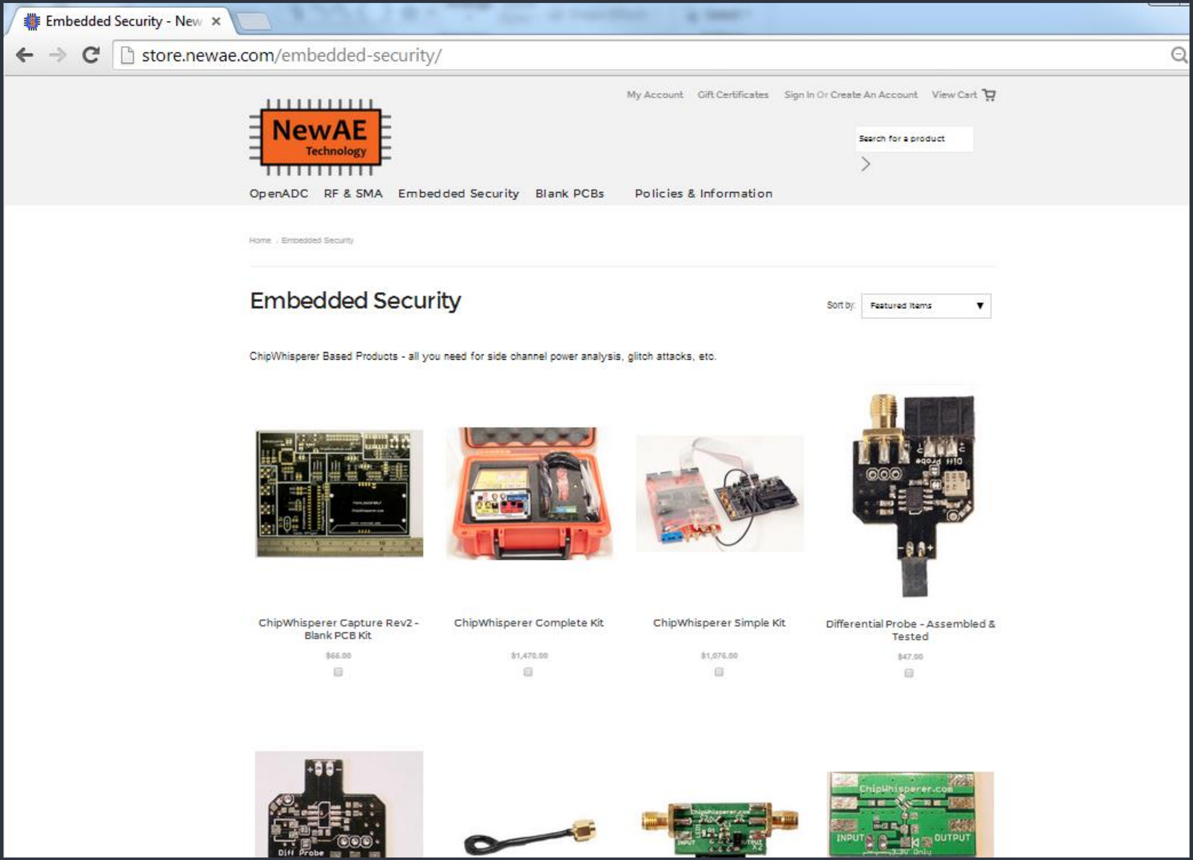
Power Analysis + Clock Glitching

Colin O'Flynn RECON 2014 – Montreal, Canada

OBJECTIVES

* LEARN ABOUT S.C.A
* SEE SOME DEMOS
* BUILD YOUR OWN S.C.A.
* BUILD SOME GLITCK HW

COMMERCIAL?



Embedded Security - New

store.newae.com/embedded-security/

My Account Gift Certificates Sign In Or Create An Account View Cart

NewAE Technology

Search for a product

OpenADC RF & SMA Embedded Security Blank PCBs Policies & Information

Home : Embedded Security

Embedded Security

Sort by: Featured Items

ChipWhisperer Based Products - all you need for side channel power analysis, glitch attacks, etc.

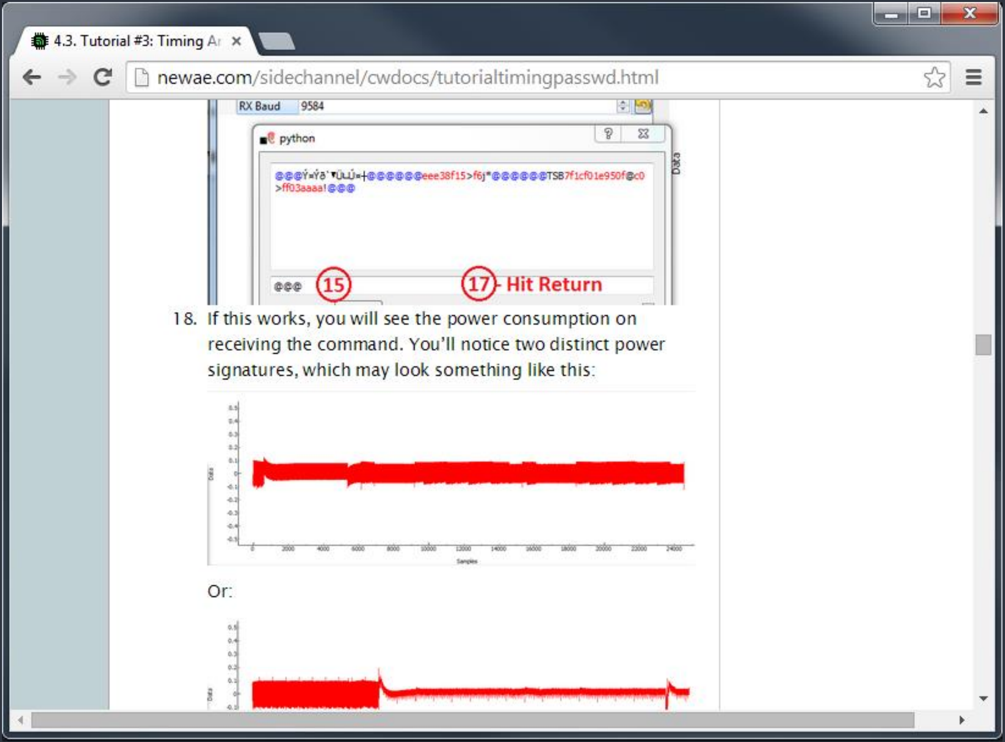
ChipWhisperer Capture Rev2 - Blank PCB Kit
$65.00

ChipWhisperer Complete Kit
$1,470.00

ChipWhisperer Simple Kit
$1,076.00

Differential Probe - Assembled & Tested
$47.00

COMMERCIAL?



4.3. Tutorial #3: Timing Ar
newae.com/sidechannel/cwdocs/tutorialtimingpasswd.html
RX Baud 9584
python
15
17 - Hit Return
Data
Samples

```
@@@Ý¤Ýð`▼ÜLLÜ¤┼@@@@@@eee38f15>f6j*@@@@@@TSB7f1cf01e950f@c0
>ff03aaaa!@@@
```

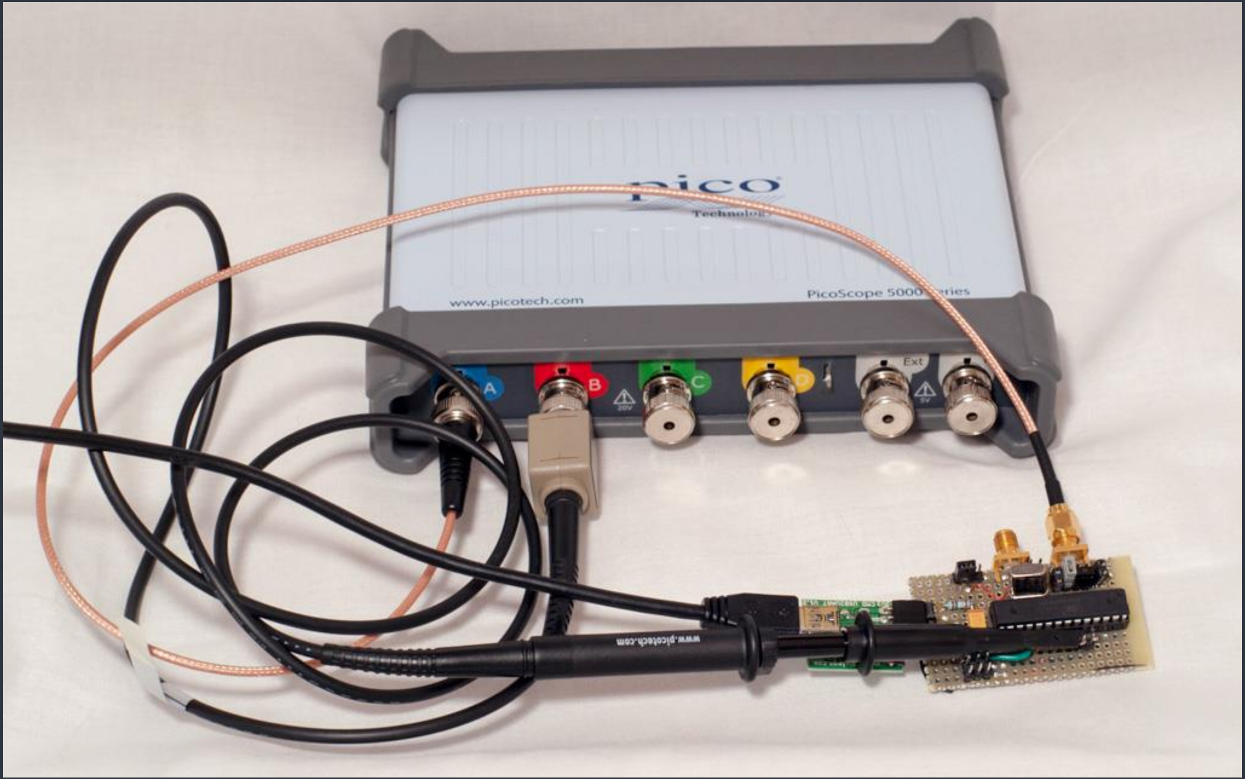
```
@@@
```

18. If this works, you will see the power consumption on receiving the command. You'll notice two distinct power signatures, which may look something like this:

Or:

http://newae.com/sidechannel/cwdocs/

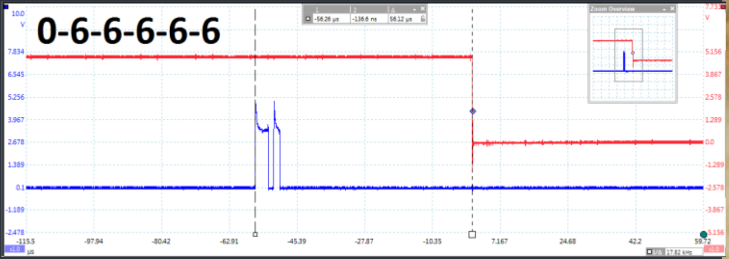
COMMERCIAL?

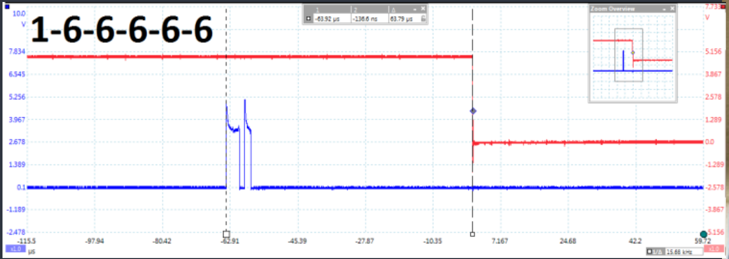

pico Technology
www.picotech.com
PicoScope 5000 Series
A
B
C
D
Ext
www.picotech.com

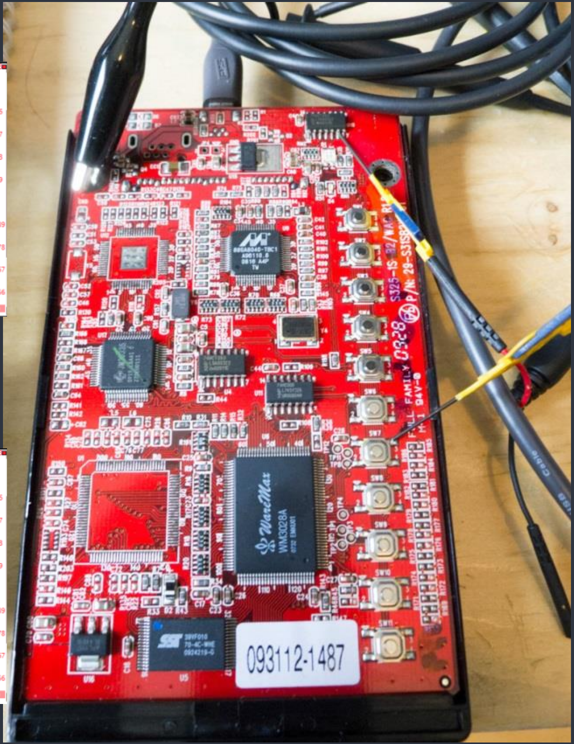
SPRINGFIELD SAUSAGE FACTORY
KIDS, COME SEE HOW IT'S MADE!
Global

SIDE CHANNEL

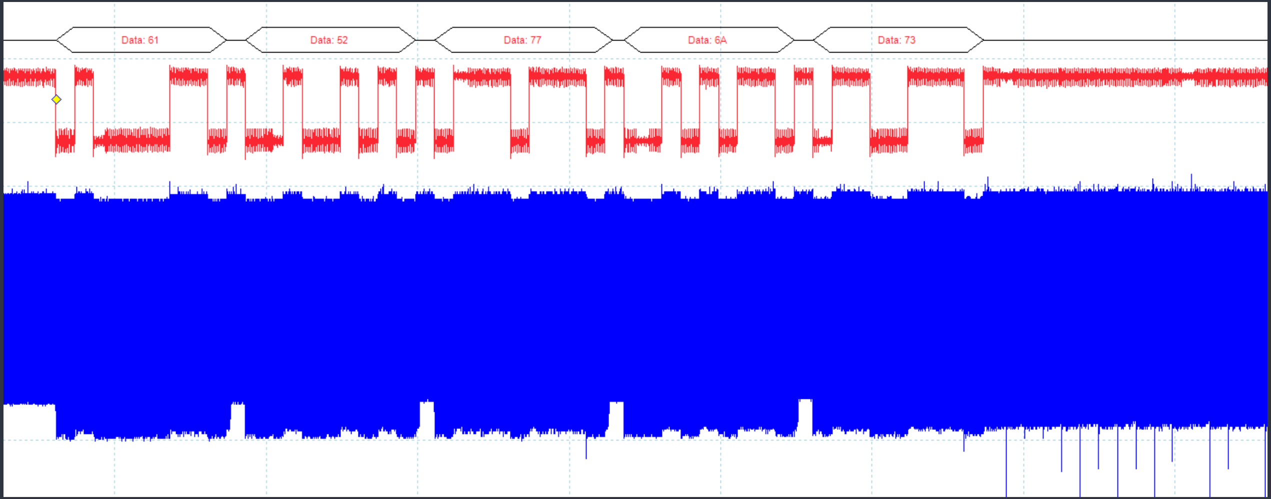
TIMING ATTACK



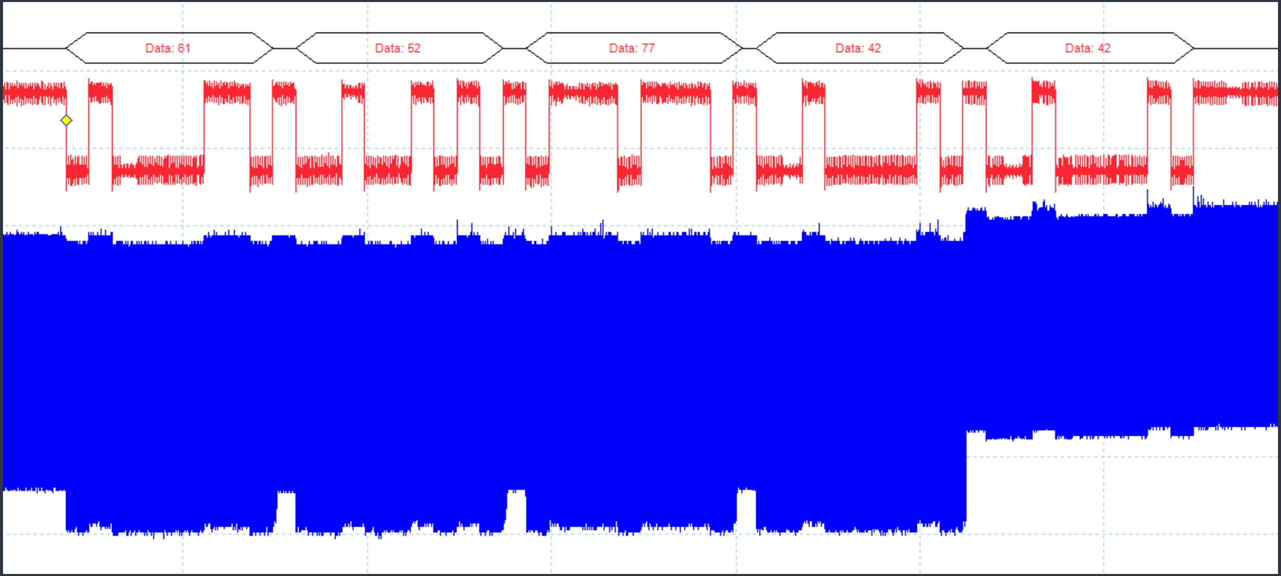




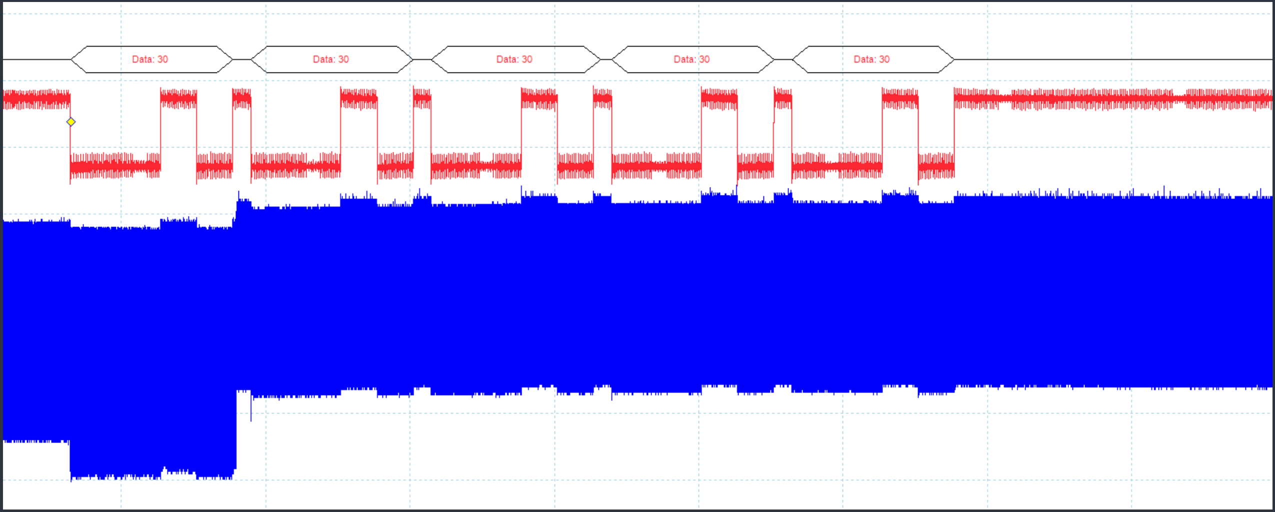
+ Power

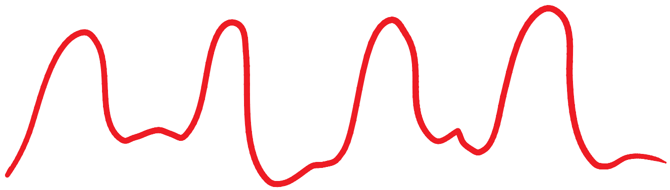


+ Power



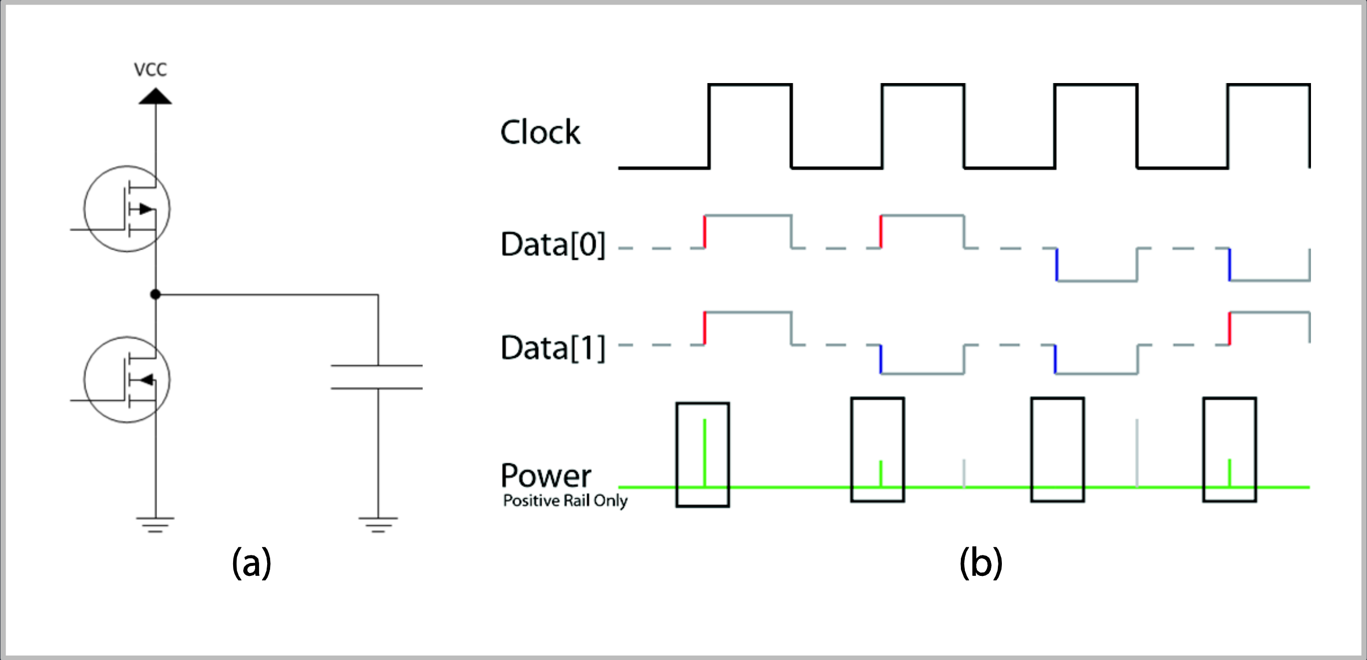
+ POWER



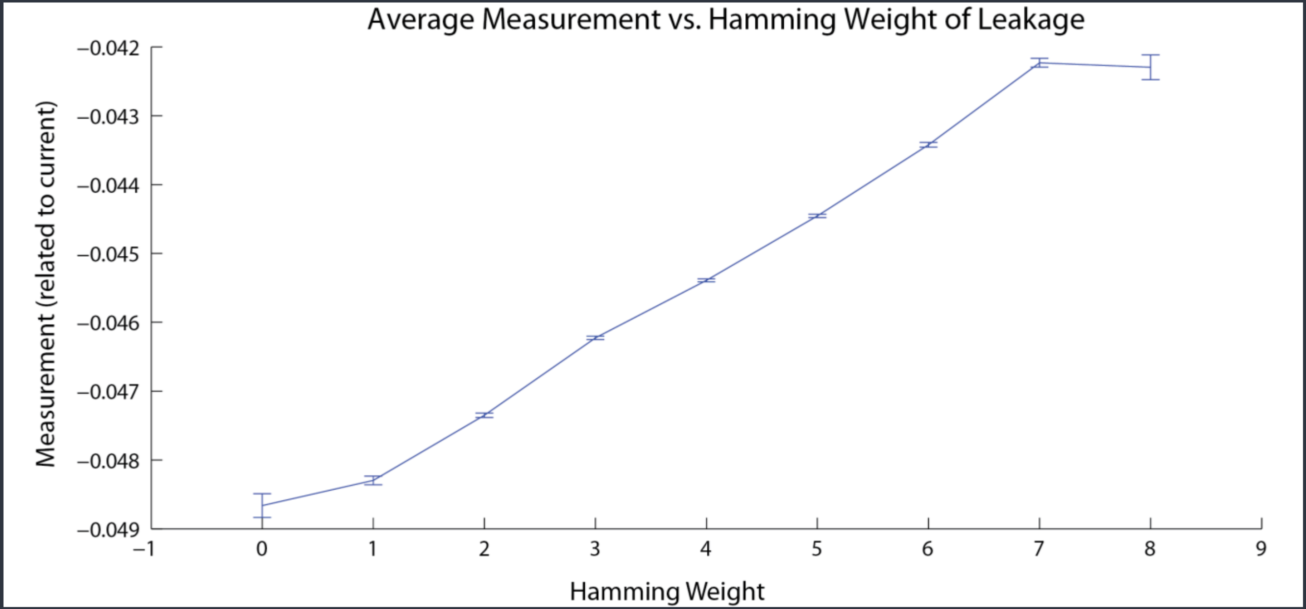


POWER ANALYSIS

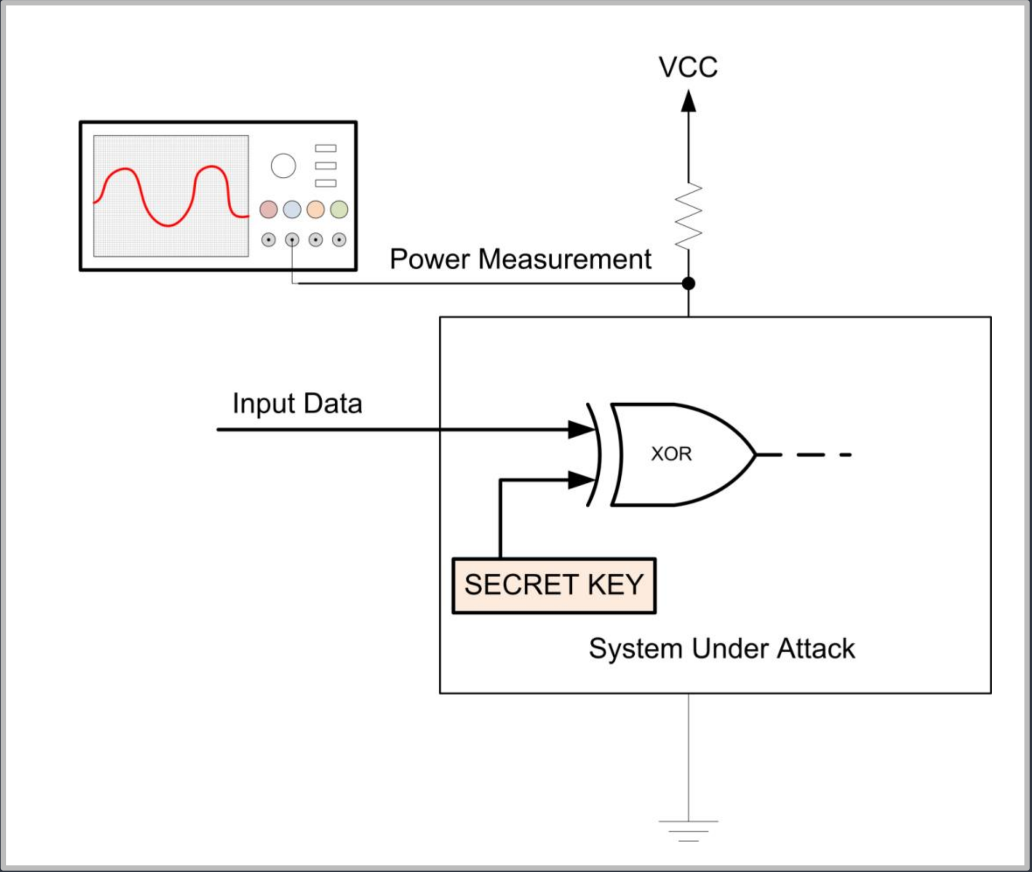
LEAKAGE



ATMEGA328P


Average Measurement vs. Hamming Weight of Leakage
Measurement (related to current)
−0.042
−0.043
−0.044
−0.045
−0.046
−0.047
−0.048
−0.049
−1
0
1
2
3
4
5
6
7
8
9
Hamming Weight

AN EXAMPLE



VCC
Power Measurement
Input Data
XOR
SECRET KEY
System Under Attack

XOR

Assume user is ‘encrypting’ a 1-byte piece of data by XORing with a 1-byte secret key (EF), and we cannot observe output of XOR. This becomes:

$$\begin{array}{l} 88 \oplus \mathrm{EF} = 67 \\ 56 \oplus \mathrm{EF} = \mathrm{B9} \\ 32 \oplus \mathrm{EF} = \mathrm{DD} \\ A6 \oplus \mathrm{EF} = 49 \\ 35 \oplus \mathrm{EF} = \mathrm{DA} \end{array} \xrightarrow{\mathrm{HW}} \left.\begin{array}{c} 5 \\ 5 \\ 6 \\ 3 \\ 5 \end{array}\right\} \text{observations}$$

XOR

Marking the unknowns with KK or ?:

$$\begin{array}{l} 88 \oplus \text{KK} = ? \\ 56 \oplus \text{KK} = ? \\ 32 \oplus \text{KK} = ? \\ A6 \oplus \text{KK} = ? \\ 35 \oplus \text{KK} = ? \end{array} \xrightarrow{\text{HW}} \left.\begin{array}{c} 5 \\ 5 \\ 6 \\ 3 \\ 5 \end{array}\right\} \text{observations}$$

How to find?

Guess & Check!

XOR

Guess KK = 0x00

$$\left.\begin{array}{l}
88 \oplus 00 = 88 \\
56 \oplus 00 = 56 \\
32 \oplus 00 = 32 \\
A6 \oplus 00 = A6 \\
35 \oplus 00 = 35
\end{array}\right. \xrightarrow{HW} \left.\begin{array}{c}
2 \\
4 \\
3 \\
4 \\
4
\end{array}\right\} \text{Hypothesis}$$

XOR

Guess KK = 0x01

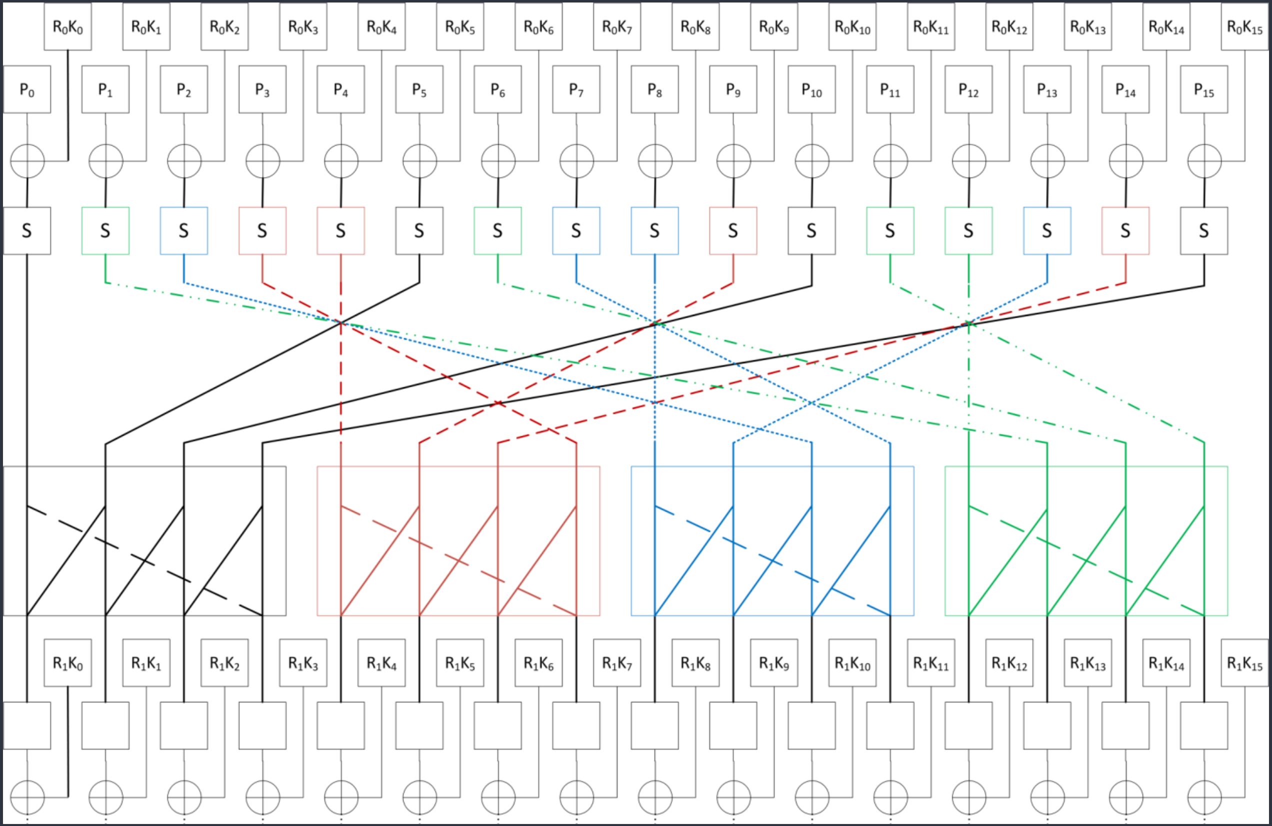
$$\begin{array}{l} 88 \oplus 01 = 89 \\ 56 \oplus 01 = 57 \\ 32 \oplus 01 = 33 \\ A6 \oplus 01 = A7 \\ 35 \oplus 01 = 34 \end{array} \xrightarrow{\text{HW}} \left.\begin{array}{c} 3 \\ 5 \\ 4 \\ 5 \\ 3 \end{array}\right\} \text{Hypothesis}$$

XOR

Guess KK = 0xEF

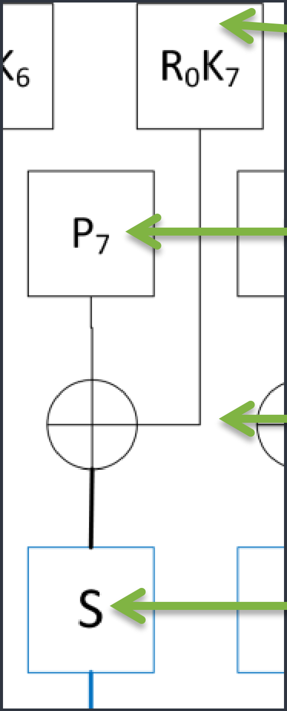
$$\begin{array}{l} 88 \oplus EF = 67 \\ 56 \oplus EF = B9 \\ 32 \oplus EF = DD \\ A6 \oplus EF = 49 \\ 35 \oplus EF = DA \end{array} \xrightarrow{\text{HW}} \left.\begin{array}{c} 5 \\ 5 \\ 6 \\ 3 \\ 5 \end{array}\right\} \text{Hypothesis}$$

WTF - HOW IS THAT GOOD?



R_0K_0 R_0K_1 R_0K_2 R_0K_3 R_0K_4 R_0K_5 R_0K_6 R_0K_7 R_0K_8 R_0K_9 R_0K_{10} R_0K_{11} R_0K_{12} R_0K_{13} R_0K_{14} R_0K_{15}
P_0 P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15}
S S S S S S S S S S S S S S S S
R_1K_0 R_1K_1 R_1K_2 R_1K_3 R_1K_4 R_1K_5 R_1K_6 R_1K_7 R_1K_8 R_1K_9 R_1K_{10} R_1K_{11} R_1K_{12} R_1K_{13} R_1K_{14} R_1K_{15}

WTF - HOW IS THAT GOOD?



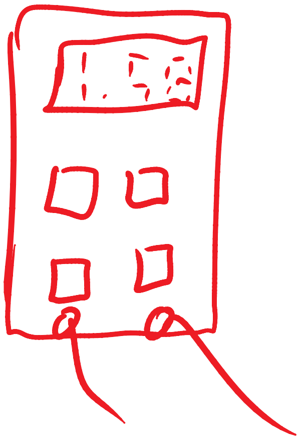
1-Byte of Key (Subkey)

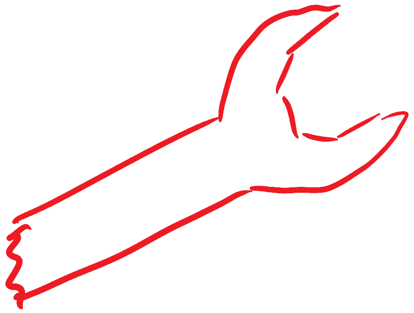
1-Byte of Input (Plaintext)

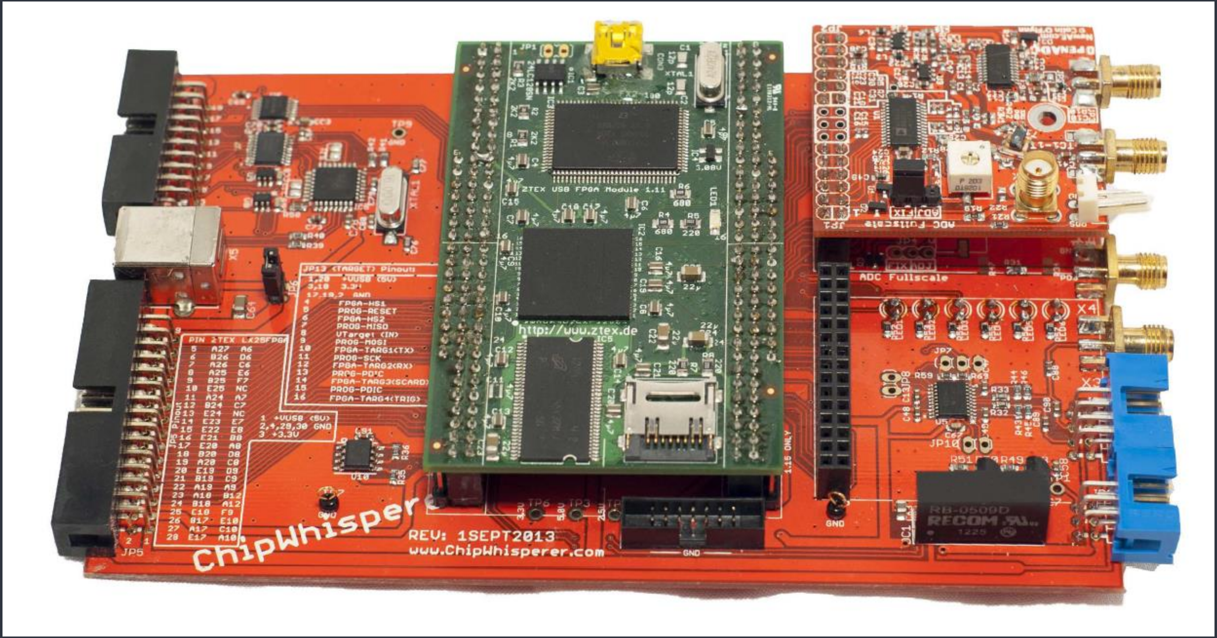
Bitwise XOR

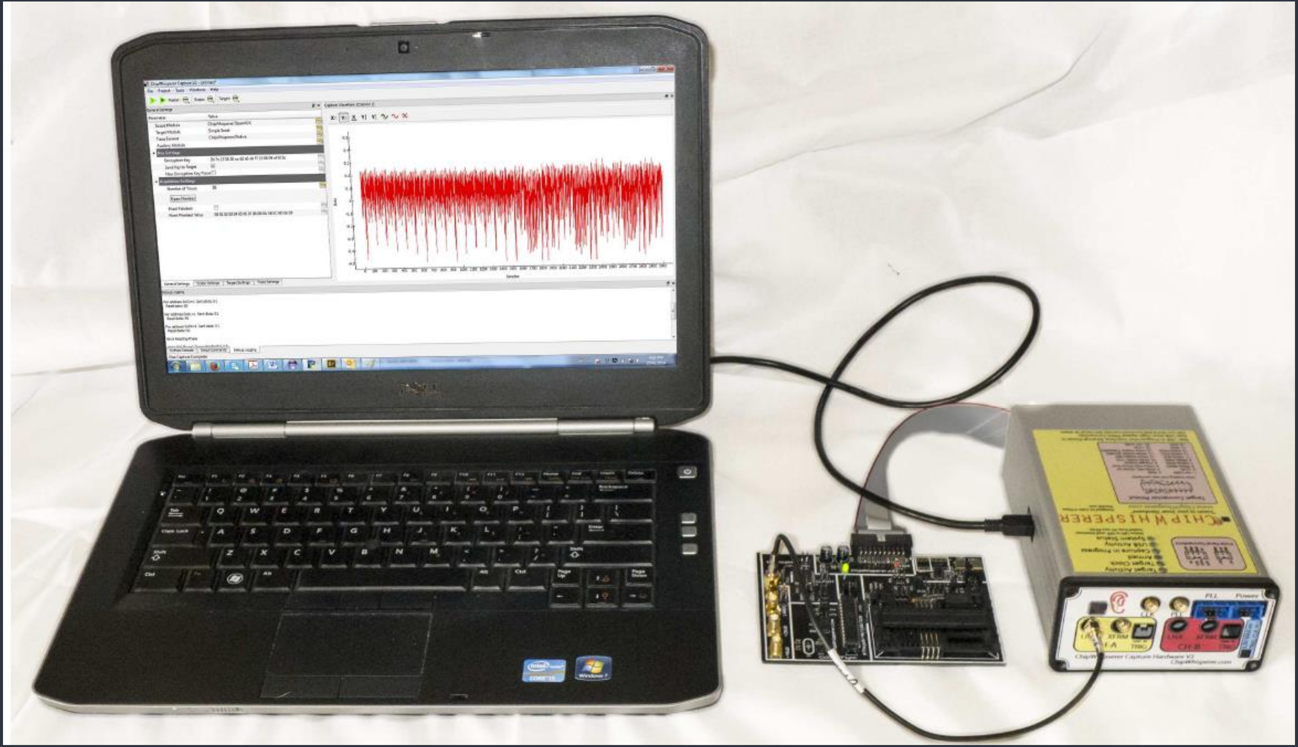
Substitution-Box (Lookup Table)

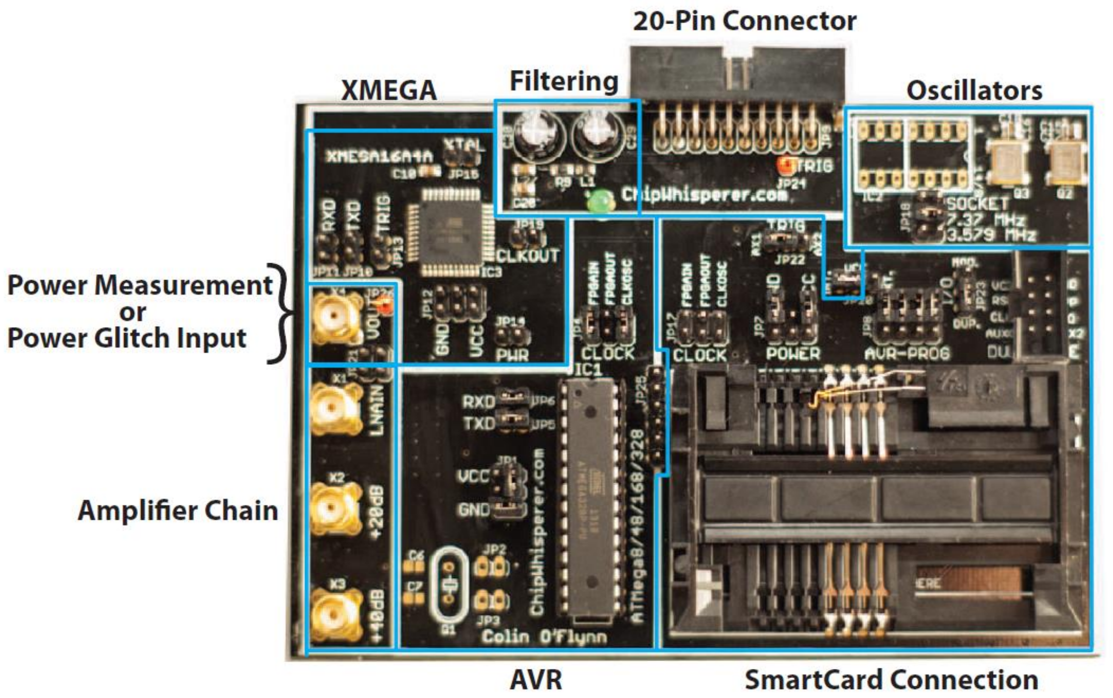
Tools

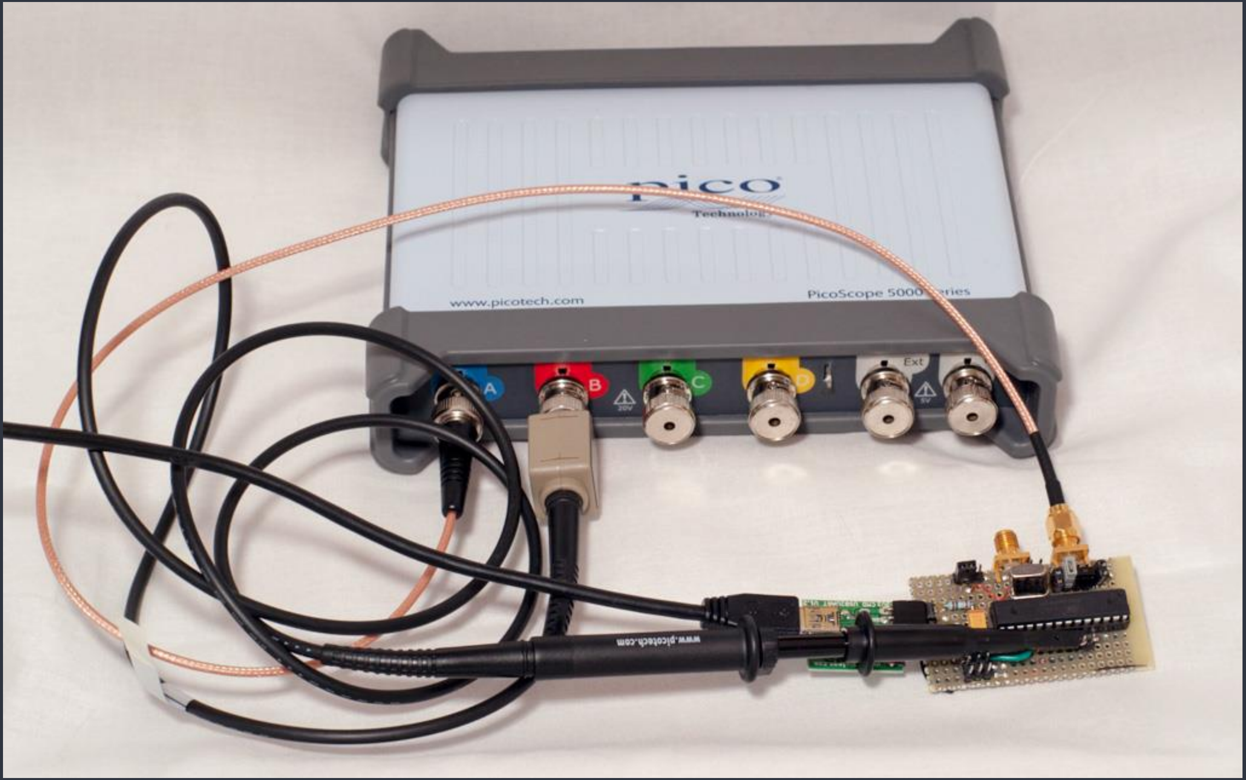




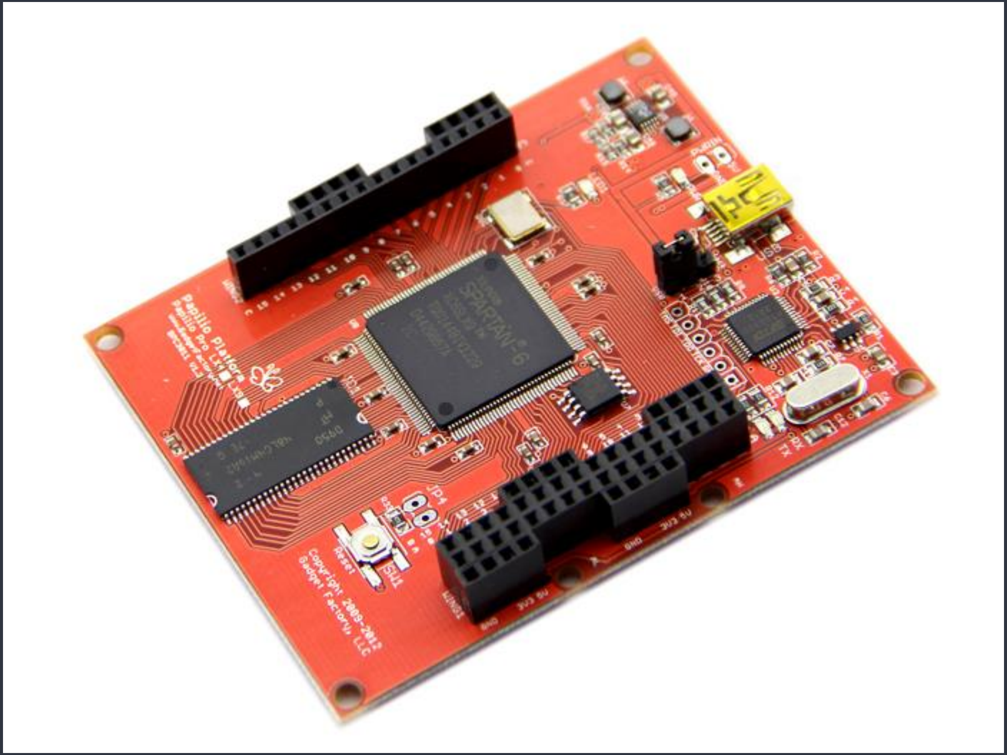

ChipWhisperer
REV: 1SEPT2013
www.ChipWhisperer.com
ZTEX USB FPGA Module 1.11
http://www.ztex.de
ADC Fullscale
RECOM

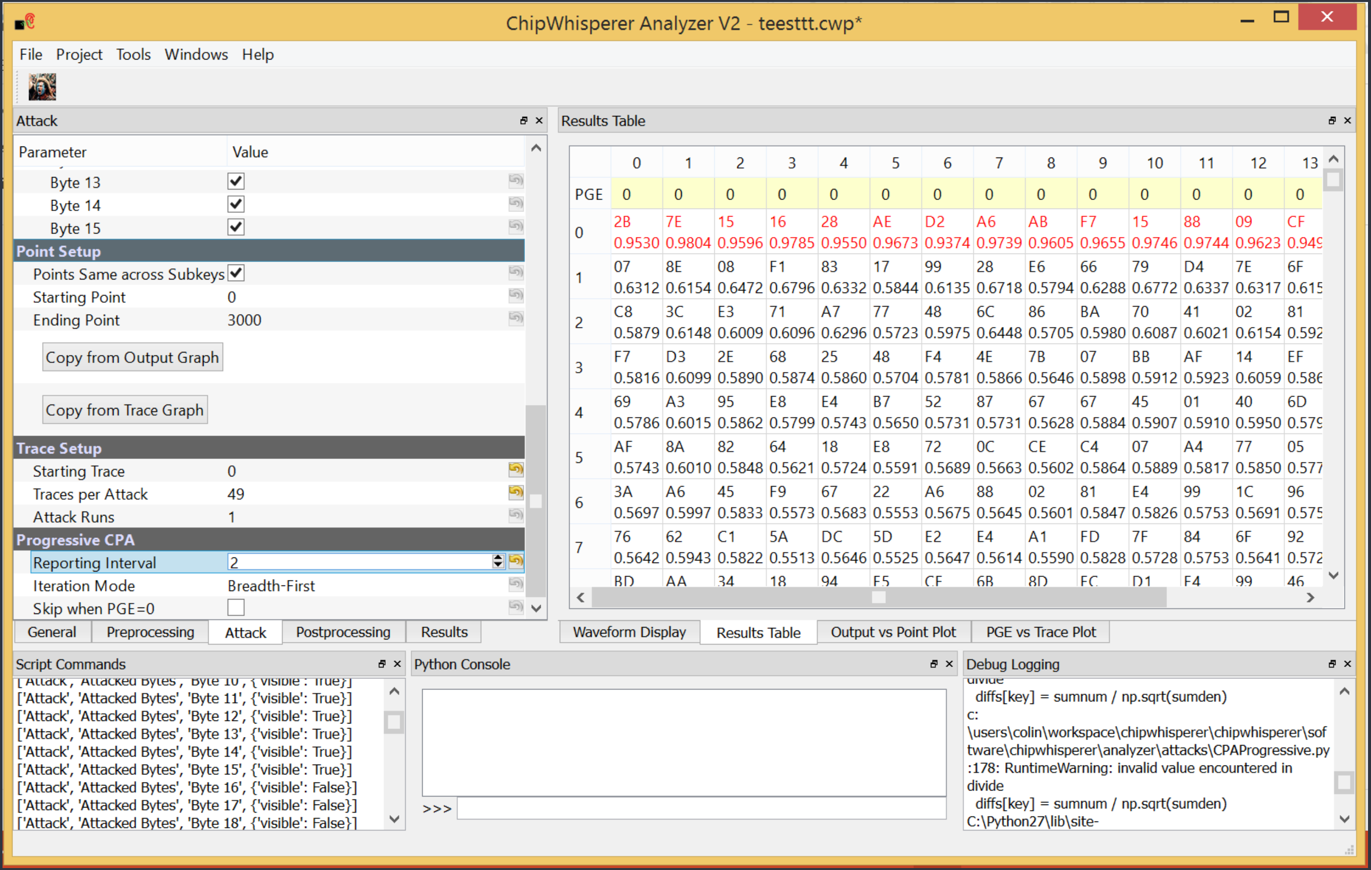

CHIPWHISPERER
ChipWhisperer Capture Hardware V2
ChipWhisperer.com
CH-A
CH-B
PLL
Power


20-Pin Connector
XMEGA
Filtering
Oscillators
Power Measurement or Power Glitch Input
Amplifier Chain
AVR
SmartCard Connection


pico
Technology
www.picotech.com
PicoScope 5000 Series
A
B
C
D
Ext
20V
5V
www.picotech.com

SPARTAN 6


SPARTAN-6
Papilio Platform
Papilio Pro LX9
Copyright 2009-2012
Gadget Factory, LLC
Reset



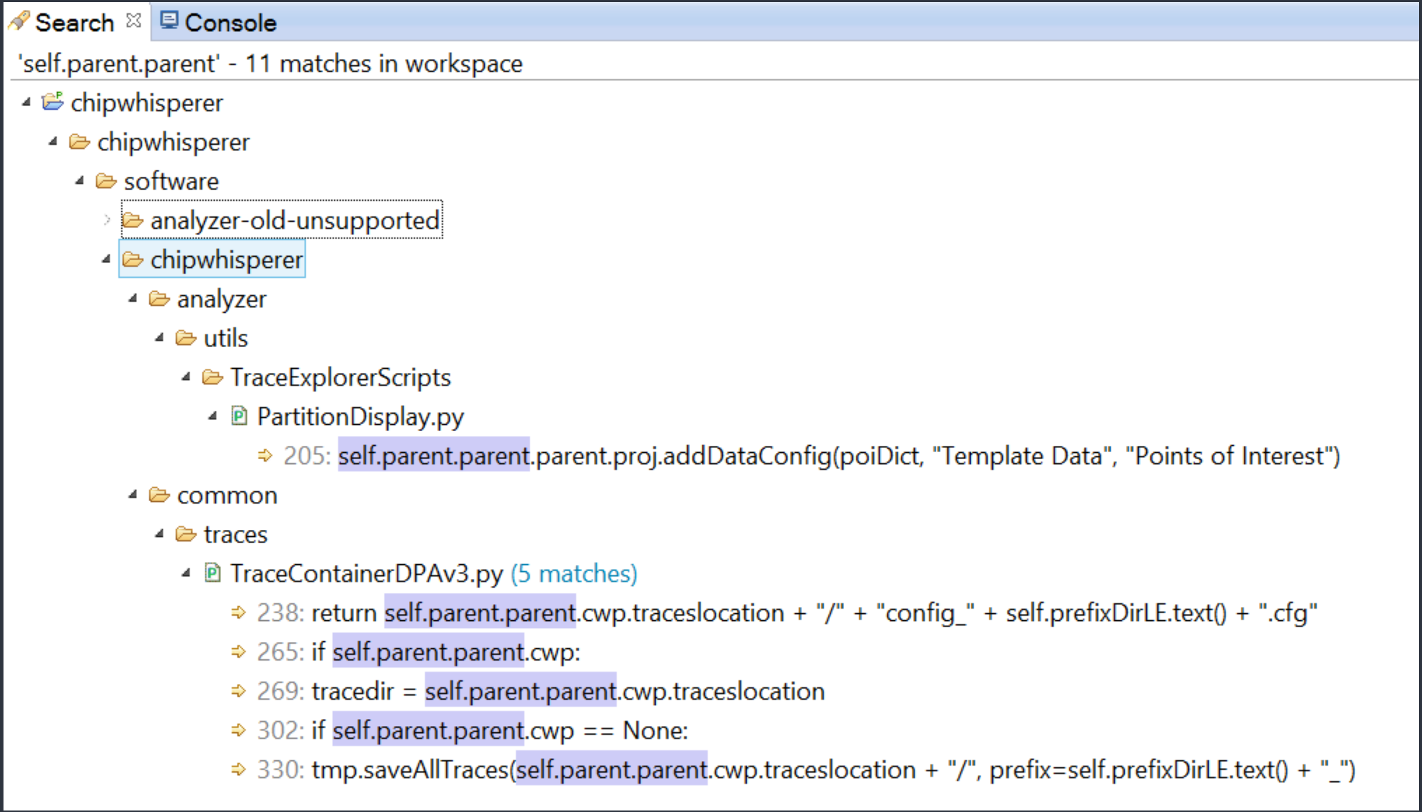
Parameter	Value
Byte 13	✓
Byte 14	✓
Byte 15	✓
Point Setup	
Points Same across Subkeys	✓
Starting Point	0
Ending Point	3000
Trace Setup	
Starting Trace	0
Traces per Attack	49
Attack Runs	1
Progressive CPA	
Reporting Interval	2
Iteration Mode	Breadth-First
Skip when PGE=0	☐

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
PGE	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2B 0.9530	7E 0.9804	15 0.9596	16 0.9785	28 0.9550	AE 0.9673	D2 0.9374	A6 0.9739	AB 0.9605	F7 0.9655	15 0.9746	88 0.9744	09 0.9623	CF 0.949
1	07 0.6312	8E 0.6154	08 0.6472	F1 0.6796	83 0.6332	17 0.5844	99 0.6135	28 0.6718	E6 0.5794	66 0.6288	79 0.6772	D4 0.6337	7E 0.6317	6F 0.615
2	C8 0.5879	3C 0.6148	E3 0.6009	71 0.6096	A7 0.6296	77 0.5723	48 0.5975	6C 0.6448	86 0.5705	BA 0.5980	70 0.6087	41 0.6021	02 0.6154	81 0.592
3	F7 0.5816	D3 0.6099	2E 0.5890	68 0.5874	25 0.5860	48 0.5704	F4 0.5781	4E 0.5866	7B 0.5646	07 0.5898	BB 0.5912	AF 0.5923	14 0.6059	EF 0.586
4	69 0.5786	A3 0.6015	95 0.5862	E8 0.5799	E4 0.5743	B7 0.5650	52 0.5731	87 0.5731	67 0.5628	67 0.5884	45 0.5907	01 0.5910	40 0.5950	6D 0.579
5	AF 0.5743	8A 0.6010	82 0.5848	64 0.5621	18 0.5724	E8 0.5591	72 0.5689	0C 0.5663	CE 0.5602	C4 0.5864	07 0.5889	A4 0.5817	77 0.5850	05 0.577
6	3A 0.5697	A6 0.5997	45 0.5833	F9 0.5573	67 0.5683	22 0.5553	A6 0.5675	88 0.5645	02 0.5601	81 0.5847	E4 0.5826	99 0.5753	1C 0.5691	96 0.575
7	76 0.5642	62 0.5943	C1 0.5822	5A 0.5513	DC 0.5646	5D 0.5525	E2 0.5647	E4 0.5614	A1 0.5590	FD 0.5828	7F 0.5728	84 0.5753	6F 0.5641	92 0.572
	BD	AA	34	18	94	F5	CF	6B	8D	FC	D1	F4	99	46

```
['Attack', 'Attacked Bytes', 'Byte 10', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 11', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 12', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 13', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 14', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 15', {'visible': True}]
['Attack', 'Attacked Bytes', 'Byte 16', {'visible': False}]
['Attack', 'Attacked Bytes', 'Byte 17', {'visible': False}]
['Attack', 'Attacked Bytes', 'Byte 18', {'visible': False}]
```

```
>>>
```

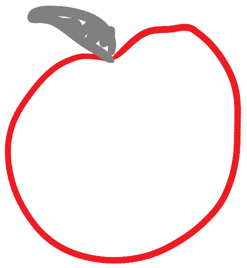
```
divide
  diffs[key] = sumnum / np.sqrt(sumden)
c:
\users\colin\workspace\chipwhisperer\chipwhisperer\sof
tware\chipwhisperer\analyzer\attacks\CPAProgressive.py
:178: RuntimeWarning: invalid value encountered in
divide
  diffs[key] = sumnum / np.sqrt(sumden)
C:\Python27\lib\site-
```



'self.parent.parent' - 11 matches in workspace

```
chipwhisperer
  chipwhisperer
    software
      analyzer-old-unsupported
      chipwhisperer
        analyzer
          utils
            TraceExplorerScripts
              PartitionDisplay.py
                205: self.parent.parent.parent.proj.addDataConfig(poiDict, "Template Data", "Points of Interest")
        common
          traces
            TraceContainerDPAv3.py (5 matches)
              238: return self.parent.parent.cwp.traceslocation + "/" + "config_" + self.prefixDirLE.text() + ".cfg"
              265: if self.parent.parent.cwp:
              269: tracedir = self.parent.parent.cwp.traceslocation
              302: if self.parent.parent.cwp == None:
              330: tmp.saveAllTraces(self.parent.parent.cwp.traceslocation + "/", prefix=self.prefixDirLE.text() + "_")
```

SOMETHING THAT'S REAL



Atmel AVR231: AES Bootloader

8-bit Atmel Microcontrollers

Application Note

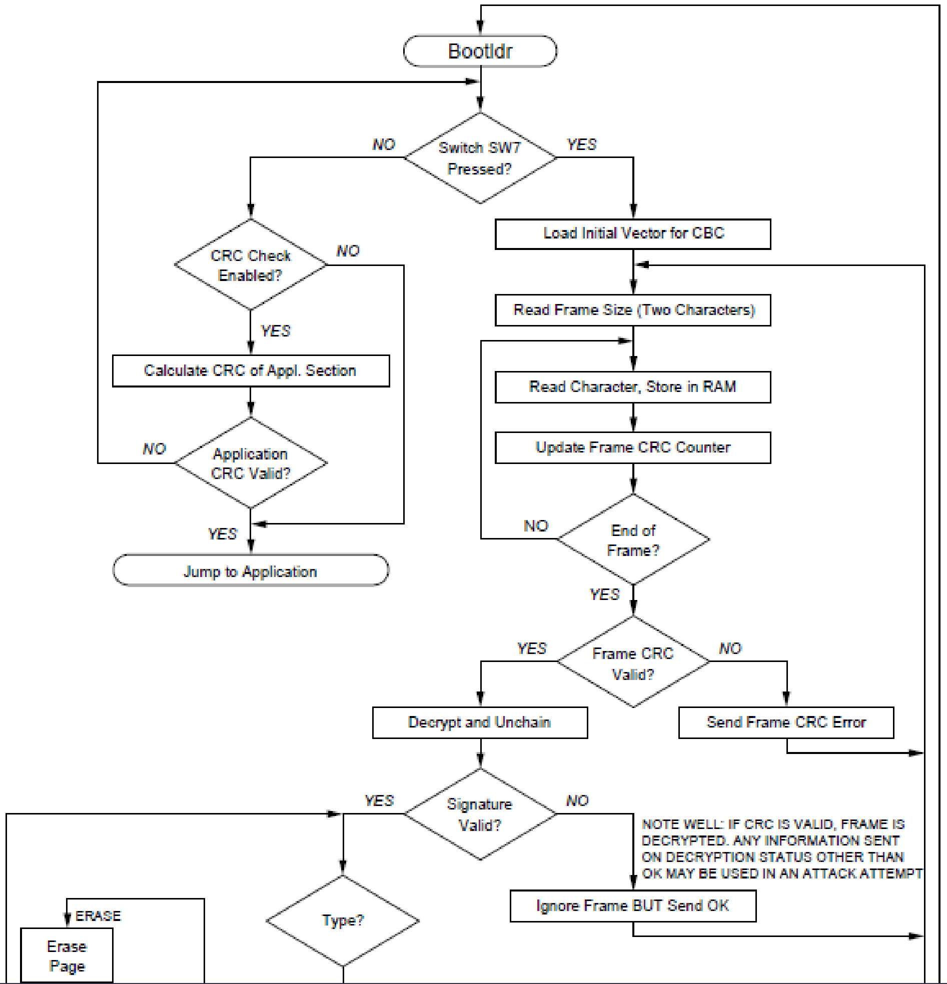
Features

- Fits Atmel® AVR® Microcontrollers with bootloader capabilities and at least 1kB SRAM
- Enables secure transfer of firmware and sensitive data to an AVR based application
- Includes easy-to-use configurable example applications:
  - Encrypting binary files and data
  - Creating target bootloaders
  - Downloading encrypted files to target
- Implements the Advanced Encryption Standard (AES):
  - 128-, 192-, and 256-bit keys
- AES Bootloader fits into 2kB
- Typical update times of a 64kB application, 115200 baud, 3.69MHz target frequency:
  - AES128: 27 seconds
  - AES192: 30 seconds
  - AES256: 33 seconds

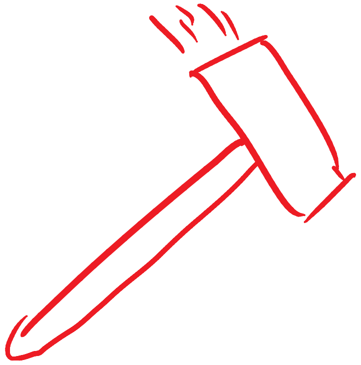
1 Introduction

This application note describes how firmware can be updated securely on AVR microcontrollers with bootloader capabilities. The method uses the Advanced Encryption Standard (AES) to encrypt the firmware.

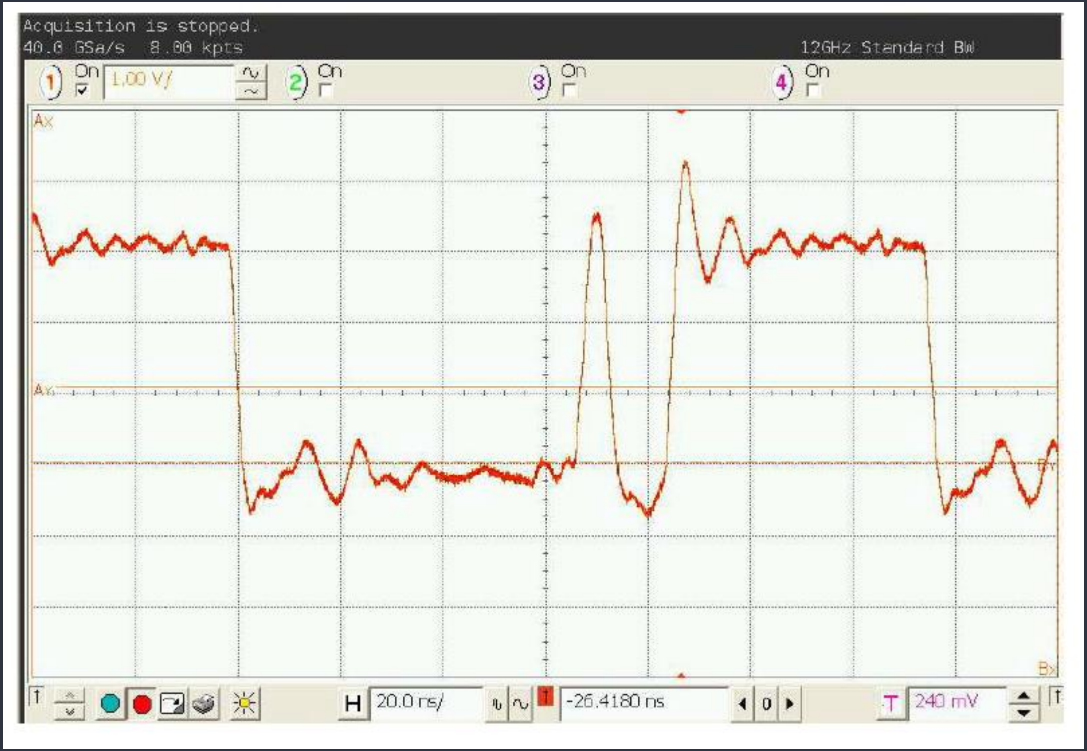
Figure 4-5. Flowchart for the AVR bootloader.


Bootldr
Switch SW7 Pressed?
NO
YES
CRC Check Enabled?
Load Initial Vector for CBC
Calculate CRC of Appl. Section
Read Frame Size (Two Characters)
Read Character, Store in RAM
Update Frame CRC Counter
Application CRC Valid?
End of Frame?
Jump to Application
Frame CRC Valid?
Decrypt and Unchain
Send Frame CRC Error
Signature Valid?
NOTE WELL: IF CRC IS VALID, FRAME IS DECRYPTED. ANY INFORMATION SENT ON DECRYPTION STATUS OTHER THAN OK MAY BE USED IN AN ATTACK ATTEMPT
ERASE
Erase Page
Type?
Ignore Frame BUT Send OK

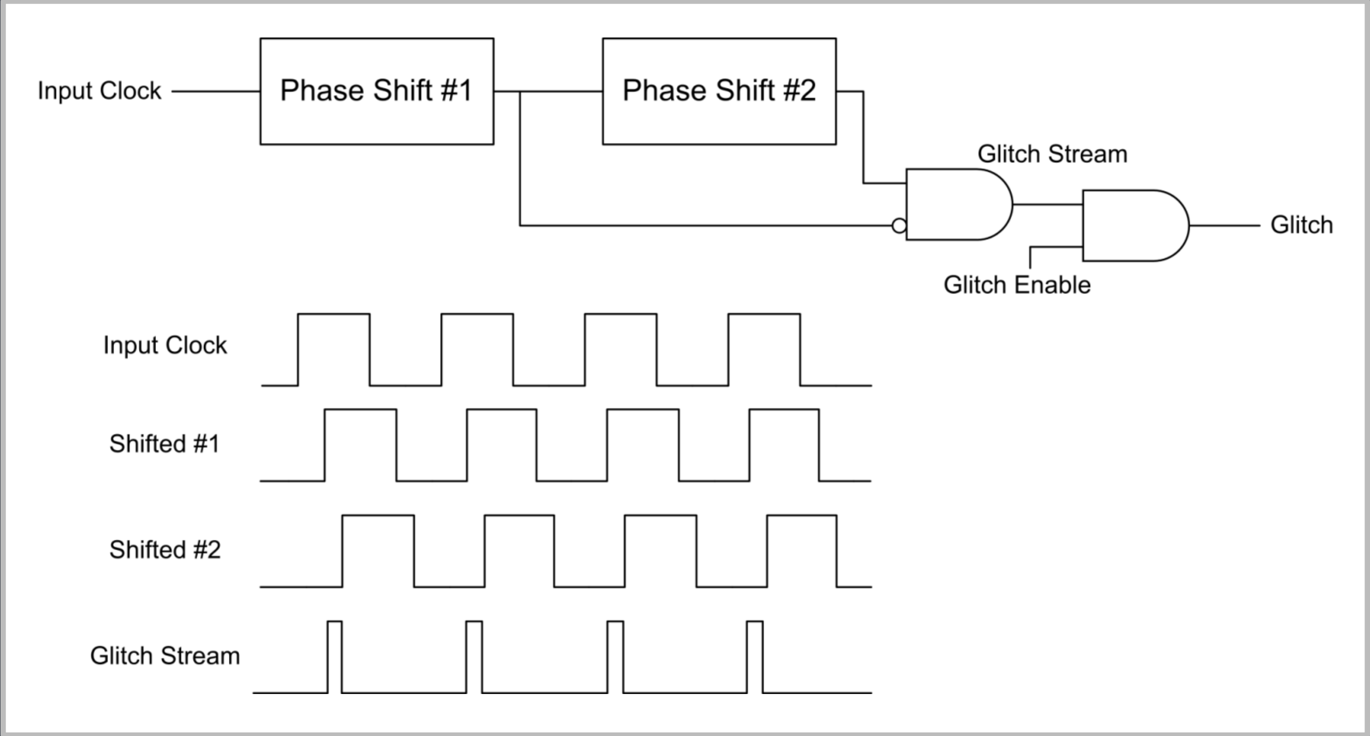
GLITCHING



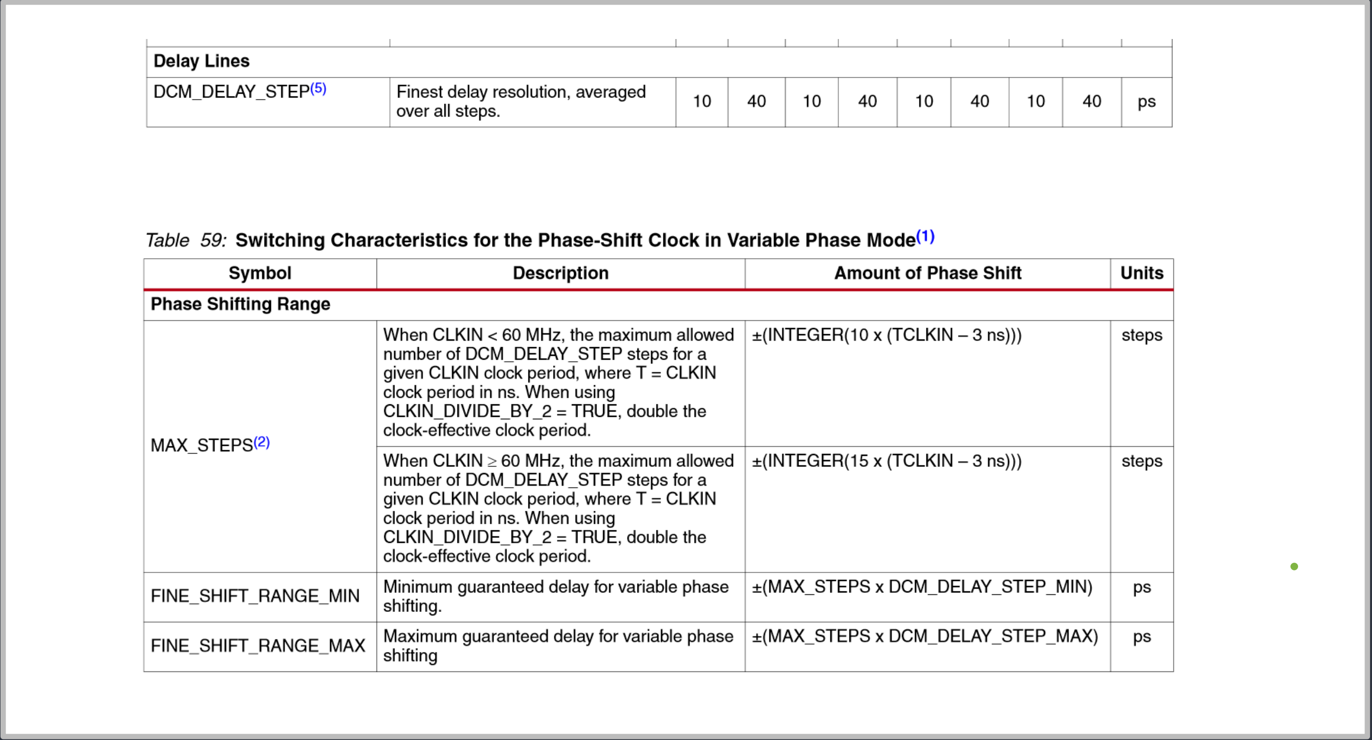
CLOCK GLITCH



GLITCH GENERATOR



A MAJOR HEADACHE

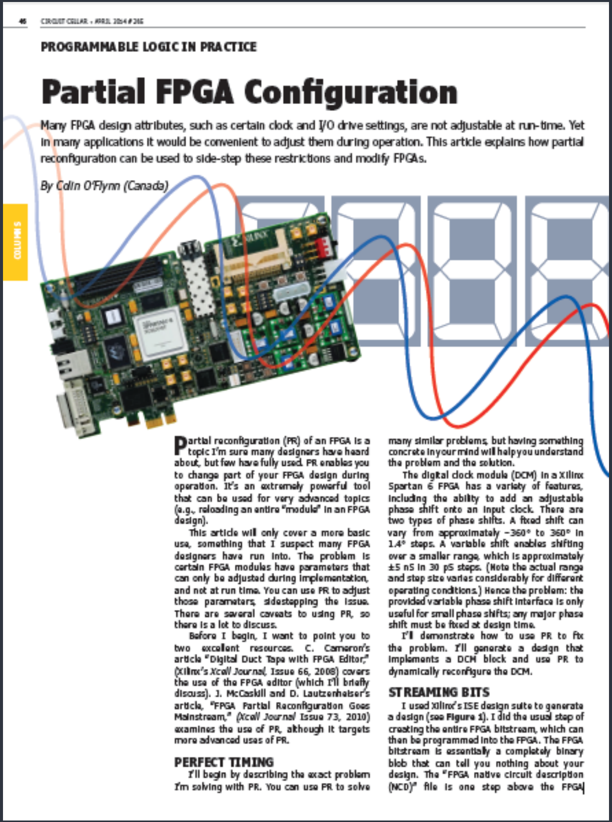


Delay Lines										
DCM_DELAY_STEP(5)	Finest delay resolution, averaged over all steps.	10	40	10	40	10	40	10	40	ps

Table 59: **Switching Characteristics for the Phase-Shift Clock in Variable Phase Mode(1)**

Symbol	Description	Amount of Phase Shift	Units
Phase Shifting Range			
MAX_STEPS(2)	When CLKIN < 60 MHz, the maximum allowed number of DCM_DELAY_STEP steps for a given CLKIN clock period, where T = CLKIN clock period in ns. When using CLKIN_DIVIDE_BY_2 = TRUE, double the clock-effective clock period.	±(INTEGER(10 x (TCLKIN – 3 ns)))	steps
	When CLKIN ≥ 60 MHz, the maximum allowed number of DCM_DELAY_STEP steps for a given CLKIN clock period, where T = CLKIN clock period in ns. When using CLKIN_DIVIDE_BY_2 = TRUE, double the clock-effective clock period.	±(INTEGER(15 x (TCLKIN – 3 ns)))	steps
FINE_SHIFT_RANGE_MIN	Minimum guaranteed delay for variable phase shifting.	±(MAX_STEPS x DCM_DELAY_STEP_MIN)	ps
FINE_SHIFT_RANGE_MAX	Maximum guaranteed delay for variable phase shifting	±(MAX_STEPS x DCM_DELAY_STEP_MAX)	ps

PARTIAL RECONFIG



46 CIRCUIT CELLAR • APRIL 2014 #285

PROGRAMMABLE LOGIC IN PRACTICE

COLUMNS

Partial FPGA Configuration

Many FPGA design attributes, such as certain clock and I/O drive settings, are not adjustable at run-time. Yet in many applications it would be convenient to adjust them during operation. This article explains how partial reconfiguration can be used to side-step these restrictions and modify FPGAs.

By Colin O'Flynn (Canada)

Partial reconfiguration (PR) of an FPGA is a topic I'm sure many designers have heard about, but few have fully used. PR enables you to change part of your FPGA design during operation. It's an extremely powerful tool that can be used for very advanced topics (e.g., reloading an entire "module" in an FPGA design).

This article will only cover a more basic use, something that I suspect many FPGA designers have run into. The problem is certain FPGA modules have parameters that can only be adjusted during implementation, and not at run time. You can use PR to adjust those parameters, sidestepping the issue. There are several caveats to using PR, so there is a lot to discuss.

Before I begin, I want to point you to two excellent resources. C. Cameron's article "Digital Duct Tape with FPGA Editor," (Xilinx's *Xcell Journal*, Issue 66, 2008) covers the use of the FPGA editor (which I'll briefly discuss). J. McCaskill and D. Lautzenheiser's article, "FPGA Partial Reconfiguration Goes Mainstream," (*Xcell Journal* Issue 73, 2010) examines the use of PR, although it targets more advanced uses of PR.

PERFECT TIMING

I'll begin by describing the exact problem I'm solving with PR. You can use PR to solve many similar problems, but having something concrete in your mind will help you understand the problem and the solution.

The digital clock module (DCM) in a Xilinx Spartan 6 FPGA has a variety of features, including the ability to add an adjustable phase shift onto an input clock. There are two types of phase shifts. A fixed shift can vary from approximately –360° to 360° in 1.4° steps. A variable shift enables shifting over a smaller range, which is approximately ±5 nS in 30 pS steps. (Note the actual range and step size varies considerably for different operating conditions.) Hence the problem: the provided variable phase shift interface is only useful for small phase shifts; any major phase shift must be fixed at design time.

I'll demonstrate how to use PR to fix the problem. I'll generate a design that implements a DCM block and use PR to dynamically reconfigure the DCM.

STREAMING BITS

I used Xilinx's ISE design suite to generate a design (see **Figure 1**). I did the usual step of creating the entire FPGA bitstream, which can then be programmed into the FPGA. The FPGA bitstream is essentially a completely binary blob that can tell you nothing about your design. The "FPGA native circuit description (NCD)" file is one step above the FPGA

Generate bitstreams for fixed phase shift

- 256 options for each DCM to cover -50% to +50%
- 2 DCMs

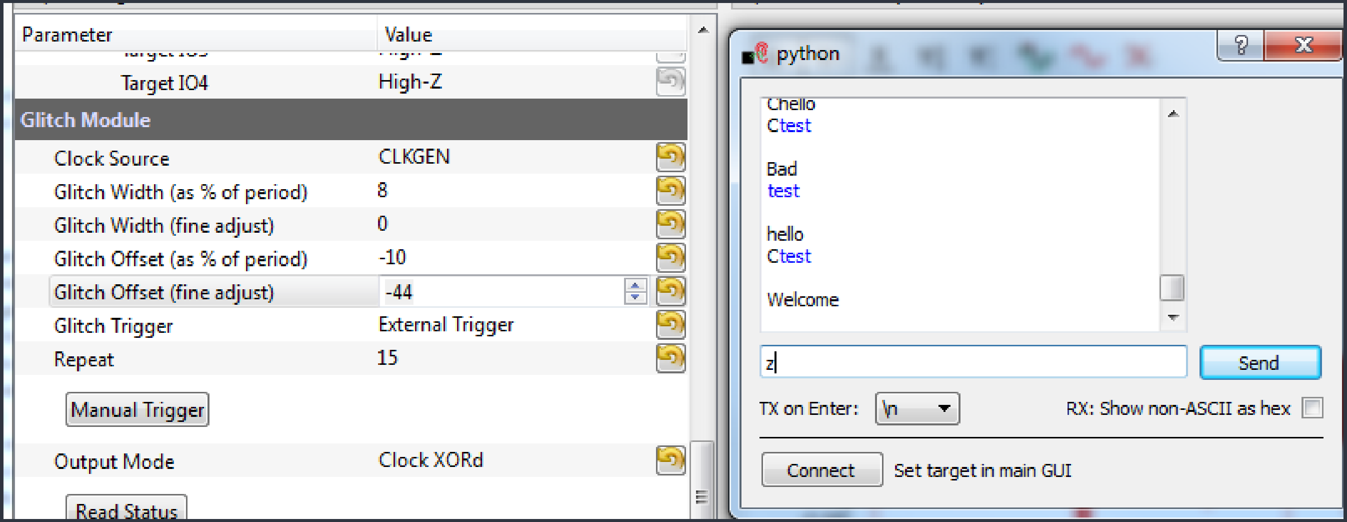
Generate 'Difference' Files for internal Partial Reconfiguration module

http://programmablelogicinpractice.com/?p=143

STUPID DEMO

```c
void glitch3()
{
    char inp[16];
    char c;
    int cnt = 0;
    output_ch_0('C');

    c = 'A';
    while((c != '\n') & (cnt < 16)){
        c = input_ch_0();
        inp[cnt] = c;
        cnt++;
    }

    char passwd[] = "touch";
    char passok = 1;

    trigger_high();
    trigger_low();

    //Simple test - doesn't check for too-long password!
    for(cnt = 0; cnt < 5; cnt++){
        if (inp[cnt] != passwd[cnt]){
            passok = 0;
        }
    }

    if (!passok){
        output_ch_0('B');
        output_ch_0('a');
        output_ch_0('d');
        output_ch_0('\n');
    } else {
        output_ch_0('W');
        output_ch_0('e');
        output_ch_0('l');
        output_ch_0('c');
        output_ch_0('o');
        output_ch_0('m');
        output_ch_0('e');
        output_ch_0('\n');
    }
}
```

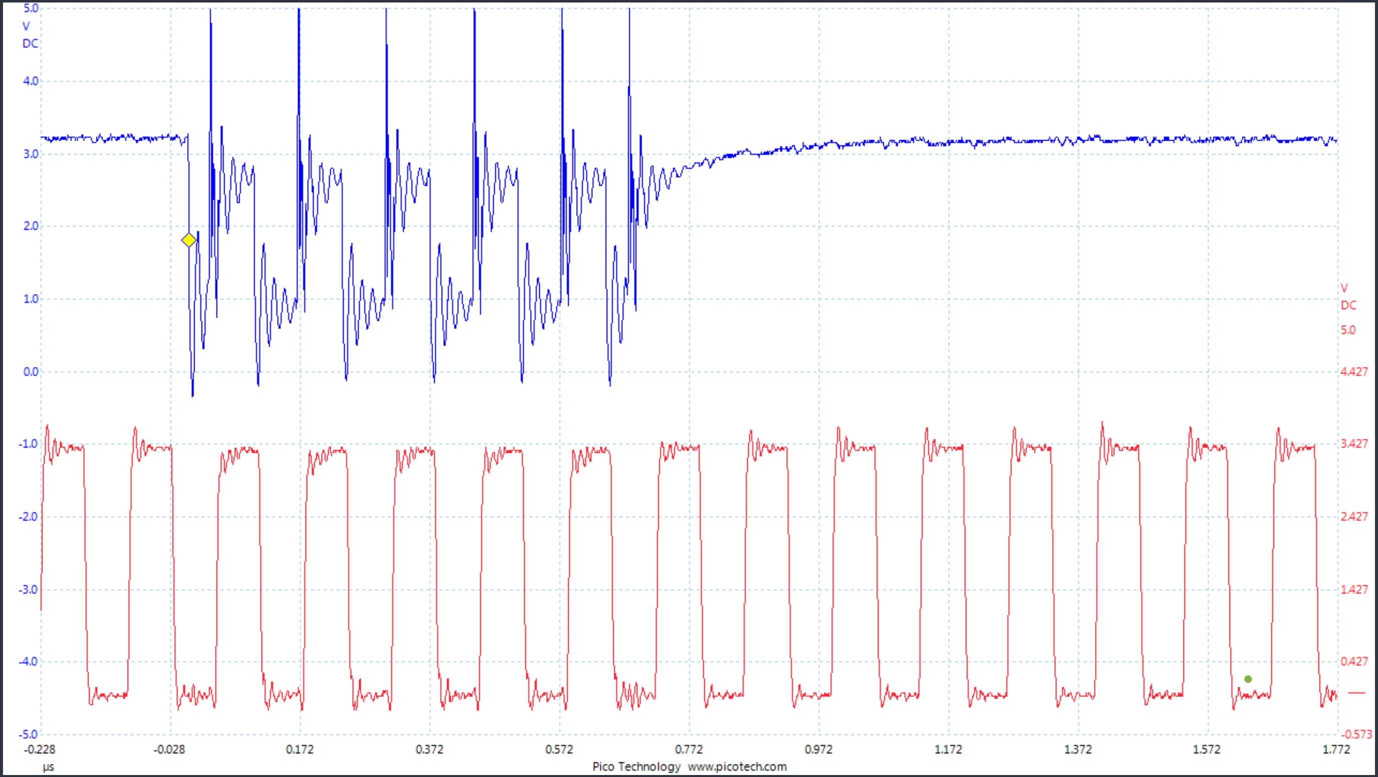
IT WERKS!



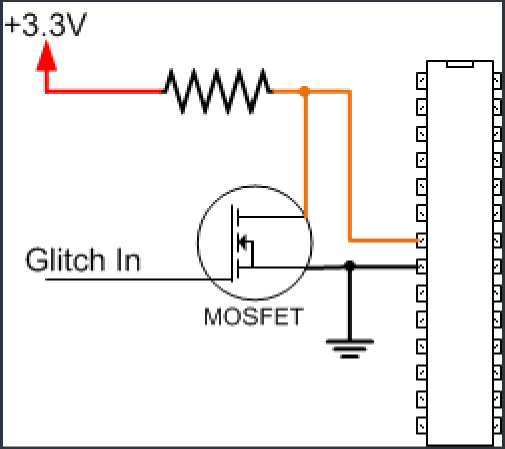
Parameter	Value
Target IO4	High-Z
Glitch Module	
Clock Source	CLKGEN
Glitch Width (as % of period)	8
Glitch Width (fine adjust)	0
Glitch Offset (as % of period)	-10
Glitch Offset (fine adjust)	-44
Glitch Trigger	External Trigger
Repeat	15
Output Mode	Clock XORd

```
Chello
Ctest

Bad
test

hello
Ctest

Welcome
```

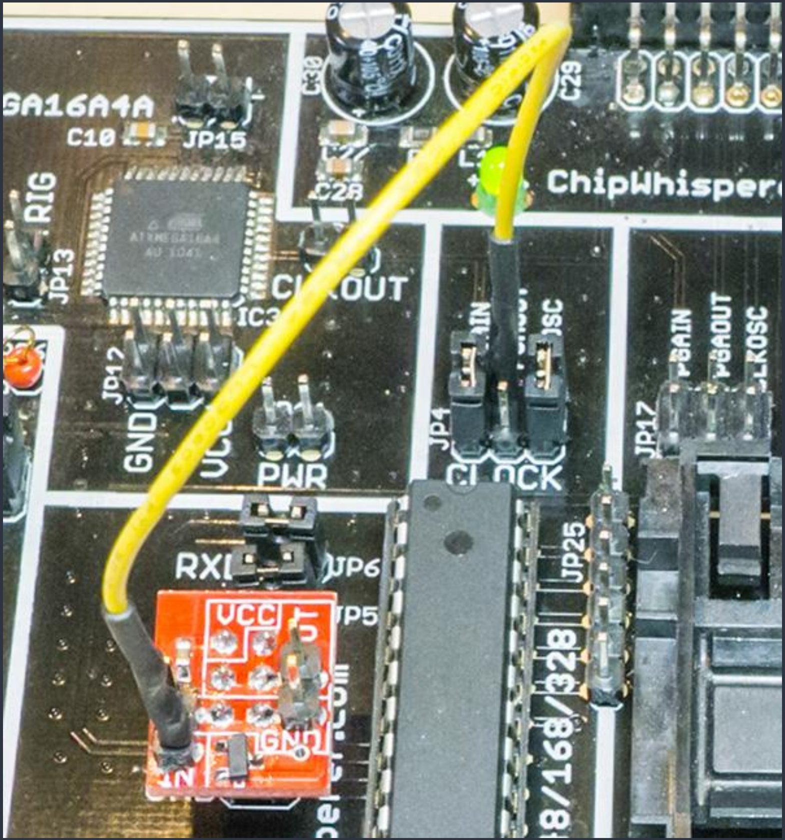
Vcc Glitch


5.0
V
DC
4.0
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-4.0
-5.0
V
DC
5.0
4.427
3.427
2.427
1.427
0.427
-0.573
-0.228
-0.028
0.172
0.372
0.572
0.772
0.972
1.172
1.372
1.572
1.772
µs
Pico Technology www.picotech.com

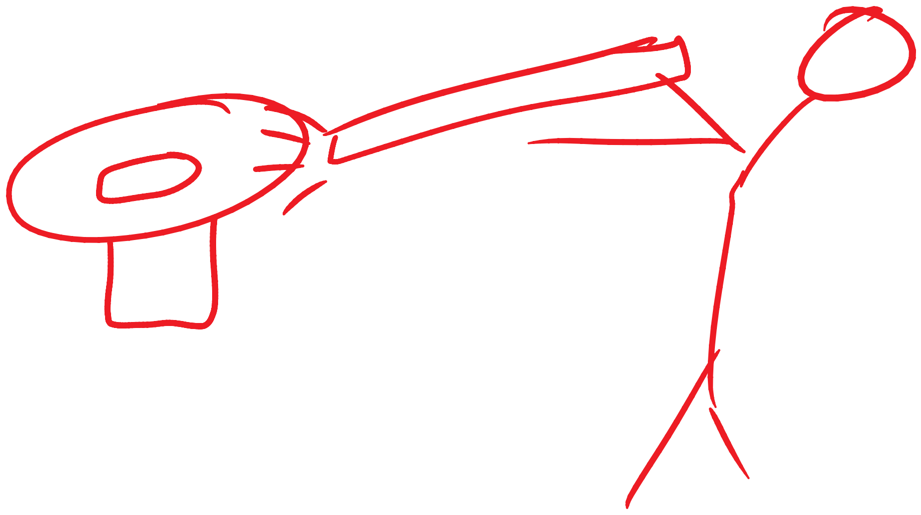
GLITCH HW

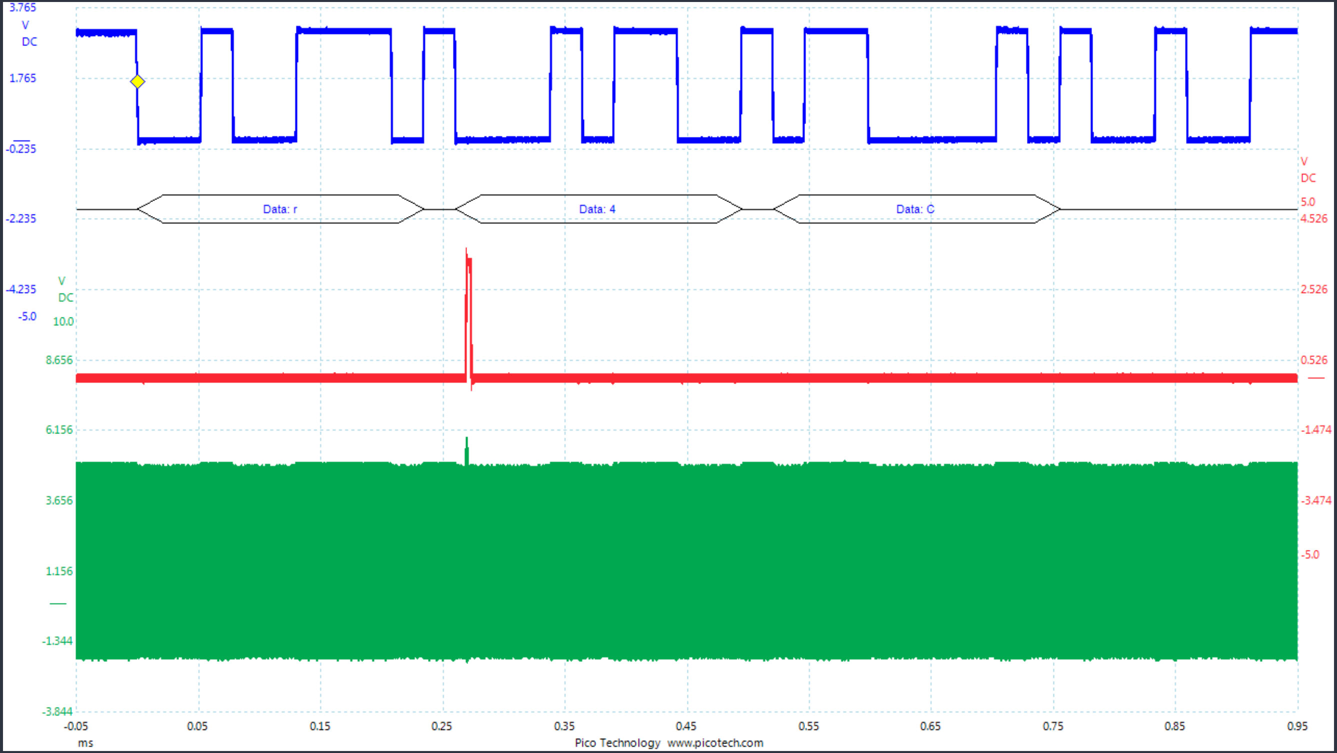


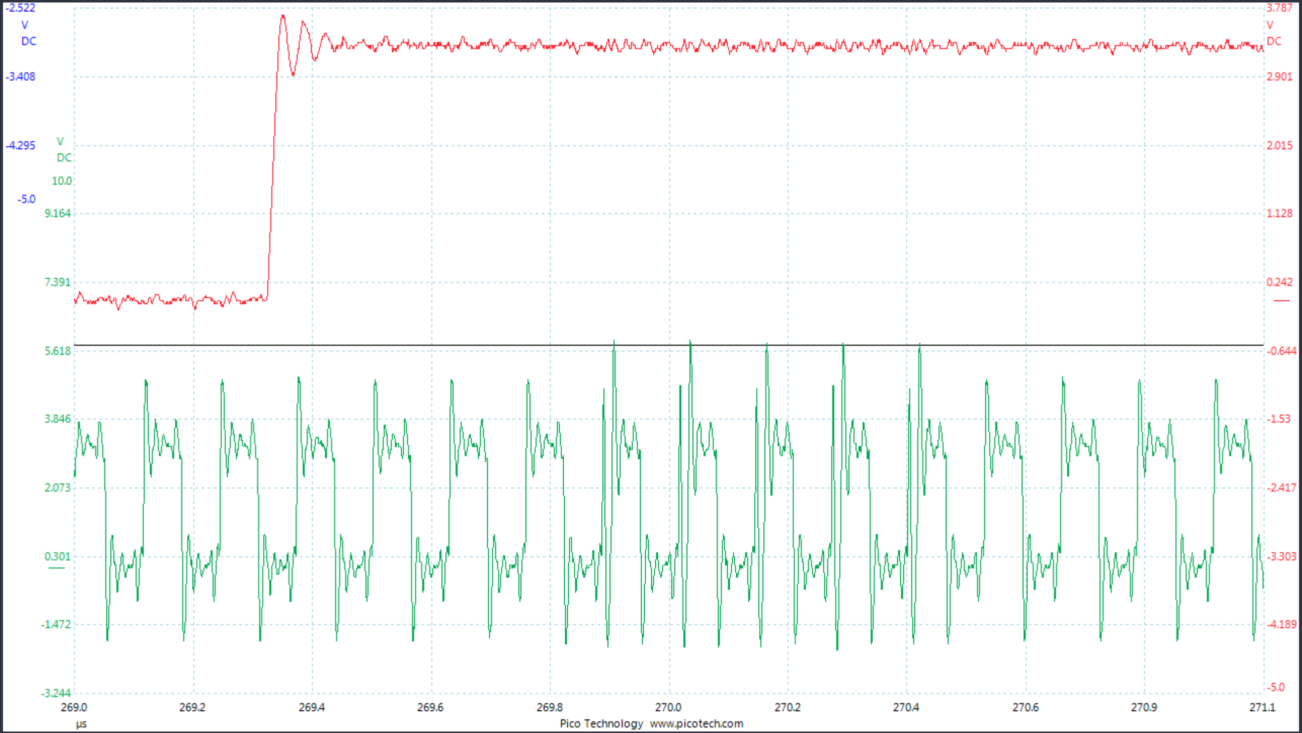
I. R. L.

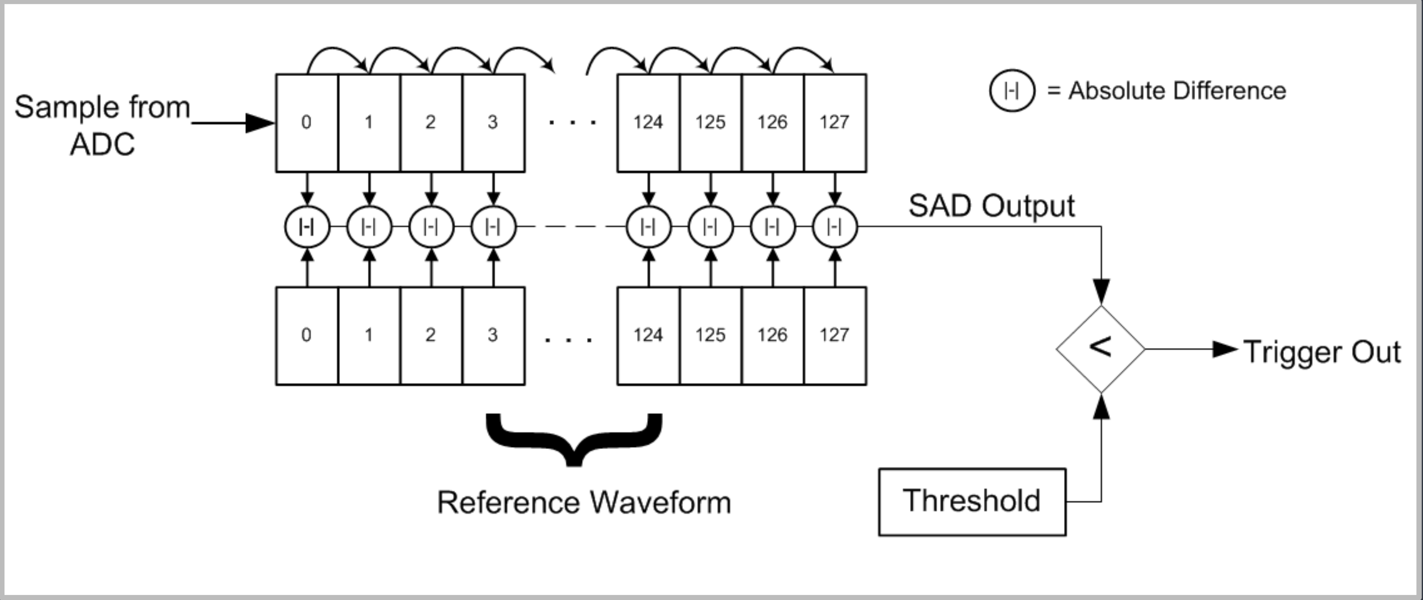

GA16A4A
C10
JP15
C30
C29
C28
ChipWhisperer
RIG
JP13
IC3
CLKOUT
JP12
GND
VCC
PWR
JP4
CLOCK
OSC
JP17
GAIN
GAOUT
LKOSC
RXD
JP6
JP5
VCC
OUT
GND
IN
JP25
8/168/328

TRIGGERING..

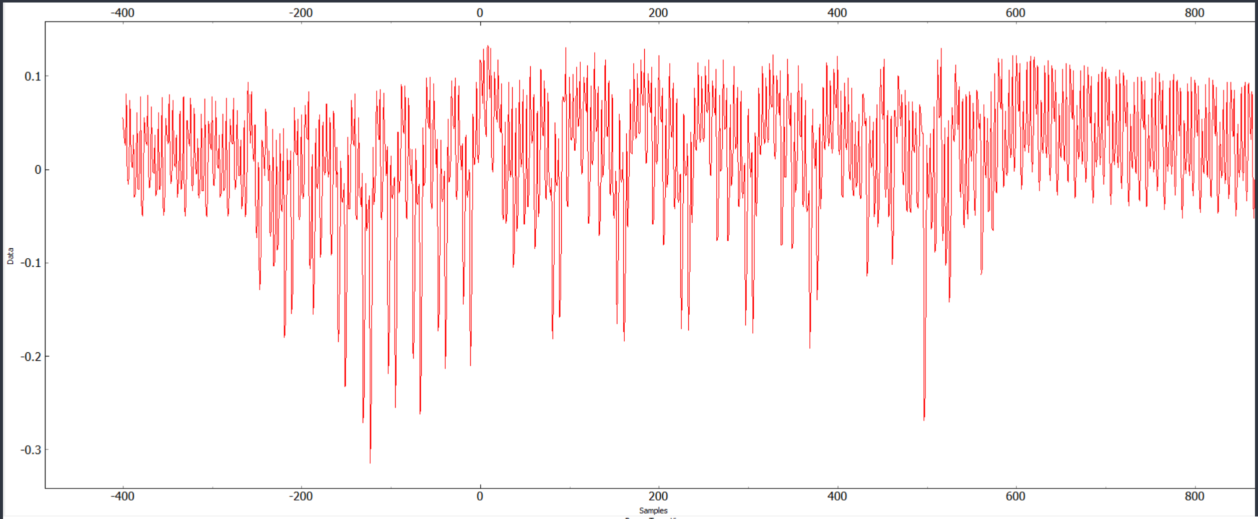



3.765
V
DC
1.765
-0.235
-2.235
-4.235
-5.0
V
DC
10.0
8.656
6.156
3.656
1.156
-1.344
-3.844
Data: r
Data: 4
Data: C
V
DC
5.0
4.526
2.526
0.526
-1.474
-3.474
-5.0
-0.05
0.05
0.15
0.25
0.35
0.45
0.55
0.65
0.75
0.85
0.95
ms
Pico Technology www.picotech.com

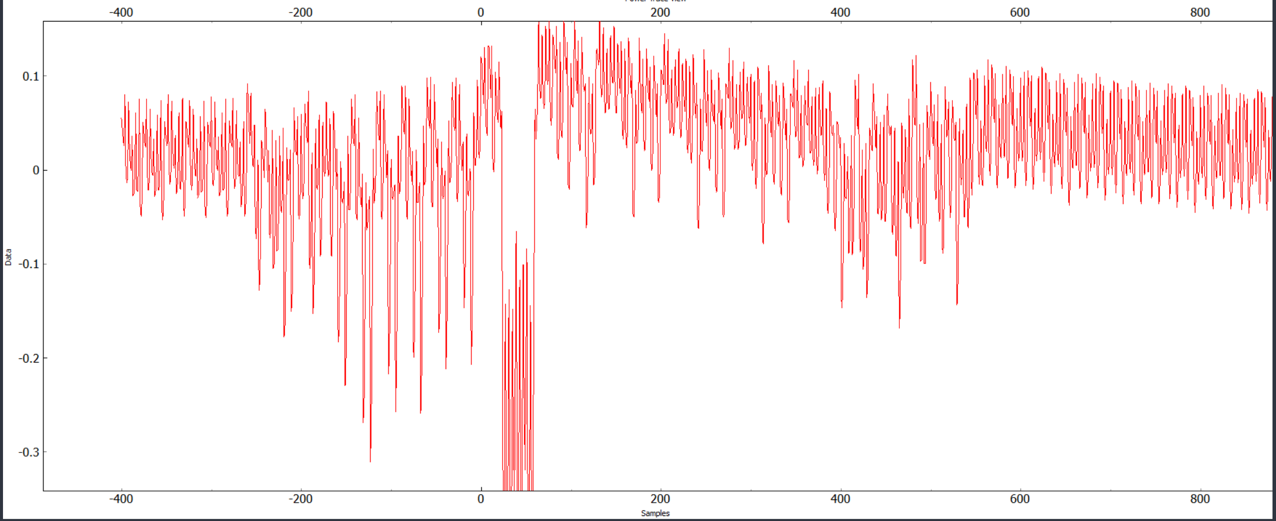

-2.522
V
DC
-3.408
-4.295
-5.0
V
DC
10.0
9.164
7.391
5.618
3.846
2.073
0.301
-1.472
-3.244
3.787
V
DC
2.901
2.015
1.128
0.242
-0.644
-1.53
-2.417
-3.303
-4.189
-5.0
269.0
µs
269.2
269.4
269.6
269.8
270.0
270.2
270.4
270.6
270.9
271.1
Pico Technology www.picotech.com



Sum of Absolute Diff

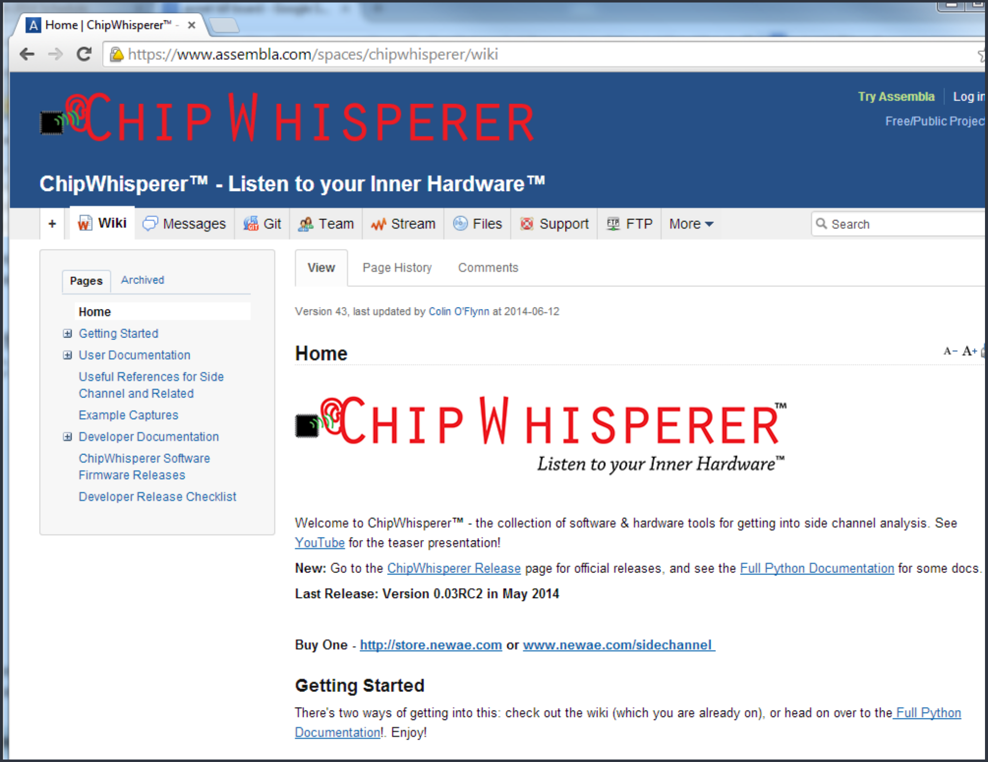


Power Trace View



WHERE TO?

ChipWhisperer.com



Home | ChipWhisperer™

https://www.assembla.com/spaces/chipwhisperer/wiki

Try Assembla | Log in

Free/Public Projec

ChipWhisperer™ - Listen to your Inner Hardware™

+ Wiki Messages Git Team Stream Files Support FTP More Search

Pages Archived

- Home
- Getting Started
- User Documentation
- Useful References for Side Channel and Related
- Example Captures
- Developer Documentation
- ChipWhisperer Software Firmware Releases
- Developer Release Checklist

View Page History Comments

Version 43, last updated by Colin O'Flynn at 2014-06-12

Home

CHIPWHISPERER™ *Listen to your Inner Hardware™*

Welcome to ChipWhisperer™ - the collection of software & hardware tools for getting into side channel analysis. See YouTube for the teaser presentation!

New: Go to the ChipWhisperer Release page for official releases, and see the Full Python Documentation for some docs.

Last Release: Version 0.03RC2 in May 2014

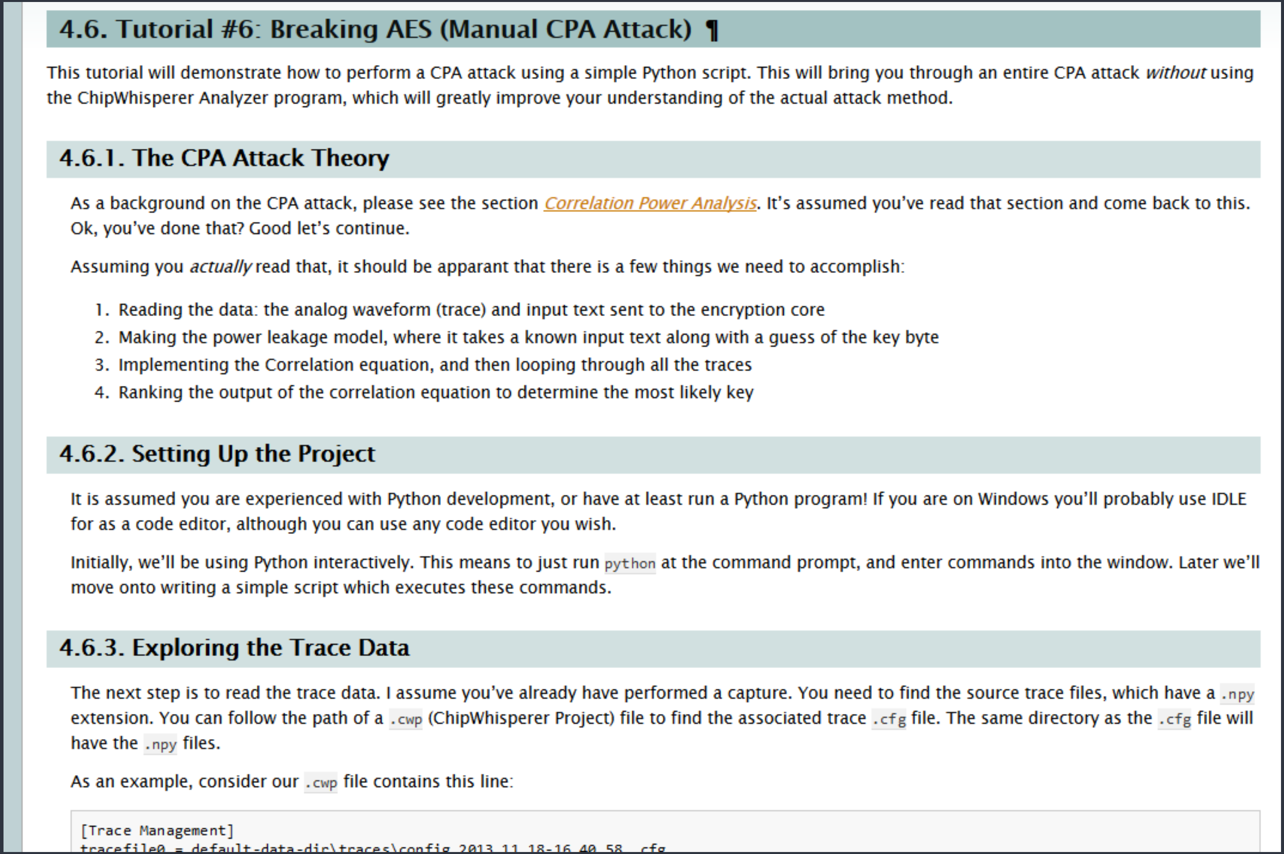
Buy One - http://store.newae.com or www.newae.com/sidechannel

Getting Started

There's two ways of getting into this: check out the wiki (which you are already on), or head on over to the Full Python Documentation!. Enjoy!

TODO LIST

* CHECK FULL DOCS
* DOWNLOAD TRACES
* RUN THE TUTORIALS
* BUILD SOME HW?



4.6. Tutorial #6: Breaking AES (Manual CPA Attack) ¶

This tutorial will demonstrate how to perform a CPA attack using a simple Python script. This will bring you through an entire CPA attack *without* using the ChipWhisperer Analyzer program, which will greatly improve your understanding of the actual attack method.

4.6.1. The CPA Attack Theory

As a background on the CPA attack, please see the section *Correlation Power Analysis*. It's assumed you've read that section and come back to this. Ok, you've done that? Good let's continue.

Assuming you *actually* read that, it should be apparant that there is a few things we need to accomplish:

1. Reading the data: the analog waveform (trace) and input text sent to the encryption core
2. Making the power leakage model, where it takes a known input text along with a guess of the key byte
3. Implementing the Correlation equation, and then looping through all the traces
4. Ranking the output of the correlation equation to determine the most likely key

4.6.2. Setting Up the Project

It is assumed you are experienced with Python development, or have at least run a Python program! If you are on Windows you'll probably use IDLE for as a code editor, although you can use any code editor you wish.

Initially, we'll be using Python interactively. This means to just run `python` at the command prompt, and enter commands into the window. Later we'll move onto writing a simple script which executes these commands.

4.6.3. Exploring the Trace Data

The next step is to read the trace data. I assume you've already have performed a capture. You need to find the source trace files, which have a `.npy` extension. You can follow the path of a `.cwp` (ChipWhisperer Project) file to find the associated trace `.cfg` file. The same directory as the `.cfg` file will have the `.npy` files.

As an example, consider our `.cwp` file contains this line:

```
[Trace Management]
tracefile0 = default-data-dir\traces\config_2013_11_18-16_40_58_.cfg
```

COFLYNN@NEWAE.COM

CHIPWHISPERER.COM

ChipWhisperer is a Trademark of NewAE Technology Inc.