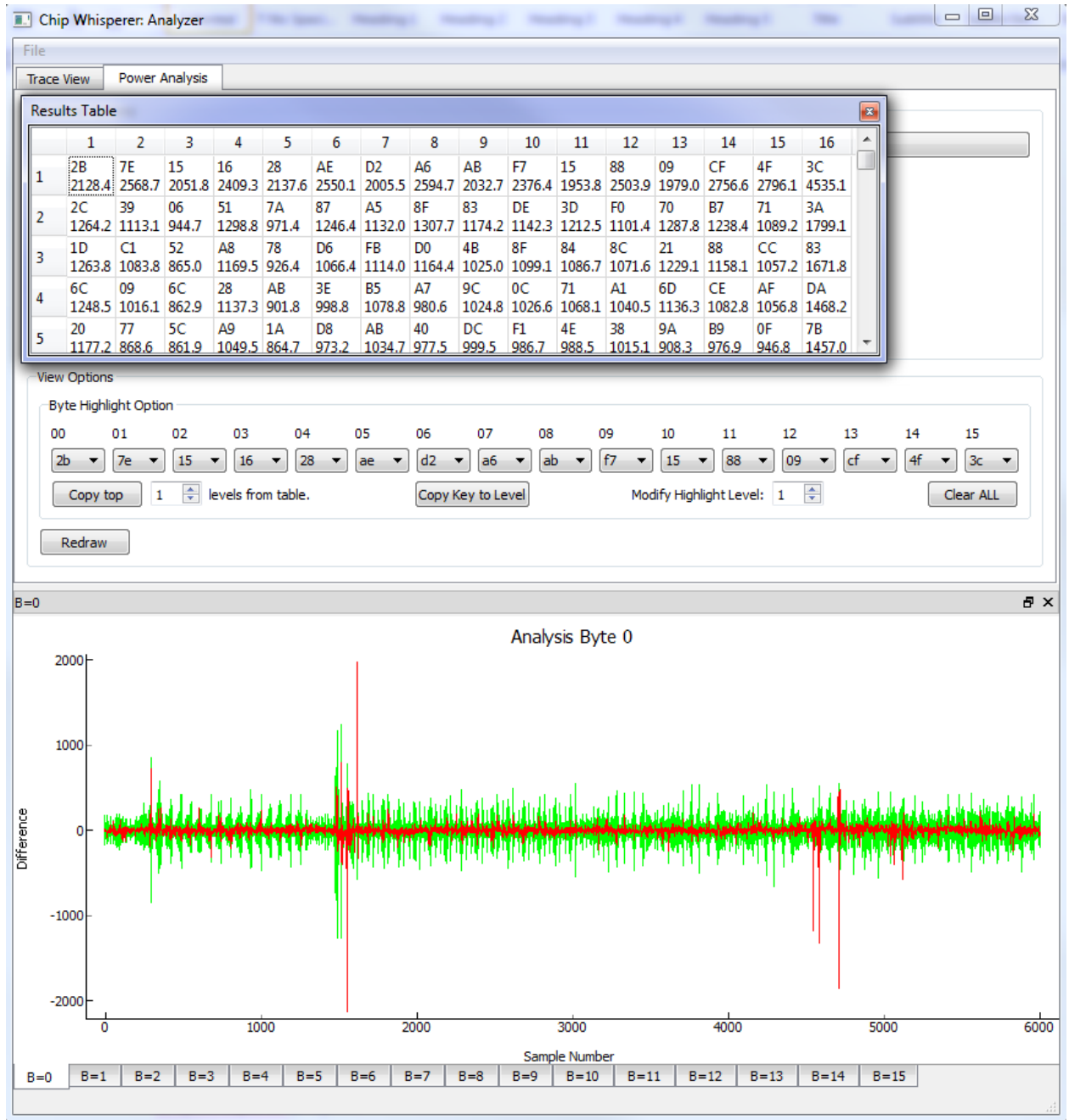


Hit ‘Copy Key to Level’ which will visualize where the correct key response occurred, and just hit ‘Attack’. This will take a while so if you want to attack a single byte first, in the ‘Byte Selection’ window select only 0 & 1 for example.

Once the attack finishes, you will have a few new windows. On the bottom will be all the differences from the DPA attack. On the right will be a table of what the top-ranked option for each byte is, along with rankings of all other possible options. It might look like this:



You'll want to peel off windows or resize things to get a better handle on what's happening! Something like this:



Note in the table it has successfully recovered the correct key! That is to say the tab-ranked key byte for each location IS the actual encryption key originally used.

For more details on the ChipWhisperer Analyzer check out chipwhisperer.com which contains the Wiki documentation, along with YouTube videos demonstrating it in action.

Further Attacks

I'm going to be adding more attacks. Check for updates at newae.com/blackhat or look at my ChipWhisperer project.

As an alternative, you might want to see what other people are doing. There are several good sources of attack software I know of:

1. OpenSCA at <http://www.cs.bris.ac.uk/home/eoswald/opensca.html>
2. Examples from the DPA Book at <http://www.dpabook.org>
3. The DPAContest have several references available at <http://www.dpacontest.org>

CD Resources

The CD has several things of interest:

1. Copies of the source code for the OpenADC control application & attack (also available at www.assembla.com/spaces/aesexplorer)
 - a. These run on Python. You'll need to install a bunch of dependencies:
 - i. Python 2.7
 - ii. PySerial
 - iii. PySide
 - iv. NumPY
 - v. PyqtgraphIf installing on Windows, I highly suggest to save some time and download the add-ons from <http://www.lfd.uci.edu/~gohlke/pythonlibs/> . Specific modules will require more add-ons such as pycard for Smart Cards. Typically when you try to run it you will see an error message printed about a missing package, just go and download the referenced package.
2. Eagle sources for the Differential Probe and some of the Target Boards
 - a. You'll need Eagle to open these, although PDF versions are provided too. See <http://www.cadsoftusa.com/> - there is a version which is available for free.
3. Datasheet for the OpenADC
4. Example Traces. These are in directories called 'capture-XXXX' describing the setup. You can attack them with DPA (or other scripts). The encryption key for all these traces is:
2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
5. Firmware Source Code

NOTE: The source code for the microcontrollers is ABSENT from the physical CD – but you must have downloaded this version from newae.com/blackhat, so you should have the firmware folder.

A Buyers Guide

This very briefly goes over what you might want to purchase.

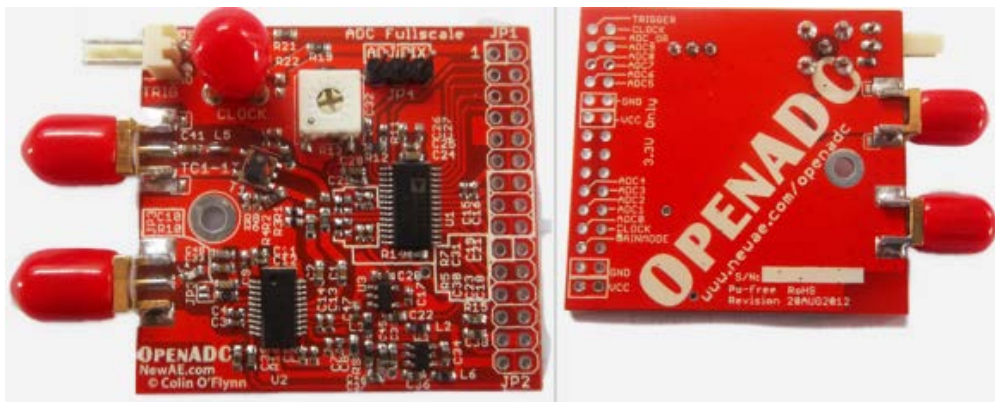
Scope

I designed the OpenADC for side-channel analysis. If you want to use it, you'll need a FPGA board which you can use with the OpenADC, I recommend the Spartan 6 LX9 Board from AVNET (part number AES-S6MB-LX9-G). But you can use almost any FPGA board you want with a little bit of effort, as all the Verilog source code is there.

The LX9 Board is \$90 and looks like this:



The OpenADC just plugs on top. Note with the current USB-serial interface the download speed is pretty limited, so it won't be remotely as fast as a proper oscilloscope. The OpenADC looks like this:




A nicer alternative is the ZTEX FPGA Modules, which have a wide range of FPGA sizes & enough I/O to run multiple capture boards. If you are reading the white paper from the CD I encourage you to check for updates, as I've probably got this working already if you check at chipwhisperer.com

If you want a generic scope, there are lots of nice USB ones. Expect to spend over \$1000 USD depending how fancy you want.

One USB oscilloscope is the Cleverscope, which has an option to add an external sample clock input (option CS810). The cleverscope is the lowest-cost oscilloscope with this synchronous sample option:



TiePie makes scopes with a higher sampling rate, but more limited synchronous ability. The reference clock must be around 10 MHz, and the TiePie generates higher sample frequencies based on this. For example the HS5 is a high-speed scope (500 MS/s):



Handyscope HS5 the unbeatable USB oscilloscope

The new 500 MS/s 14-bit dual channel High Resolution Oscilloscope with function generator.

This powerful high speed USB oscilloscope combines fast sampling up to 500 MS/s with high resolutions of 12, 14 and 16 bit, a large memory of 64 MSamples and an extremely accurate built-in 30 MHz 14 bit arbitrary waveform generator. The oscilloscope supports continuous streaming measurements up to 20 MS/s and can be synchronized with other oscilloscopes to form a multi channel combined instrument with synchronized timebase. The flexibility and quality that the **Handyscope HS5** offers is unparalleled by any other oscilloscope in its class.

[Buy](#) Base price: **€930.00**
[Datasheet](#) [Manual](#) [Contact](#)

[Use the oscilloscope comparison to compare the Handyscope HS5 to other USB oscilloscopes:](#)
[TiePieSCOPE HS805](#) | [Handyprobe HP3](#) | [Handyscope HS4](#) | [Handyscope HS4 DIFF](#) | [Handyscope HS3](#)

Another alternative is the PicoScope brand, which is often used in these projects. They have a range of sample rates available (up to 5 GS/s of publication), with a fairly low cost. Some of their oscilloscopes (PS6000 series) have a generic ‘Clock Input’ port available too! Check with Pico Technology to be sure this feature can be used as desired.




Magnetic Field Probe

Search for terms such as 'SMA Male RF405', 'SMA Male Semi-Rigid', 'SMA Male Semi-Flex'.

Suggested just to look on EBay:

Back to search results | Consumer Electronics > Radio Communication > Parts & Accessories > Coax, Cables & Connectors



6in RG405

Sell one like this

6in SMA male to male plug Pigtail flexible Cable RG405

Item condition: **New**

Quantity: More than 10 available

Price: **US \$3.50**
Approximately C \$3.51 [Buy It Now](#)

[Add to Watch list](#)

Shipping: **FREE** Economy Intl Shipping | [See all details](#)
See details about international shipping here. @

Delivery: **Varies for items shipped from an international location** @
Seller ships within 1 day after receiving cleared payment.

Returns: **14 day money back or item exchange, buyer pays return shipping** | [Read details](#)

Coverage: **Pay with PayPal** and your full purchase price is covered | [See terms](#)

Top-rated seller
maya.yoooyoo (13191) ★

99.9% Positive feedback

- ✓ Consistently receives highest buyers' ratings
- ✓ Ships items quickly
- ✓ Has earned a track record of excellent service

[Ask a question](#)
[Save this seller](#)
[See other items](#)

Visit store: [Adapter Connector Pigtail cable](#)

Other item info

Item number: 160599161824

Item location: **Hong Kong, Hong Kong**

Ships to: **United States, Europe, Canada, Australia**
[See exclusions](#)

Payments: **PayPal**
[See payment information](#)

History: **110 sold**

[Print](#) | [Report item](#)

[Description](#) | [Shipping and payments](#) [Share](#)

Seller assumes all responsibility for this listing
Last updated on Apr 24, 2012 07:46:43 EDT [View all revisions](#)

Use self-fusing tape & Polyimide Tape for insulation purposes:

Photos



Fusion Pro Self-fusing Black Silicone Tape
Product #67-0023-6
[See more Tapes, Glue & Sealants or compare](#)

Reg. \$5.99

Be the first to [write a review](#) | [Ask a Question](#)

[Add To My List](#) | [Add Rate Alerts](#)



Polyimide Heat Resistant/High Temperature Adhesive Tape (8MM*33M/260-C)

SKU: 21351

Qty: 1

Price: **US \$ 0.19** [View Price](#)

Shipping: [Free Shipping](#) To the US & Canada

Delivery: [Tracked Ship in 1-2 US Day](#)

[Add to Cart](#) | [Add to Wish List](#) | [Print Item](#) | [Report Error](#)

<http://dx.com/polyimide-heat-resistant-high-temperature-adhesive-tape-8mm-33m-260-c-21351?item=2>

Or try coating entire thing in rubber/plastic, such as with:

Plasti Dip

[Pin It](#)



Low Noise Amplifier

Easiest thing is to buy a LNA from somewhere like Mini-Circuits, such as ZFL-500LN or ZFL-1000LN. Combine that with a variable attenuator if you need less amplification (output is too big).

As a hack you can turn the voltage down on the LNA. As the supply voltage goes down the LNA gain goes down due to biasing – you will also see some performance degradation, so this can't be done too much!

If you want to build your own, see the Mini-Circuits application example at <http://www.minicircuits.com/pcb/WTB-411-8+ P02.pdf>, which is the schematic I based my LNA on.

Other companies make small LNA chips too, you can search e.g. Digikey.

AVR Stuff

In North America, the easiest place to buy AVR is at Digikey. You can also buy the programmers here too. Your options are:

- ATAVRISP2 – Atmel ISP MK2 (\$35). Only usable for programming.
- ATAVRDRAGON – Atmel Dragon (\$50). Usable for programming and debugging *most* devices. Doesn't come in a case (bare PCB).

- ATJTAGICE3 – Atmel JTAG 3 (\$99). Usable for programming & debugging all AVR/XMega devices, comes in a nice case. You probably want this if you might be doing some debugging.

SmartCard Stuff

You can buy a normal ‘commercial’ reader online. The specific one I used (SCR335) was purchased from eBay, several of them will come up when searching that.

There are many sources online for the ‘FunCard AtMega163’ used here, called ‘ATMega Card’ also. You will need a way to program it – you can purchase specific programmers (see Infinity USB SmartCard Programmer, or DuoLabs Dynamite +Plus). Note you can program these cards with an AVR programmer if you connect the proper pins to the normal 6-pin ISP header and provide a clock.

You might also want to look into using the BASIC Cards, which are more readily available. At the time of this white paper submission I hadn’t completed the discussion of the BASIC card module. Again check the updated white paper for details of this.

Alternatively, you can use the SASEBO-W board, see below.

The SASEBO Project

A company called SAKURA makes a number of useful boards for side-channel analysis. While they aren’t exactly ‘cheapskate’, they are very high performance. For instance the SASEBO-W contains a Spartan 6 LX150 on which you can run attack algorithms, and the OpenADC directly interfaces to this board:



There are a variety of other targets of interest. See <http://www.morita-tech.co.jp/SAKURA/en/hardware.html>

Where to Go From Here

You’ve reached the end! So where to go for more information?

First, there is the original paper in the field, available at <http://www.cryptography.com/public/pdf/DPA.pdf> which a shorter version at <http://www.cryptography.com/public/pdf/DPATechInfo.pdf>.

The ‘DPA Book’ is described at <http://dpabook.org/> which contains a lot of useful information, going from the bare basics onward. So if you are just starting might be an easy way forward.

A good introductory crypto book is available at <http://www.crypto-textbook.com> too.

You can also explore on Google Scholar (or similar) different papers in this field. Papers describing more advanced attacks are readily available.

There is lots of experimentation that can be done too. Before going out and purchasing lots of stuff, some of this you can do on your own. There are some example traces on the CD, and more will be posted to my website.

References

- [1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” *Advances in Cryptology—CRYPTO’99*, pp. 789–789, 1999. Available: <http://www.cryptography.com/public/pdf/DPA.pdf>
- [2] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*, vol. 31. Springer-Verlag New York Inc, 2007. <http://www.dpabook.org>
- [3] C. O’Flynn and Z. Chen, “A Case Study of Side-Channel Analysis using Decoupling Capacitor Power Measurement with the OpenADC,” *Foundation and Practices of Security*, 2012. Available: <http://www.newae.com/tiki-index.php?page=Articles>
- [4] D. C. Smith, “Signal and noise measurement techniques using magnetic field probes,” *Electromagnetic Compatibility, 1999 IEEE International Symposium on*, vol. 1, pp. 559–563, 1999.
- [5] E. De Mulder, “Electromagnetic Techniques and Probes for Side-Channel Analysis on Cryptographic Devices,” 2010.
- [6] I. Kizhvatov, “Side channel analysis of AVR XMEGA crypto engine,” *Proceedings of the 4th Workshop on Embedded Systems Security*, p. 8, 2009.

Revision History

24NOV2013 – Update for Blackhat EU. New Chipwhisperer project too, which updates the capture and now includes a graphical attack program.

30NOV2012 – Updated version, includes information about firmware in target examples, updates figures to reflect this firmware, add SmartCard interface

15NOV2012 – Original version included on Blackhat Abu Dhabi CD-ROM