Power Analysis Using Low-Cost Hardware: Lab Setup & Simple Targets

CHES 2013 Tutorial

Colin O’Flynn & Dr. Zhizhang (David) Chen. Dalhousie University, Nova Scotia, Canada.

Three Hours of Fun*

Hour 1: Background

1. Fundamentals of SCA using Power Channel
2. Why is it so expensive, let's make it Cheap
3. Introduction to Capture Hardware
4. Attacking Practical Systems
5. H-Field Probes, Amplifiers, Differential Probes
6. Introducing ChipWhisperer Software
7. Scripting CW-Capture

Hour 2: Practical Examples

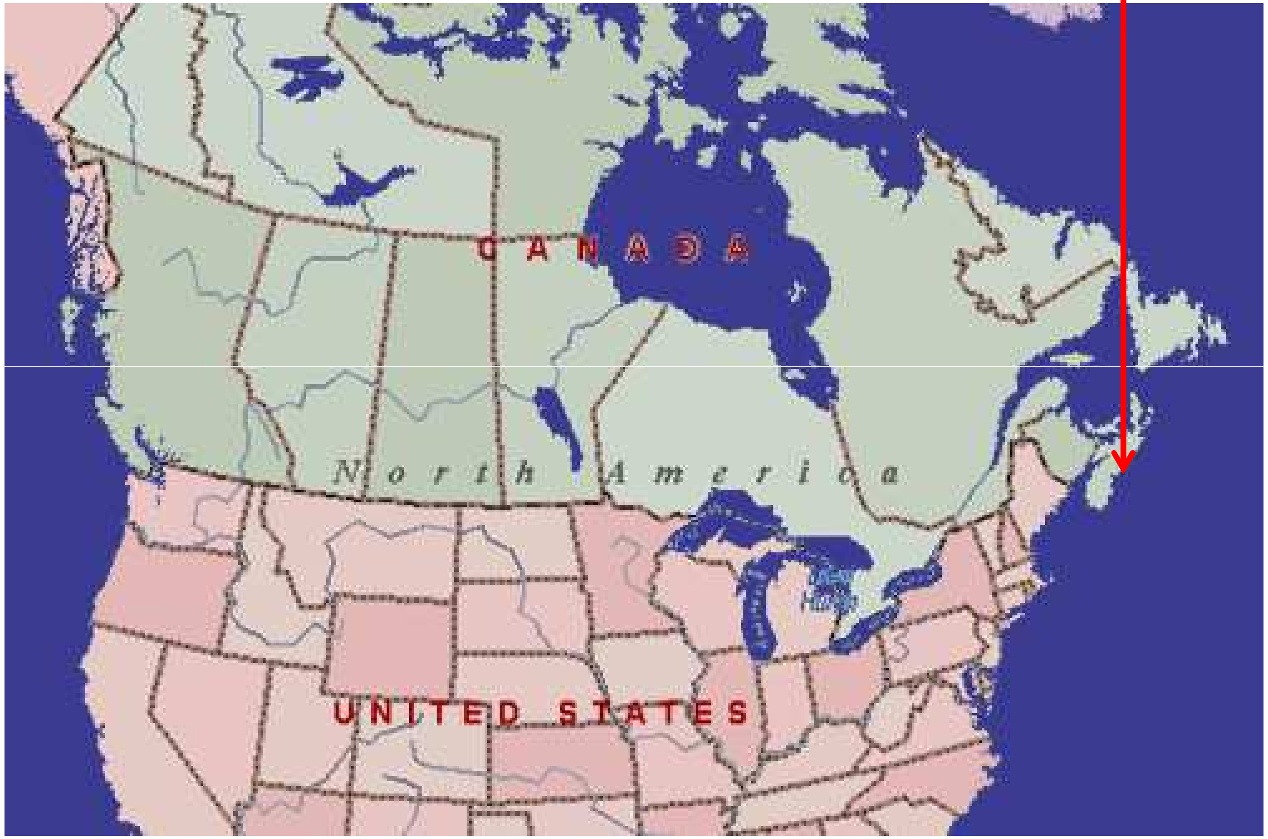
1. Initial Attacks: What should you do?
2. Simple Serial Example
3. SmartCard Example
4. CW-Capture & CW-Analyzer Tutorial
5. Advanced Trigger Modes
6. Interfacing to MATLAB

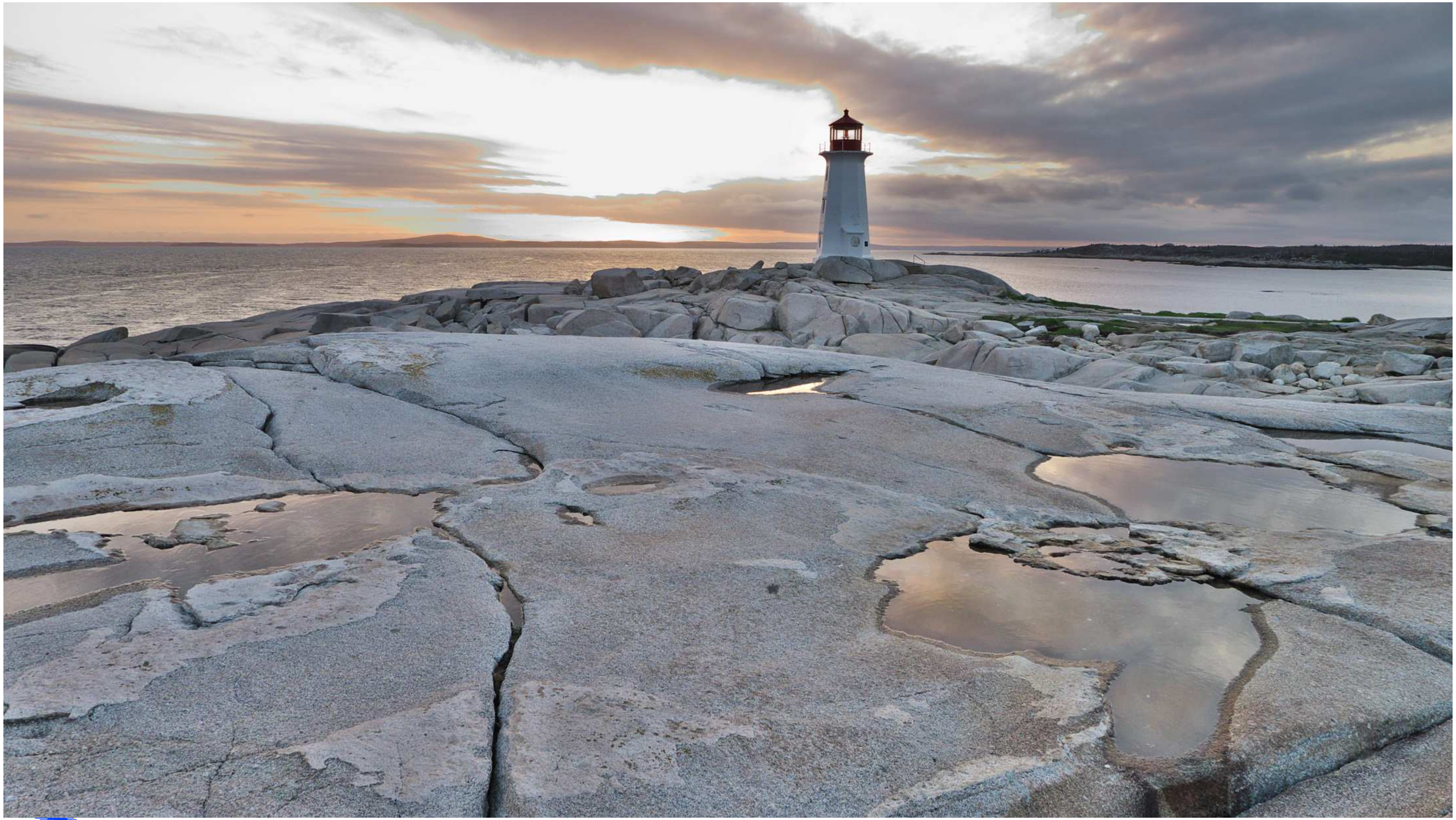
Hour 3: More advanced topics, modifying, hacking

1. Using SASEBO-W as Capture Platform
2. Adding your own modules to CW-Capture
3. Overview of the FPGA Code
4. Questions, Extra Material

***Not guaranteed to be fun**

Halifax, Nova Scotia

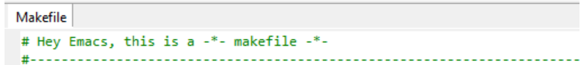




DALHOUSIE UNIVERSITY
Inspiring Minds

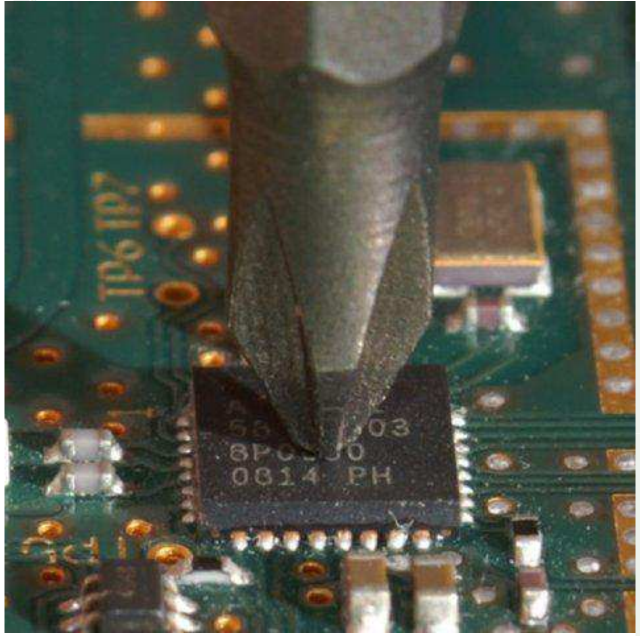
Thanks to funding providers!

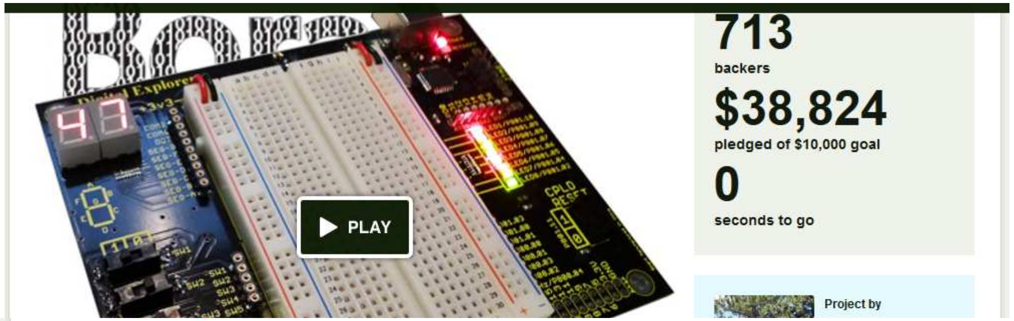
NSERC CRSNG



Makefile

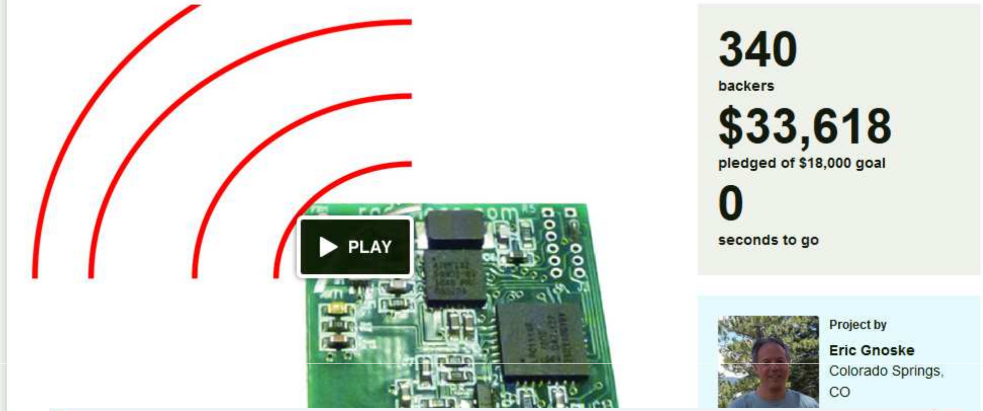
```
# Hey Emacs, this is a -*- makefile -*-
#-----------------------------------------------------------------------
```





713
backers
$38,824
pledged of $10,000 goal
0
seconds to go

Project by



340
backers
$33,618
pledged of $18,000 goal
0
seconds to go

Project by
Eric Gnoske
Colorado Springs, CO

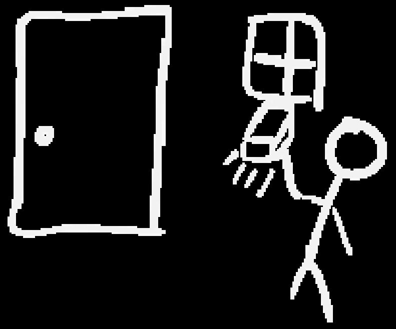
FIR Filter Design with HLS
Abusing Xilinx's Tools for Fun & Profit

Design a FIR Filter in an FPGA in 30 mins using High Level Synthesis

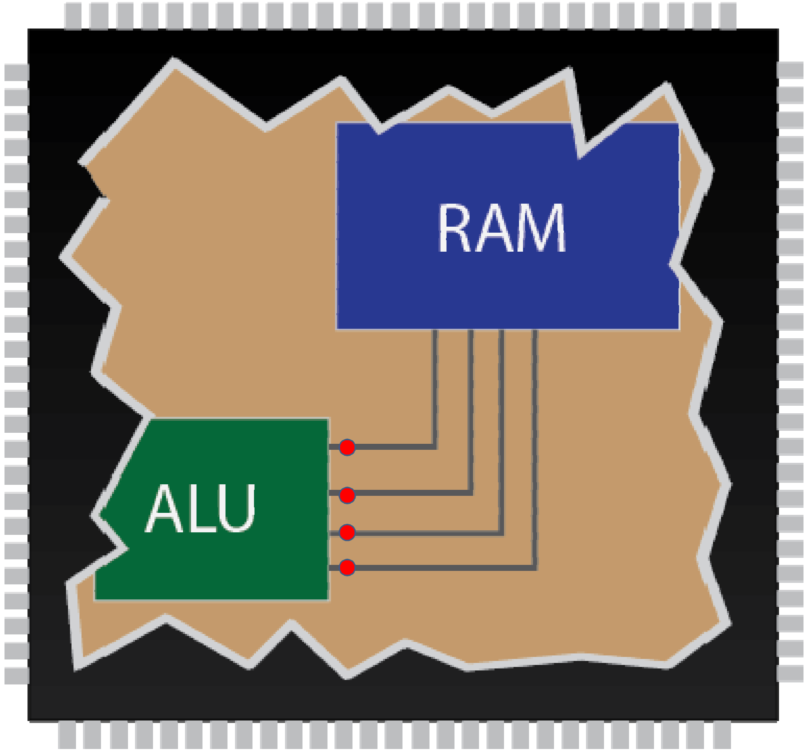
I've been working with Xilinx's new High Level Synthesis tools built into Vivado. I'm slowly working on posting some more complete tutorials. In the mean-time **here** is a simple tutorial about making **a Finite Impulse Response Filter** on a real ADC/DAC board .

Permalink | **Leave a comment**

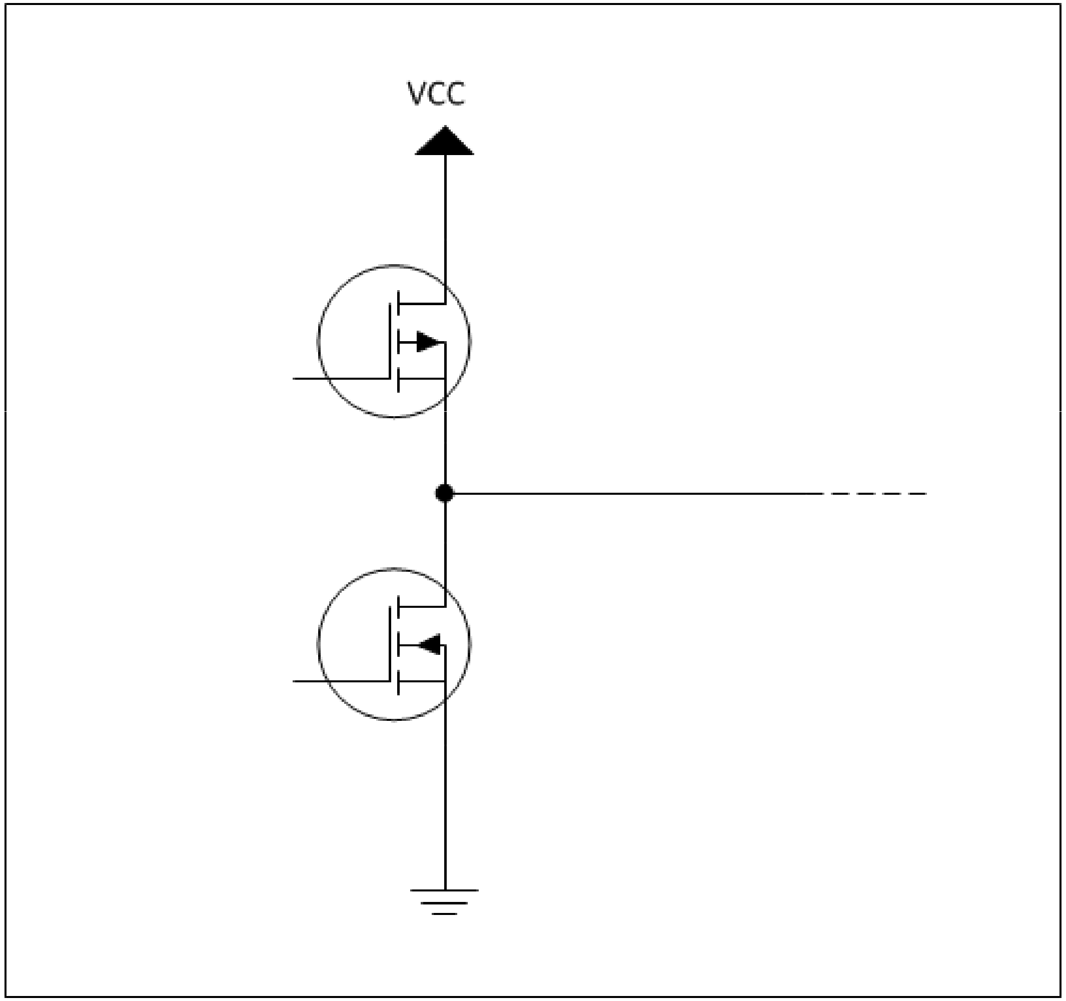
THE SIDE CHANNEL



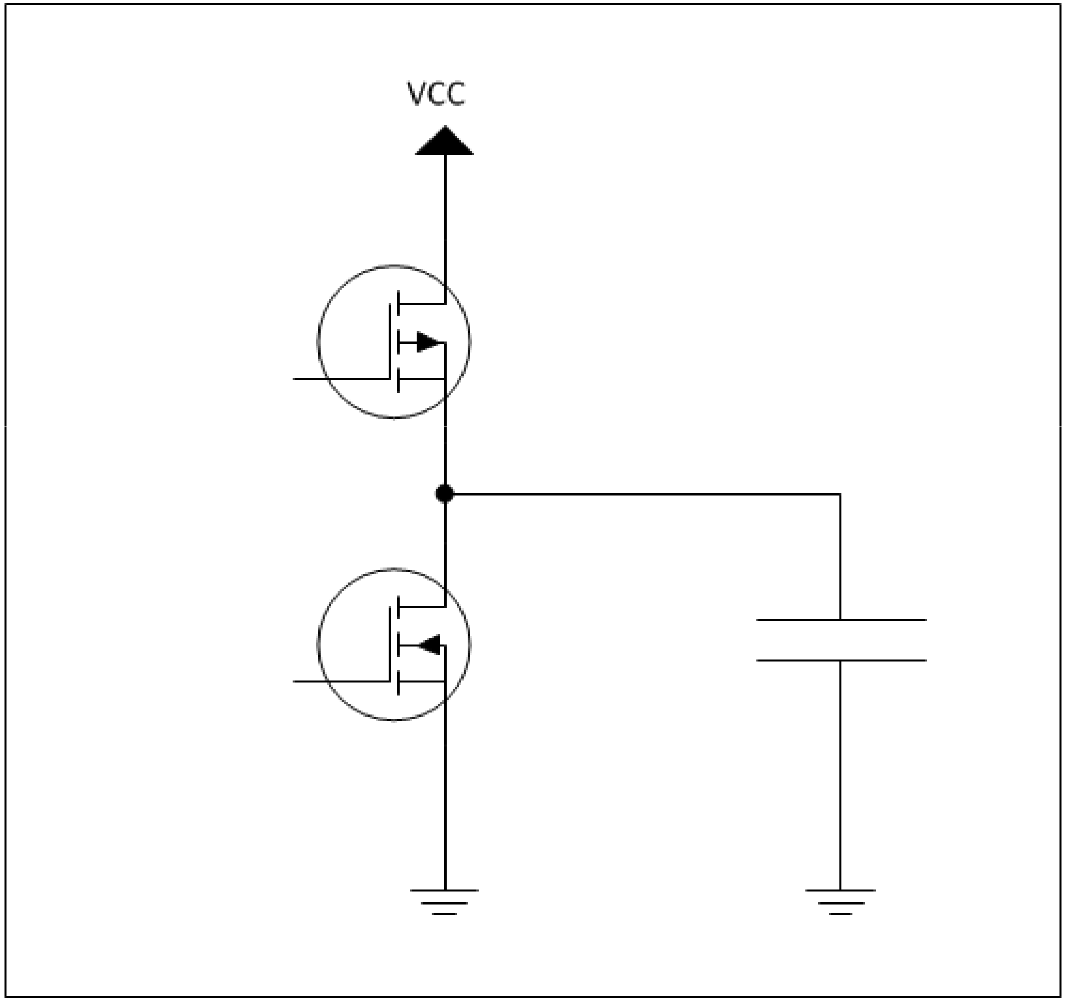
Back to Basics

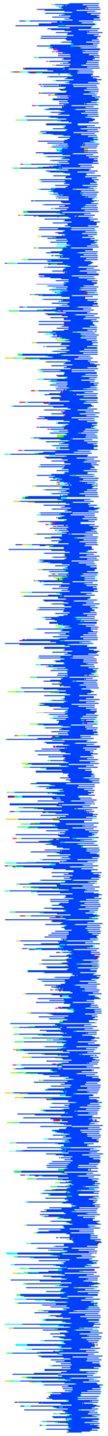


Back to Basics

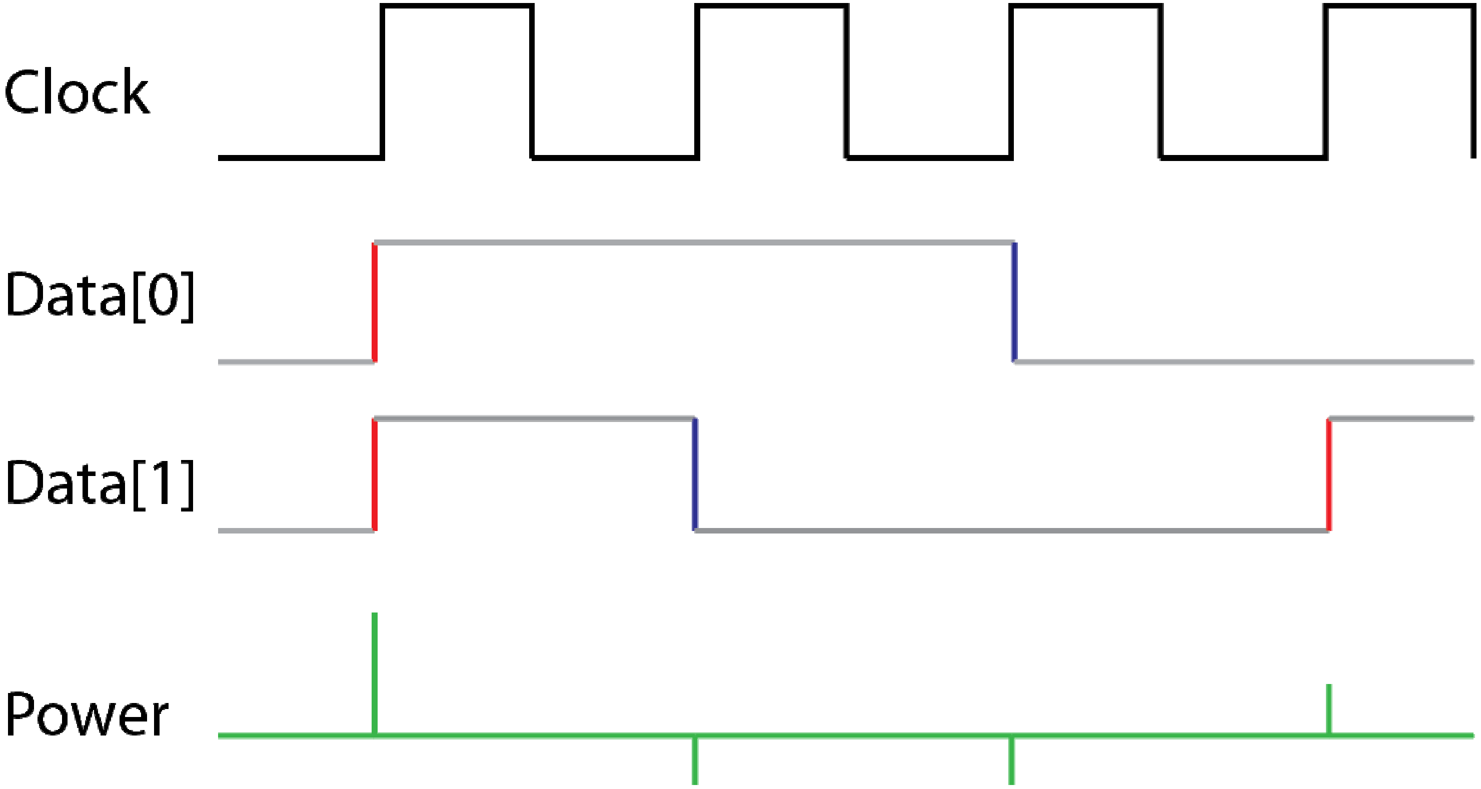


Back to Basics





Back to Basics



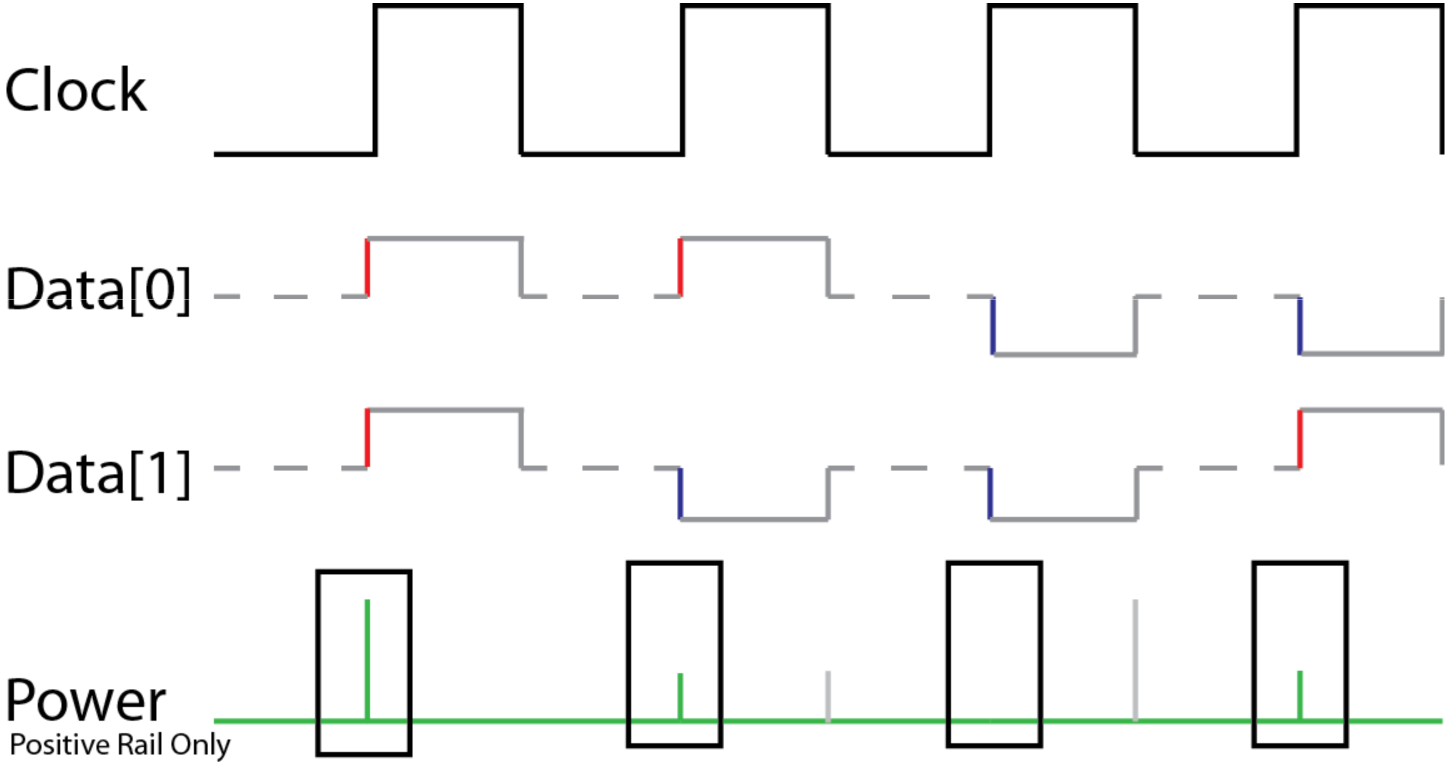
Clock

Data[0]

Data[1]

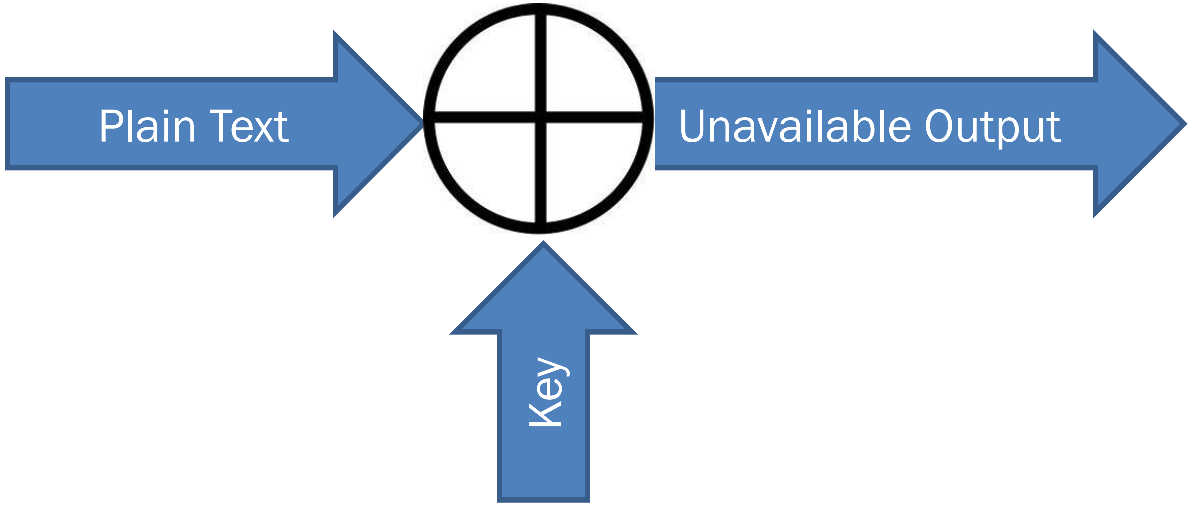
Power

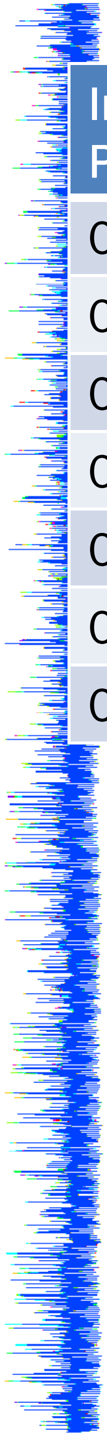
Back to Basics



Weight: 2 1 0
1

Using Power Measurement

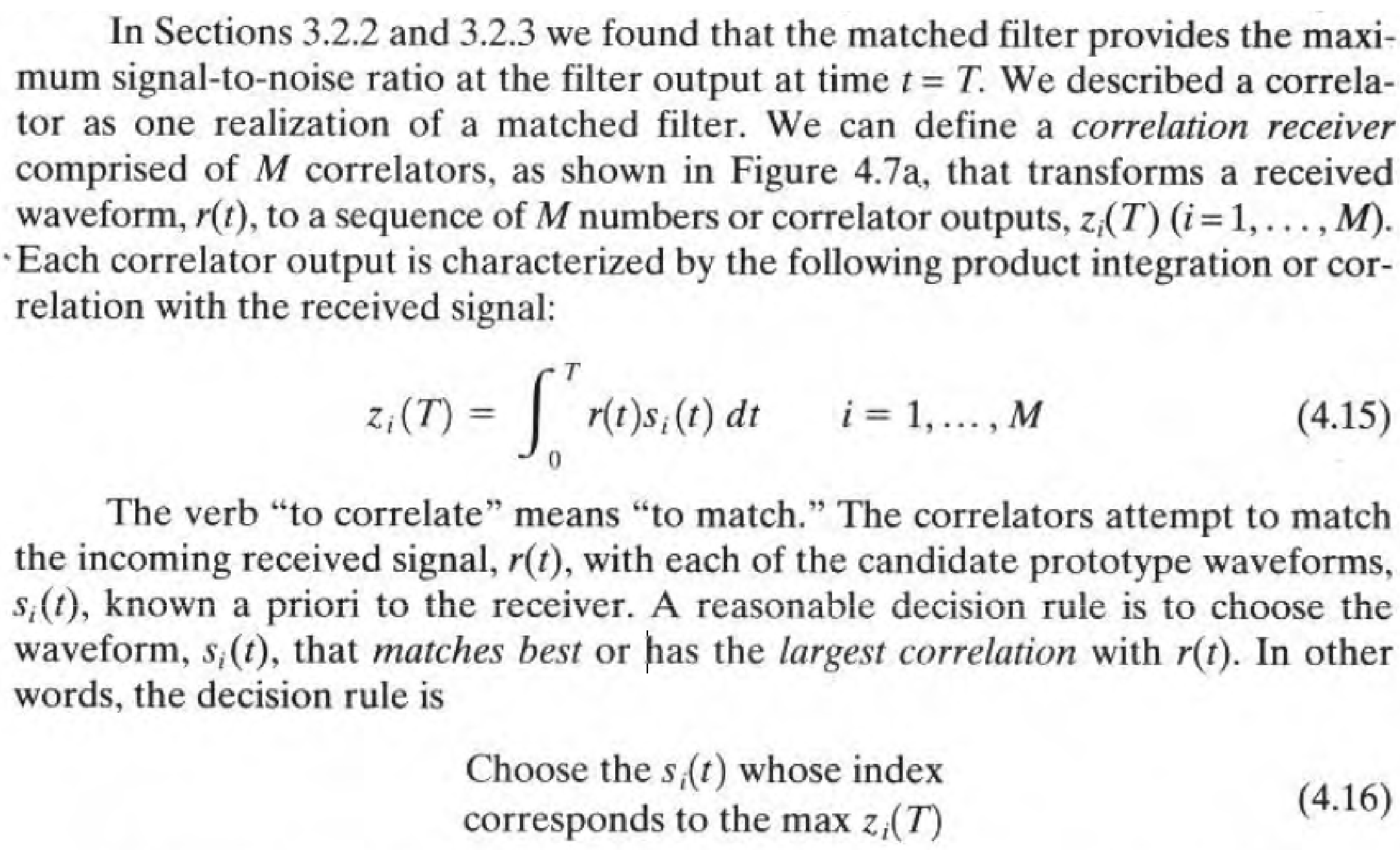




Input Plaintext	Hyp. Key = 0	Hyp Result	Hyp HW
0100 (4)	0000 (0)	0100 (4)	1
0111 (			
0010 (			
0001 (			
0000 (			
0110 (			
0101 (			

Input Plaintext	Hyp. Key=1	Hyp Result	Hyp HW
0100 (4)	0001 (1)	0101 (5)	2
0111 (7			
0010 (2			
0001 (1			
0000 (0			
0110 (6			
0101 (5			

Input Plaintext	Hyp. Key	Hyp Result	Hyp HW
0100 (4)	0010 (2)	0110 (6)	2
0111 (7)	0010 (2)	0101 (5)	2
0010 (2)	0010 (2)	0000 (0)	0
0001 (1)	0010 (2)	0011 (3)	2
0000 (0)	0010 (2)	0010 (2)	1
0110 (6)	0010 (2)	0100 (4)	1
0101 (5)	0010 (2)	0111 (7)	3



In Sections 3.2.2 and 3.2.3 we found that the matched filter provides the maximum signal-to-noise ratio at the filter output at time $t = T$. We described a correlator as one realization of a matched filter. We can define a *correlation receiver* comprised of M correlators, as shown in Figure 4.7a, that transforms a received waveform, $r(t)$, to a sequence of M numbers or correlator outputs, $z_i(T)$ $(i = 1, \ldots, M)$. Each correlator output is characterized by the following product integration or correlation with the received signal:

$$z_i(T) = \int_0^T r(t)s_i(t)\,dt \qquad i = 1, \ldots, M \tag{4.15}$$

The verb "to correlate" means "to match." The correlators attempt to match the incoming received signal, $r(t)$, with each of the candidate prototype waveforms, $s_i(t)$, known a priori to the receiver. A reasonable decision rule is to choose the waveform, $s_i(t)$, that *matches best* or has the *largest correlation* with $r(t)$. In other words, the decision rule is

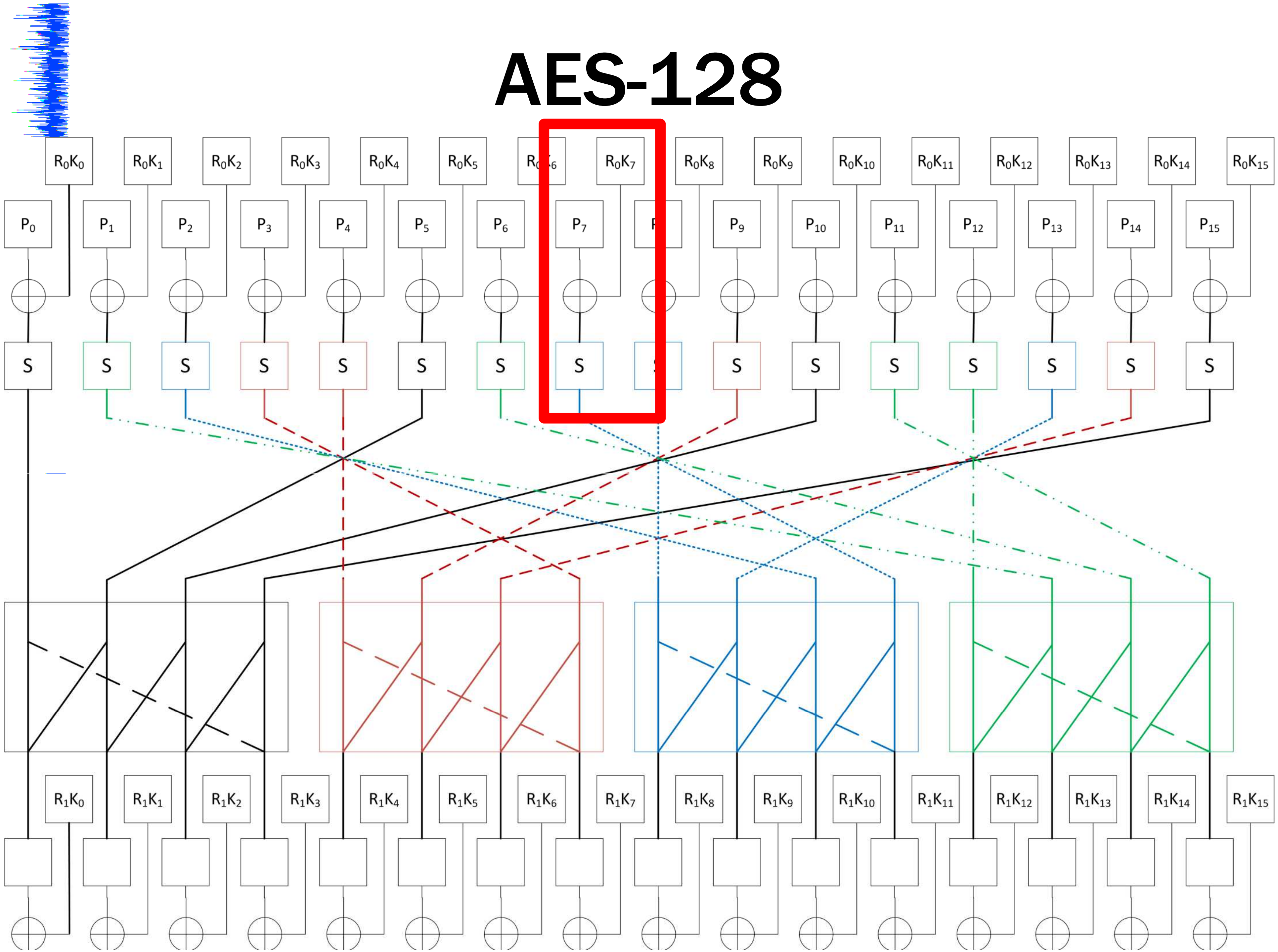
$$\begin{array}{c}\text{Choose the } s_i(t) \text{ whose index}\\ \text{corresponds to the max } z_i(T)\end{array} \tag{4.16}$$

e.g. From "Digital Communications" by Bernard Sklar

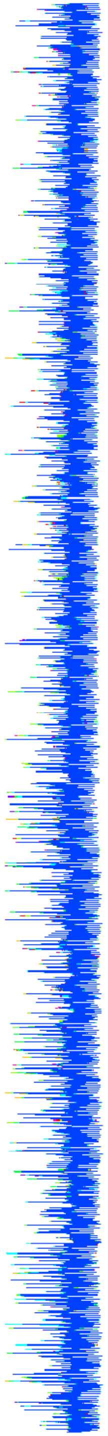
Original Paper:

North, DO. (1943). "An analysis of the factors which determine signal/noise discrimination in pulsed carrier systems"

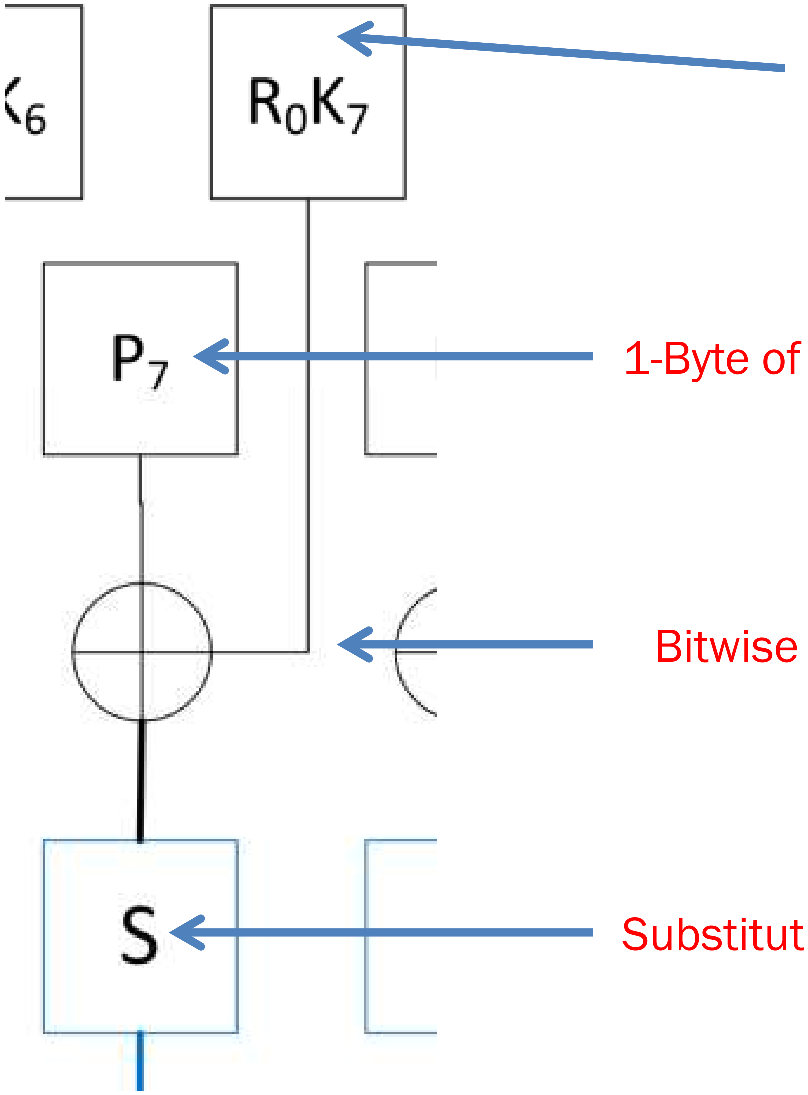
AES-128



R₀K₀
R₀K₁
R₀K₂
R₀K₃
R₀K₄
R₀K₅
R₀K₆
R₀K₇
R₀K₈
R₀K₉
R₀K₁₀
R₀K₁₁
R₀K₁₂
R₀K₁₃
R₀K₁₄
R₀K₁₅
P₀
P₁
P₂
P₃
P₄
P₅
P₆
P₇
P₉
P₁₀
P₁₁
P₁₂
P₁₃
P₁₄
P₁₅
S
R₁K₀
R₁K₁
R₁K₂
R₁K₃
R₁K₄
R₁K₅
R₁K₆
R₁K₇
R₁K₈
R₁K₉
R₁K₁₀
R₁K₁₁
R₁K₁₂
R₁K₁₃
R₁K₁₄
R₁K₁₅



AES-128 Detail



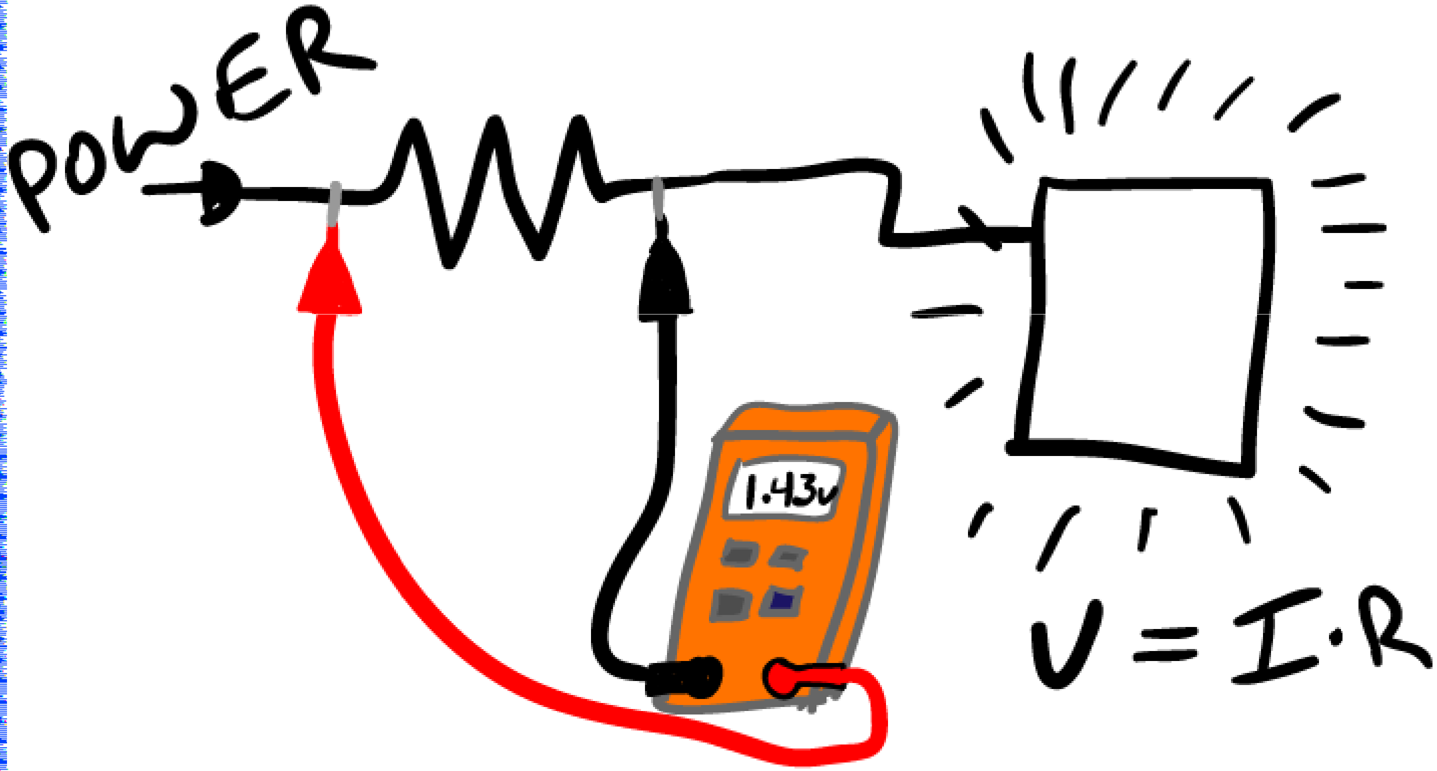
1-Byte of Key (Subkey)

1-Byte of Input (Plaintext)

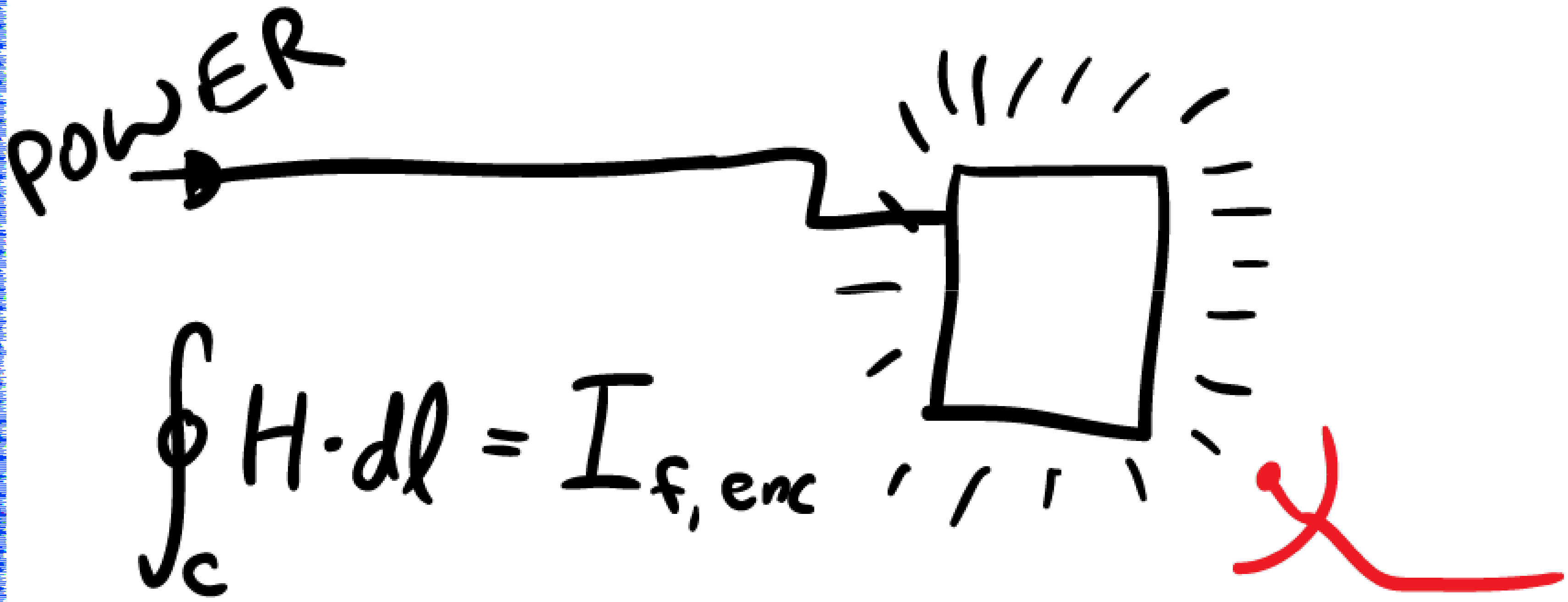
Bitwise XOR

Substitution-Box (Lookup Table)

Measuring Power

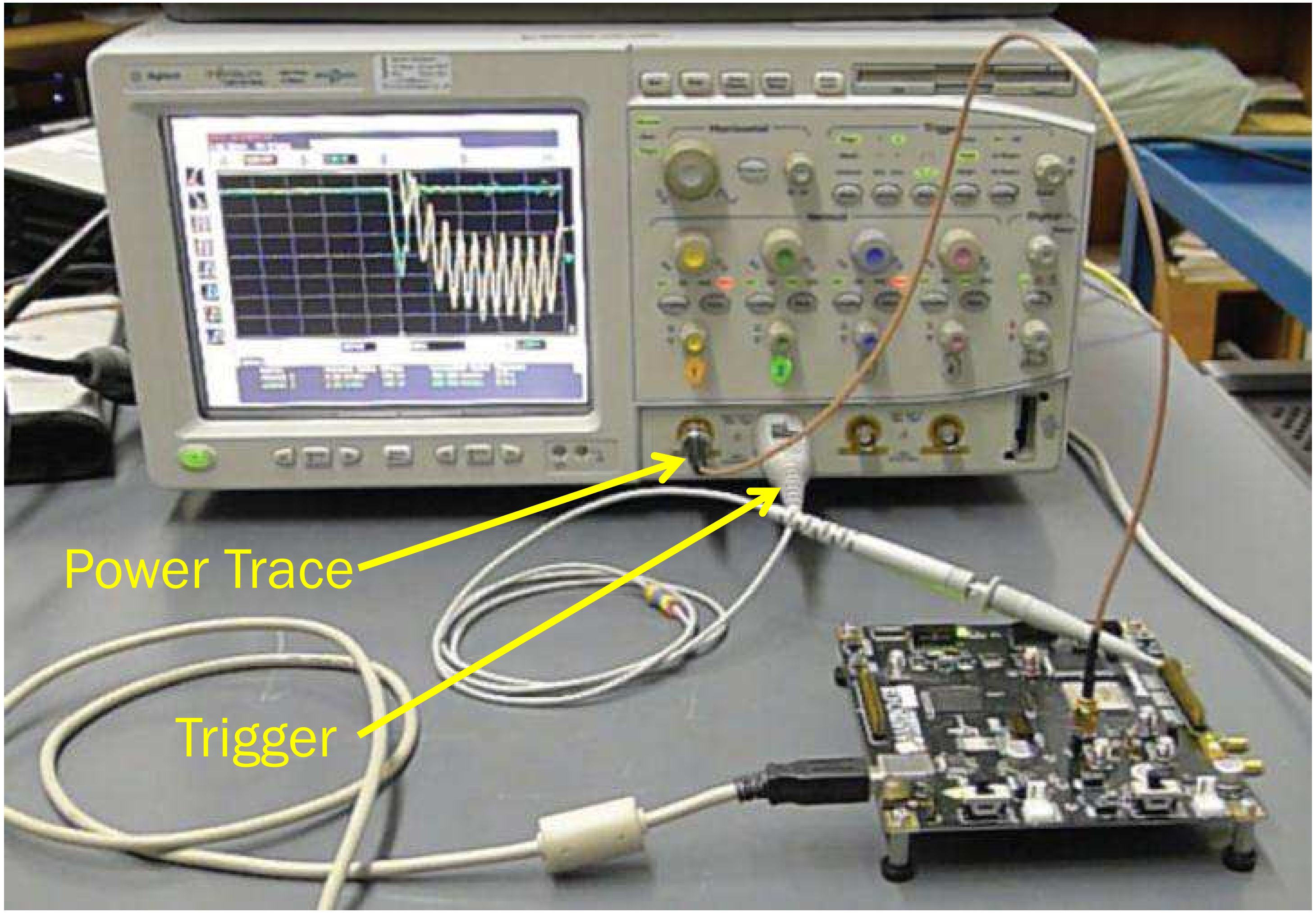


Measuring Power

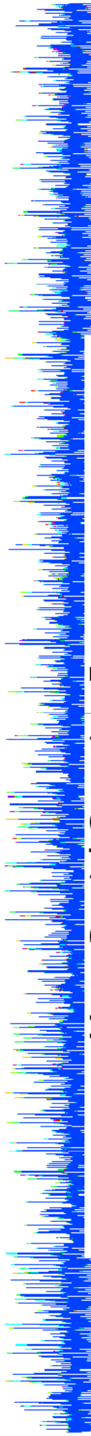


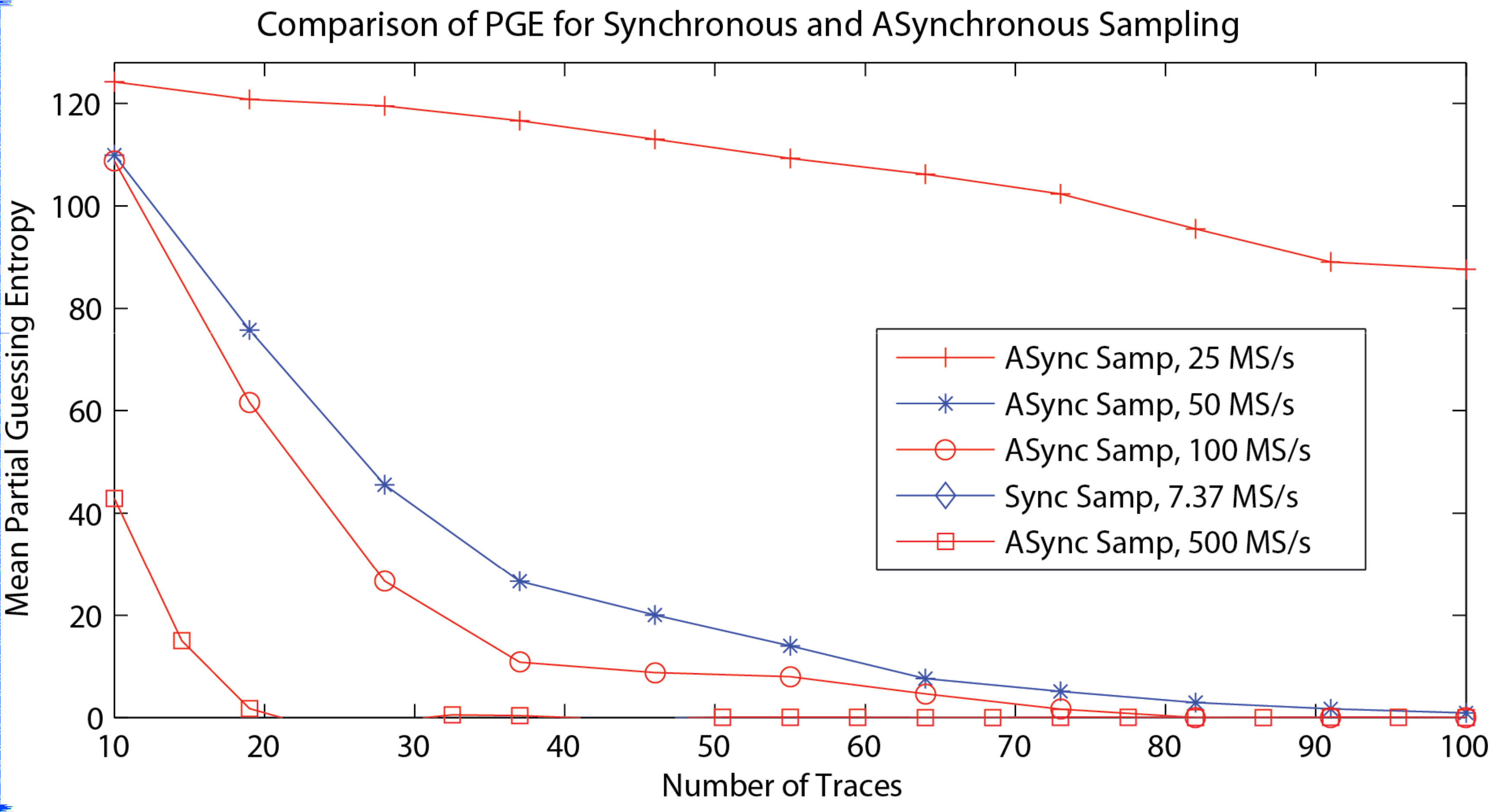
$$\oint_C H \cdot dl = I_{f,enc}$$

WHY IS IT SO EXPENSIVE?

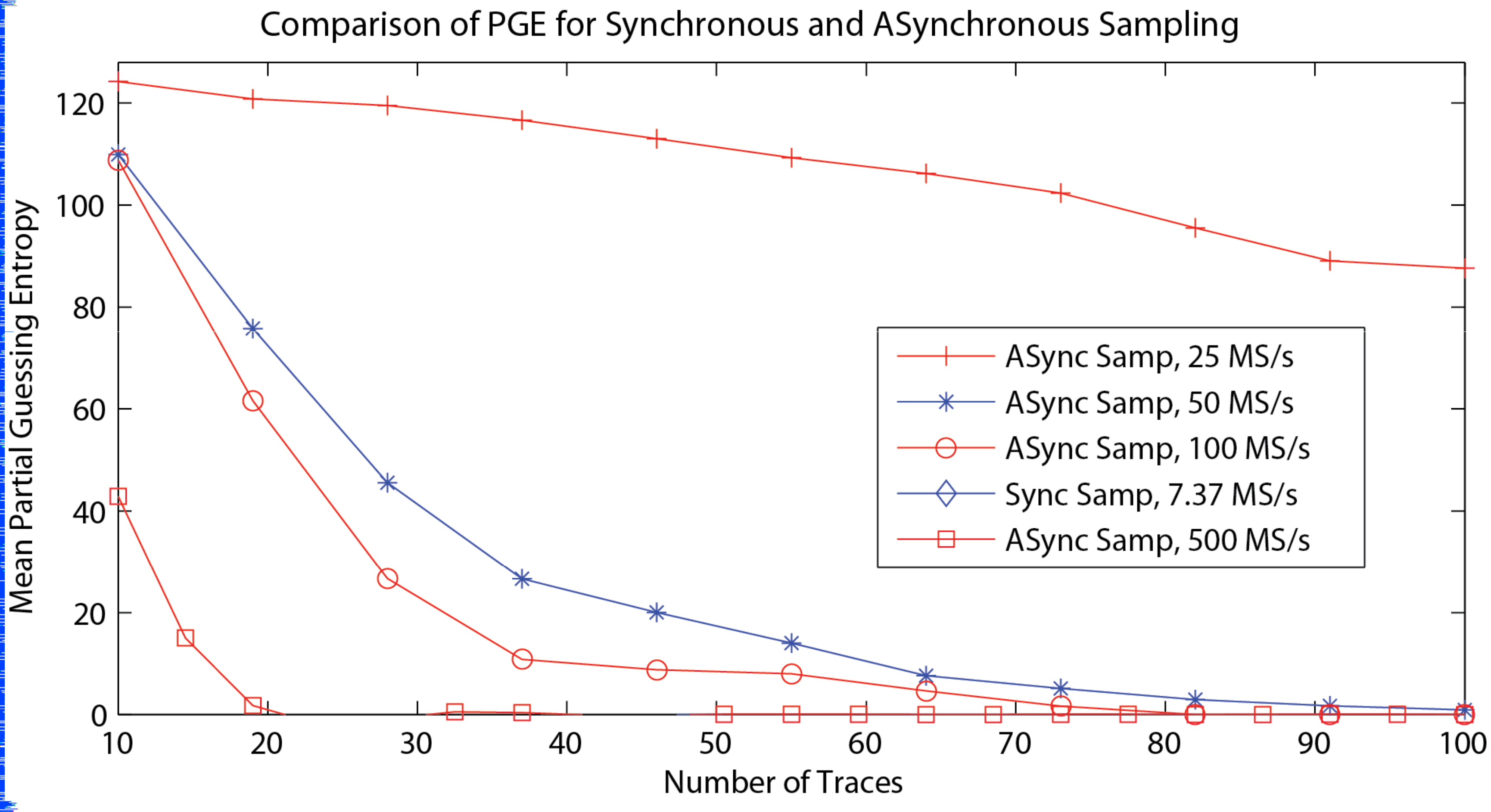

Power Trace
Trigger

Author	Work	Year	Scope	Cost (Used, 2013)
Dario Carluccio	Electromagnetic Side Channel Analysis Embedded Crypto Devices	2005	Infiniium 5432D MSO	$8000
Youssef Souissi et al.	Embedded systems security: An evaluation methodology against Side Channel Attacks	2011	Infiniium 54855	$20 000
Dakshi Agrawal et al.	The EM Side–Channel(s)	2003	100 MHz, 12 bit	$1000
F.X. Standaert et al.	Using subspace-based template attacks to compare and combine power and electromagnetic information leakages	2008	1 GHz bandwidth	$7500

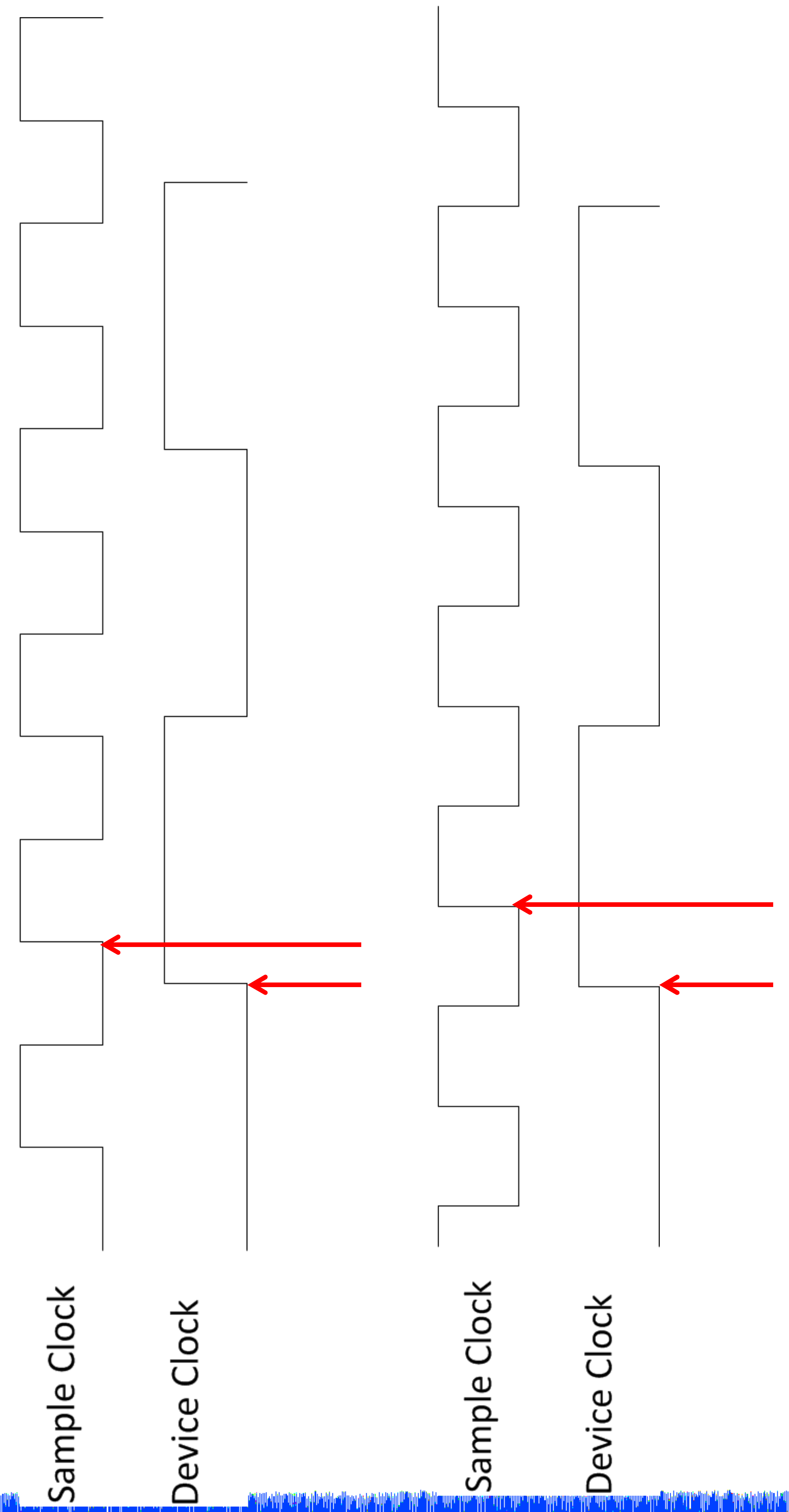




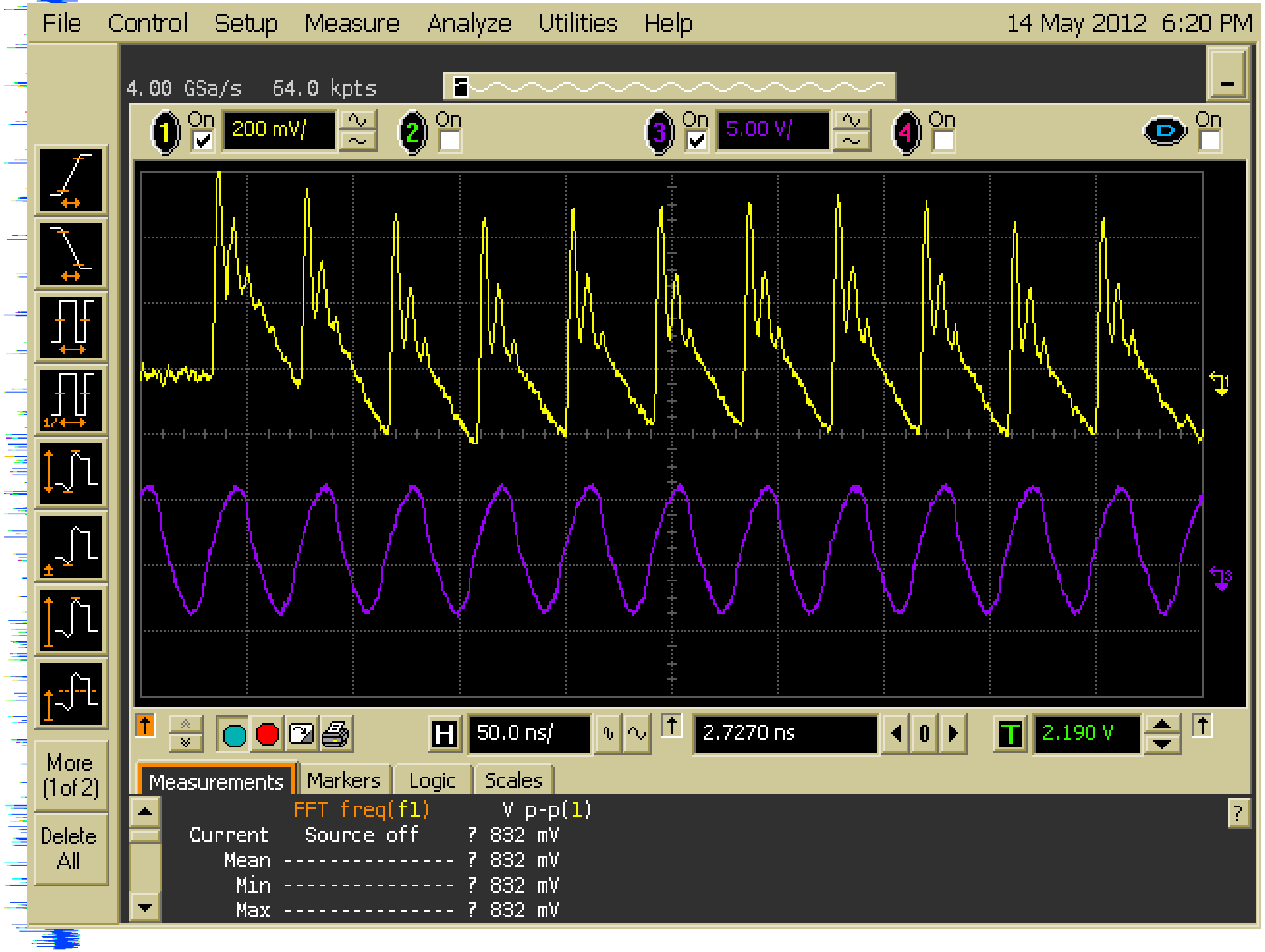
Target: Atmel AVR Running AES in C (avr-crypto-lib)



Target: Atmel AVR Running AES in C (avr-crypto-lib)


Sample Clock
Device Clock
Sample Clock
Device Clock

SASEBO-GII Example



14 May 2012 6:20 PM

4.00 GSa/s 64.0 kpts

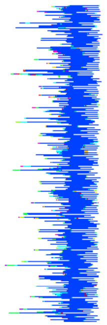
1 On ☑ 200 mV/ 2 On □ 3 On ☑ 5.00 V/ 4 On □ D On □

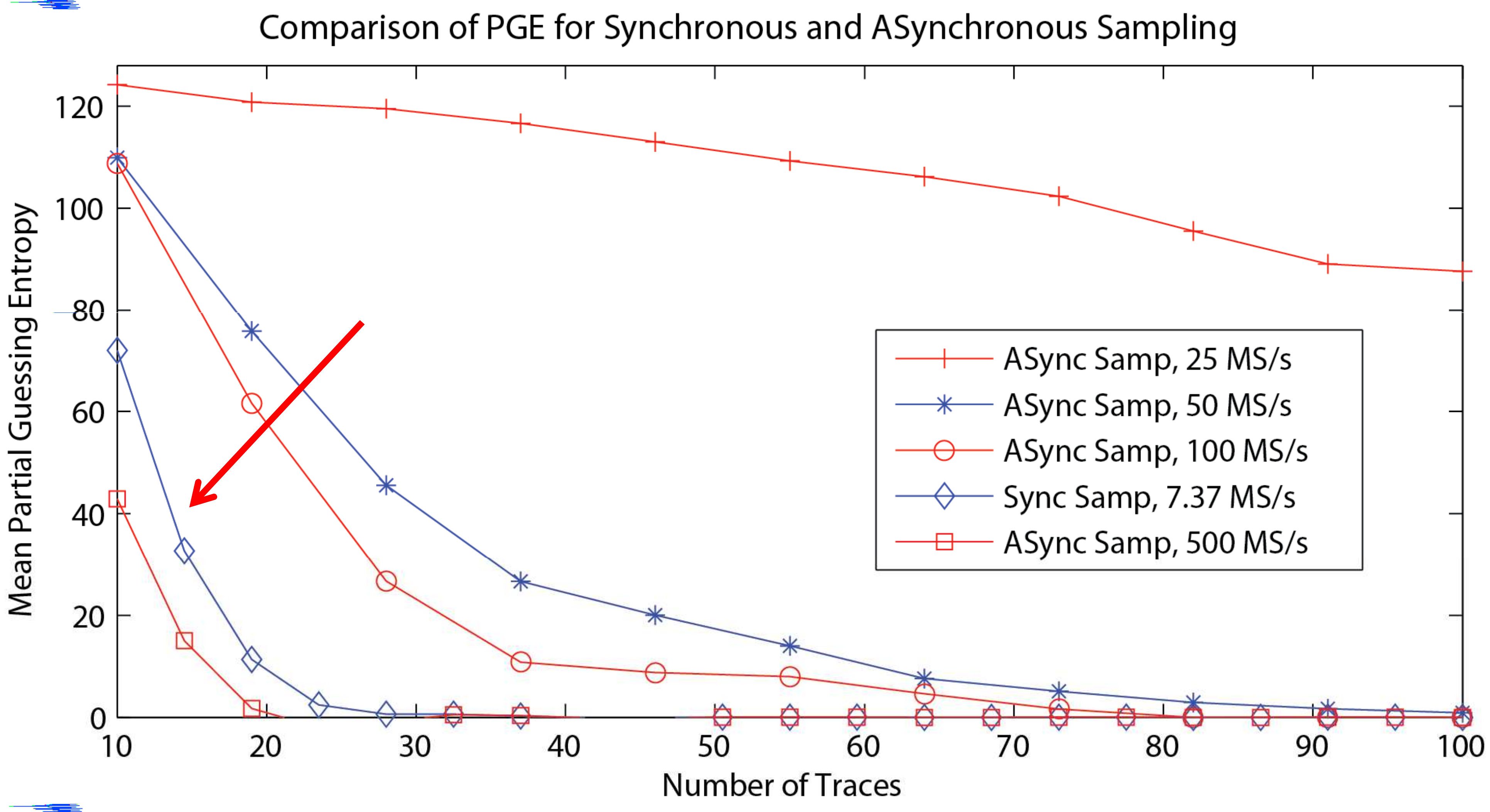
H 50.0 ns/ 2.7270 ns 0 T 2.190 V

```
              FFT freq(f1)        V p-p(1)
  Current     Source off      ? 832 mV
     Mean ---------------     ? 832 mV
      Min ---------------     ? 832 mV
      Max ---------------     ? 832 mV
```

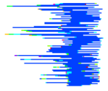
Power

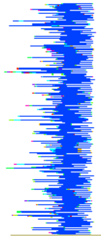
Clock

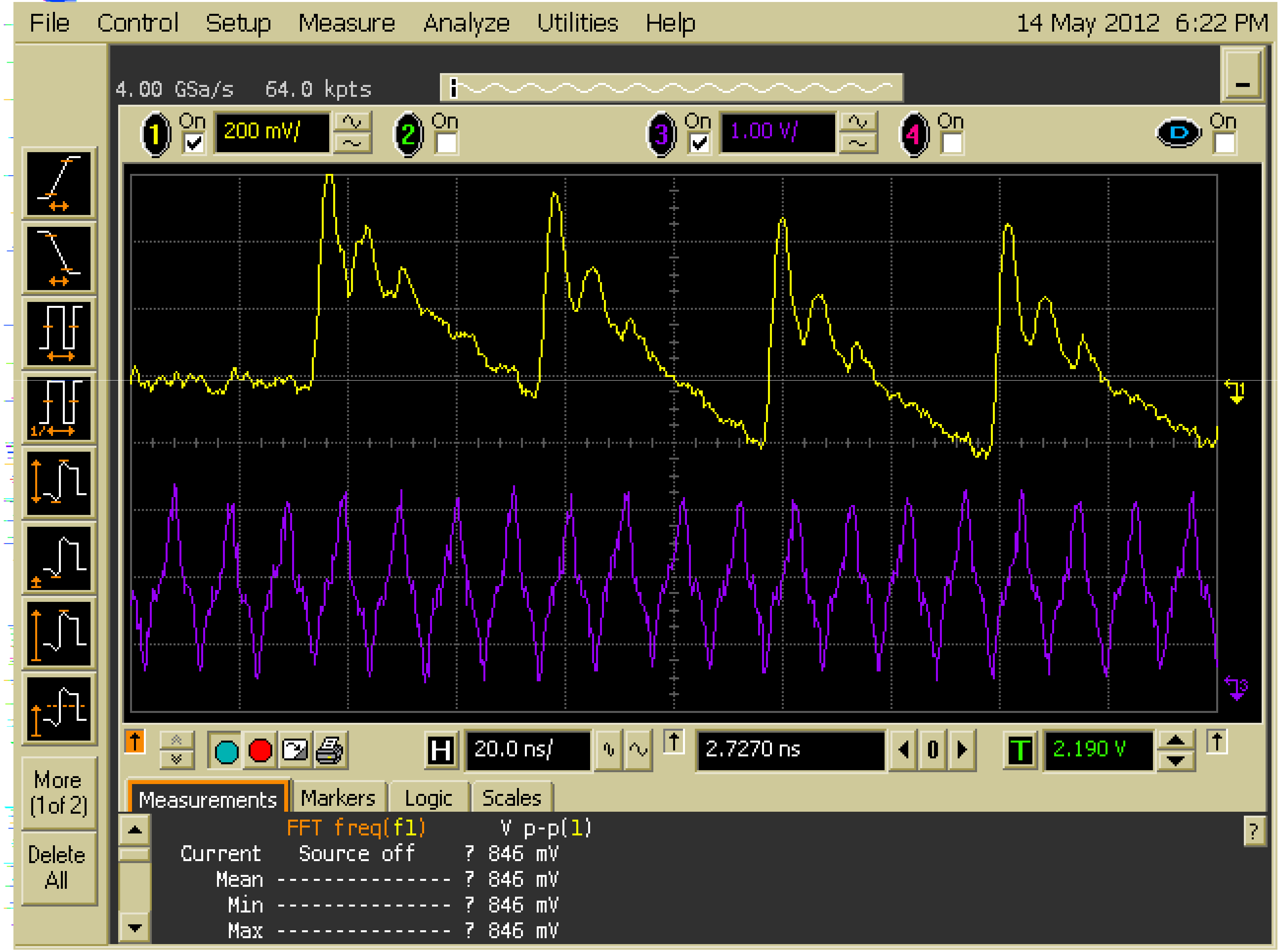




Comparison of PGE for Synchronous and ASynchronous Sampling
Mean Partial Guessing Entropy
0
20
40
60
80
100
120
Number of Traces
10
20
30
40
50
60
70
80
90
100
ASync Samp, 25 MS/s
ASync Samp, 50 MS/s
ASync Samp, 100 MS/s
Sync Samp, 7.37 MS/s
ASync Samp, 500 MS/s



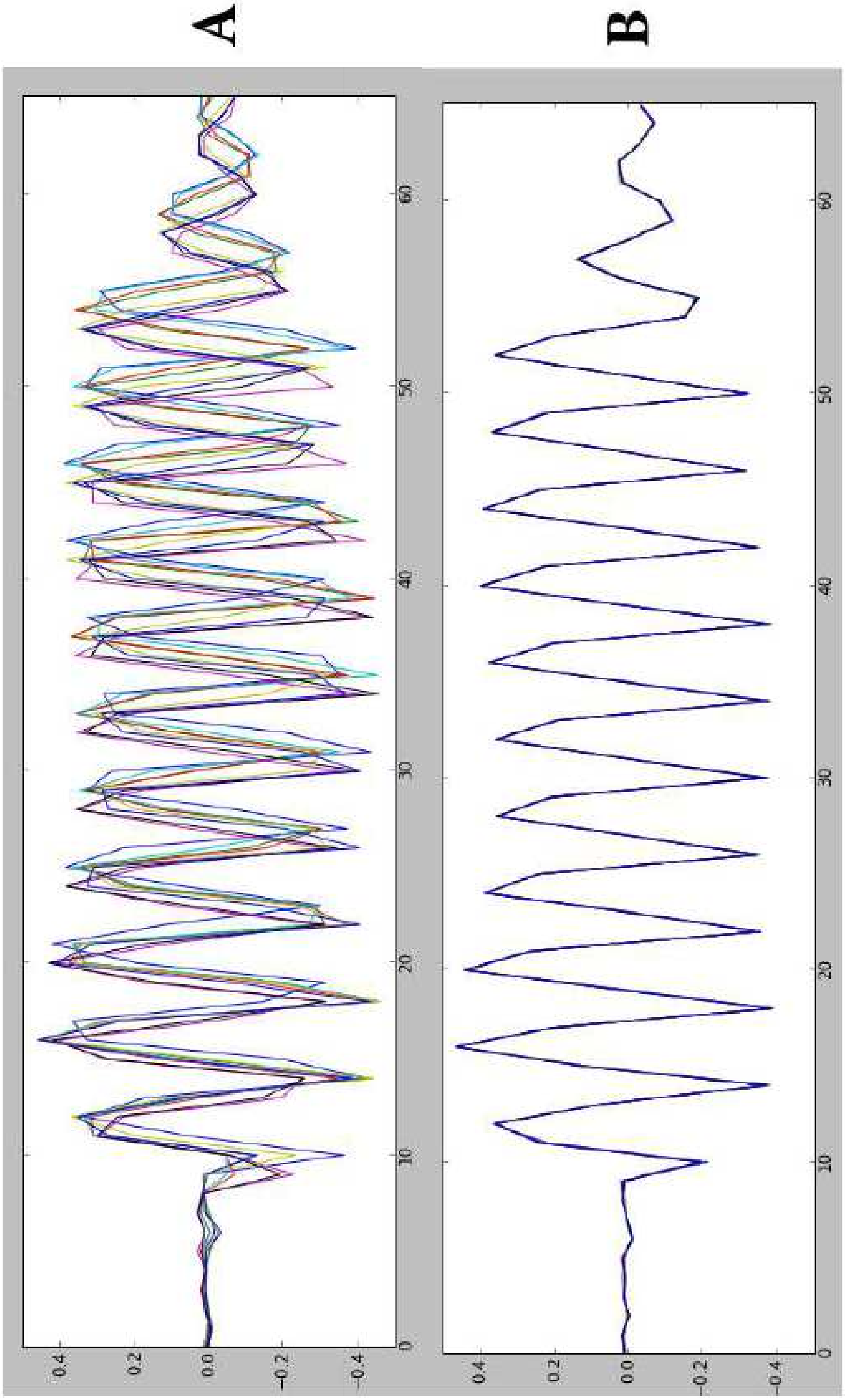


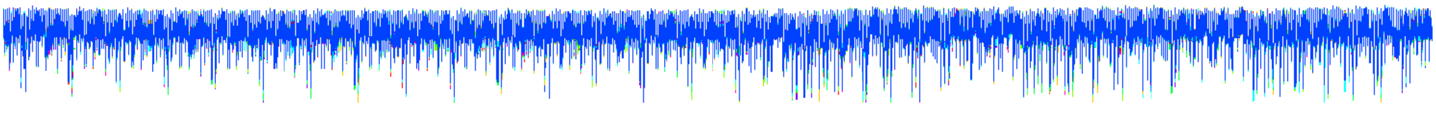


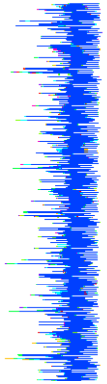
```
            FFT freq(f1)       V p-p(1)
   Current    Source off     ? 846 mV
      Mean ---------------   ? 846 mV
       Min ---------------   ? 846 mV
       Max ---------------   ? 846 mV
```

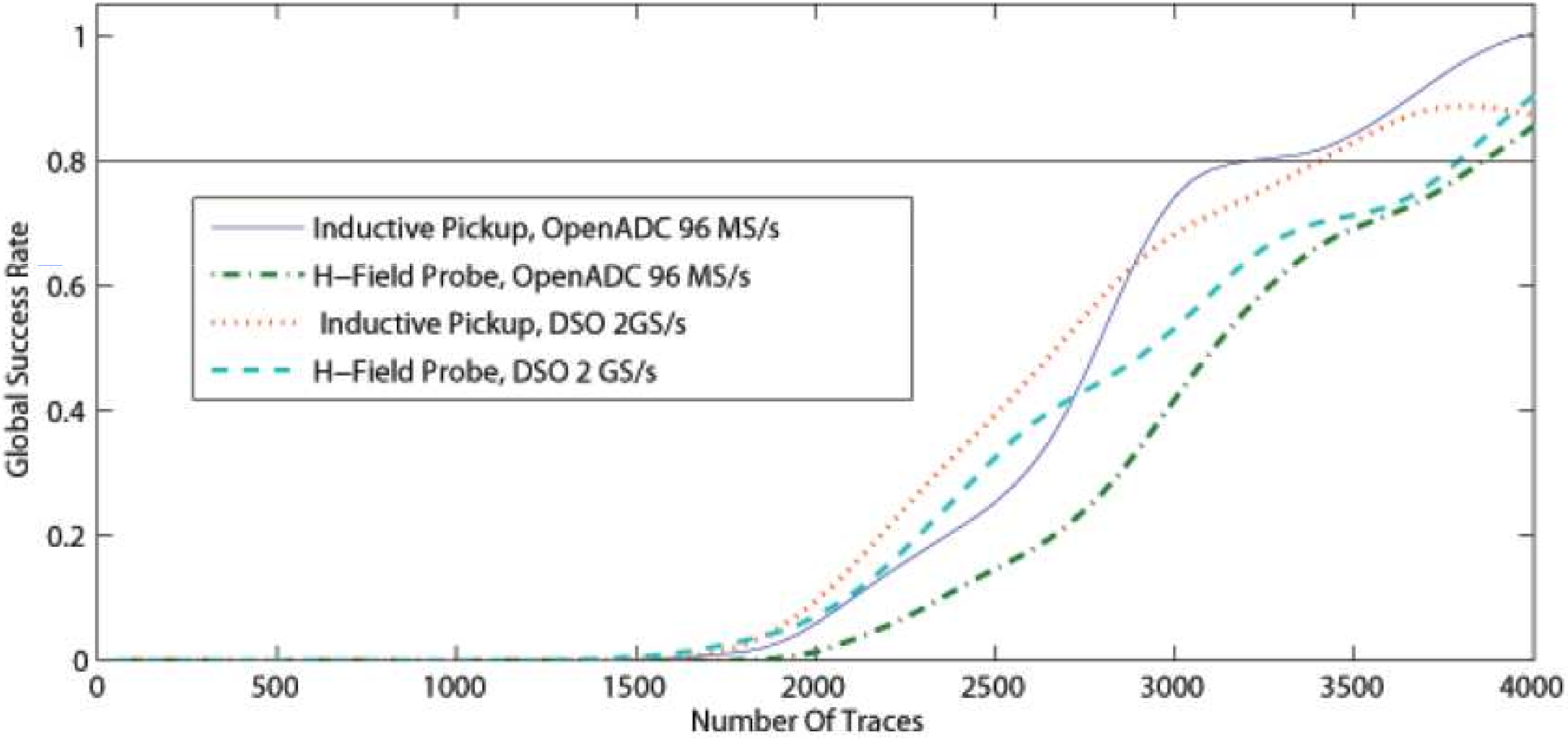
Power

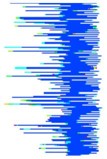
Capture Clock = 4x Device


A
B
0.4
0.2
0.0
−0.2
−0.4
0
10
20
30
40
50
60

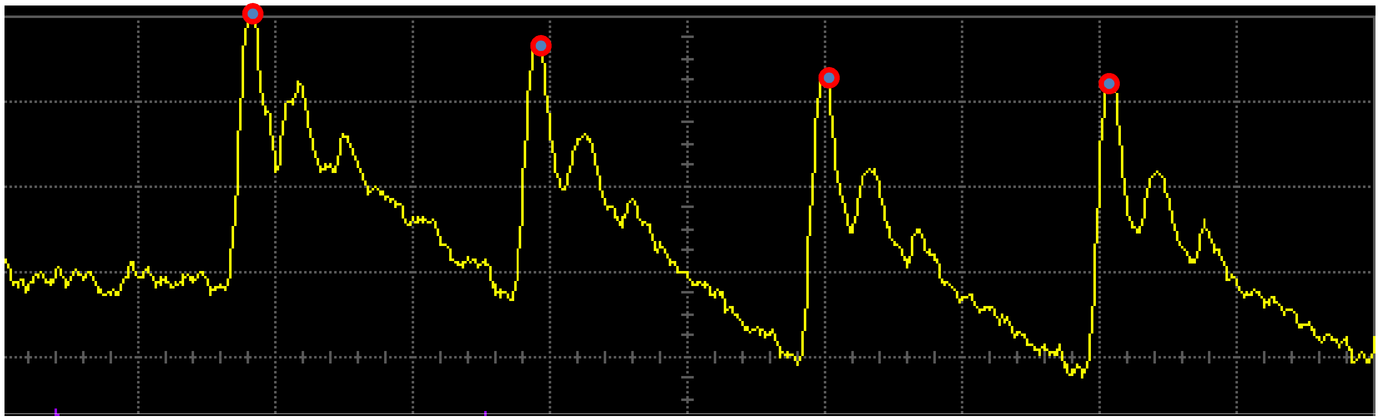


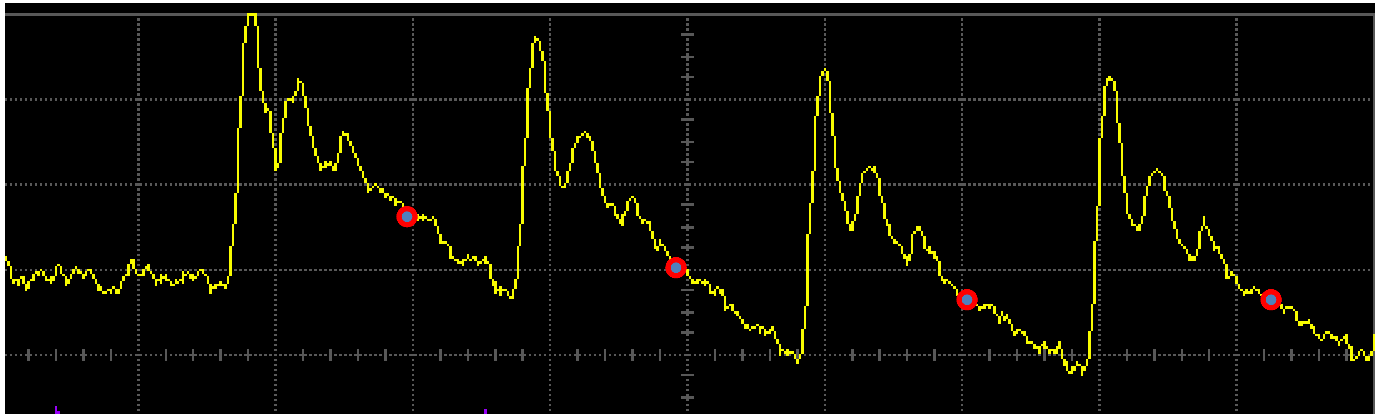



Inductive Pickup, OpenADC 96 MS/s
H–Field Probe, OpenADC 96 MS/s
Inductive Pickup, DSO 2GS/s
H–Field Probe, DSO 2 GS/s
Global Success Rate
0
0.2
0.4
0.6
0.8
1
0
500
1000
1500
2000
2500
3000
3500
4000
Number Of Traces



Phase Shift

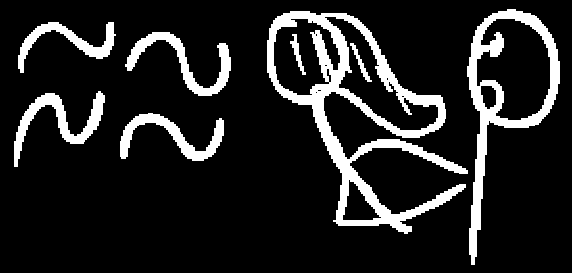


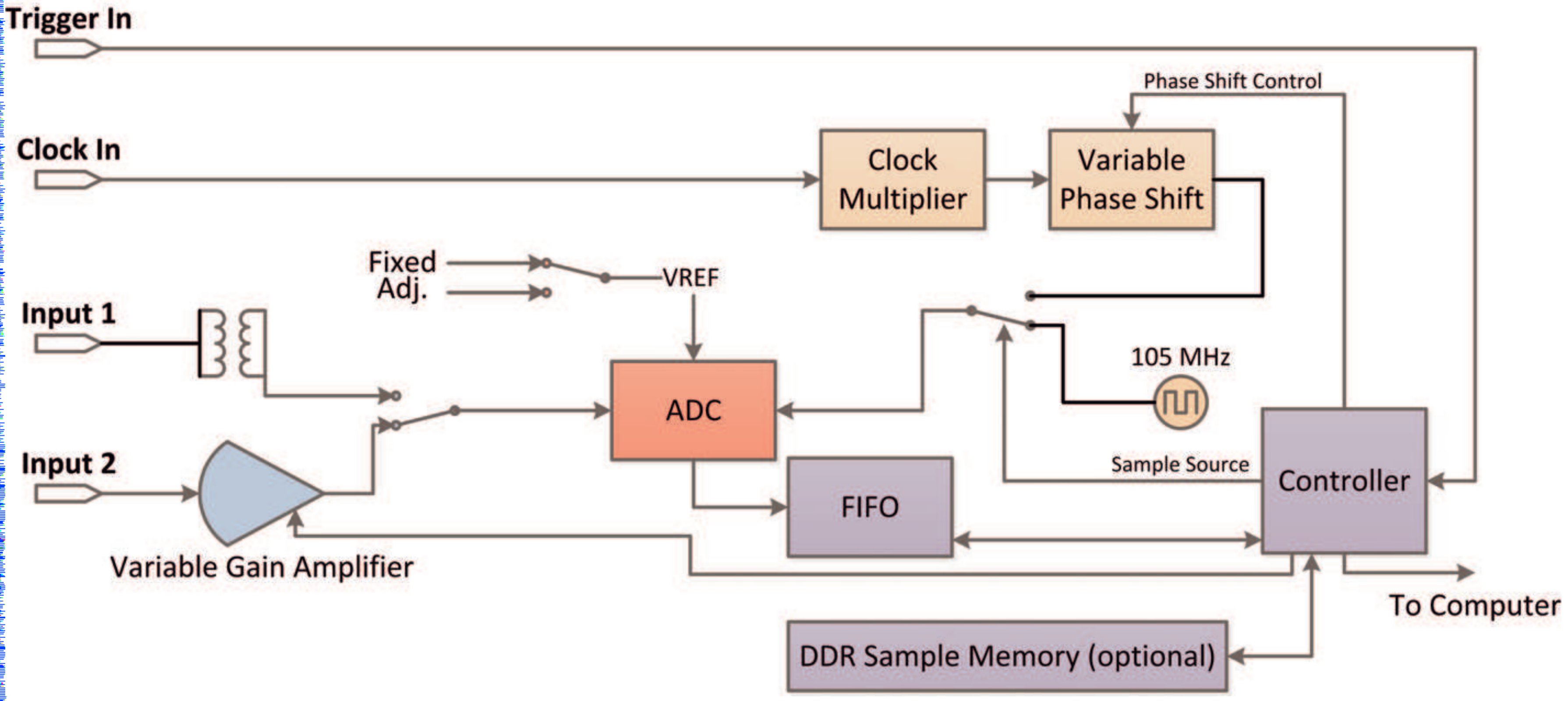


What if using a regular scope?

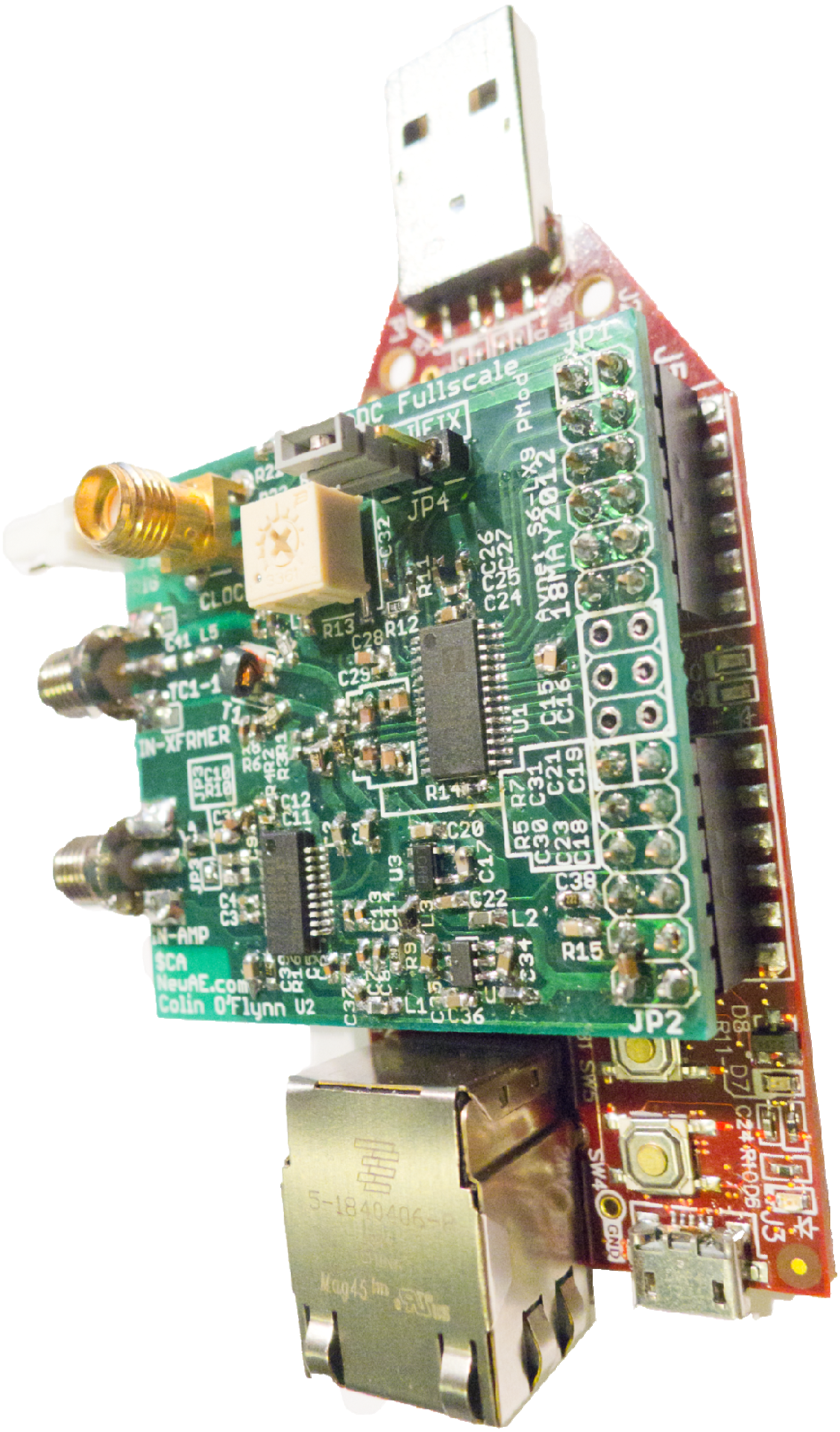
- Can hack scope to output sampling time-base, run D.U.T. from this clock or derived from this clock
- Some scopes tell you time between trigger & first sample, use this to upsample, shift offset, and downsample traces
  - Agilent calls this ‘XOffset’ parameter
- Sample at highest possible rate & downsample yourself

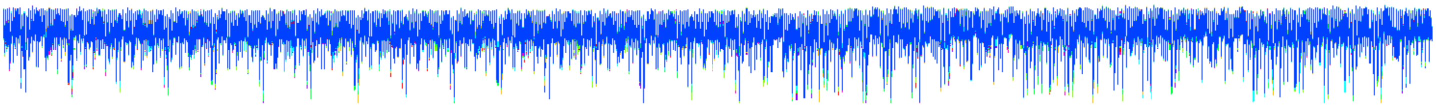
WAVEFORM ACQUISITION FOR CHEAP





See "*A Case Study of Side-Channel Analysis using Decoupling Capacitor Power Measurement with the OpenADC*" by Colin O'Flynn & Zhizhang Chen

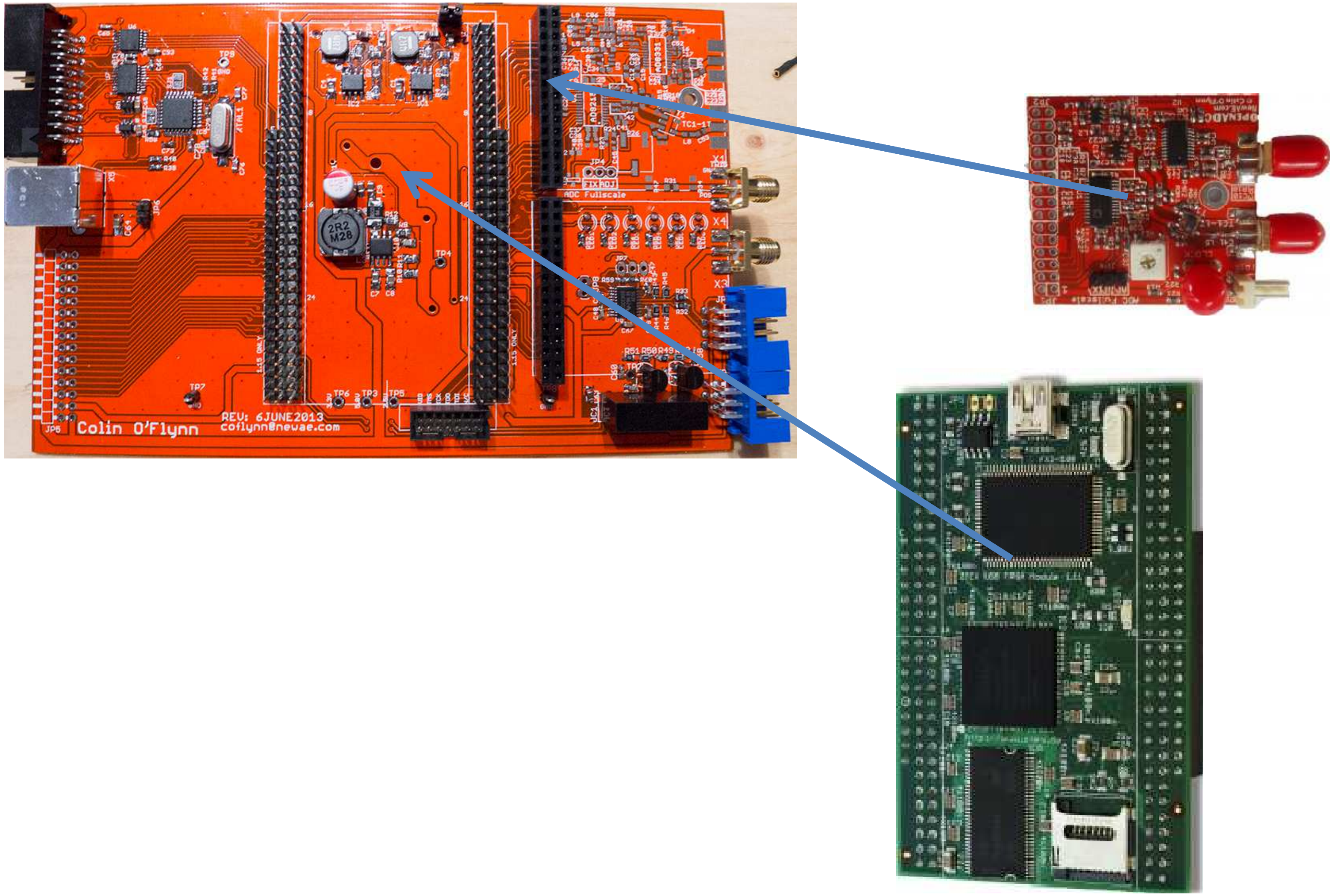

ADC Fullscale
JP1
JP4
pMod
AVNET S6-LX9
18MAY2012
CLOCK
TC1-1
IN-XFRMER
IN-AMP
SCA
NewAE.com
Colin O'Flynn V2
JP2
SW4
GND
5-1840406-8
Mag45



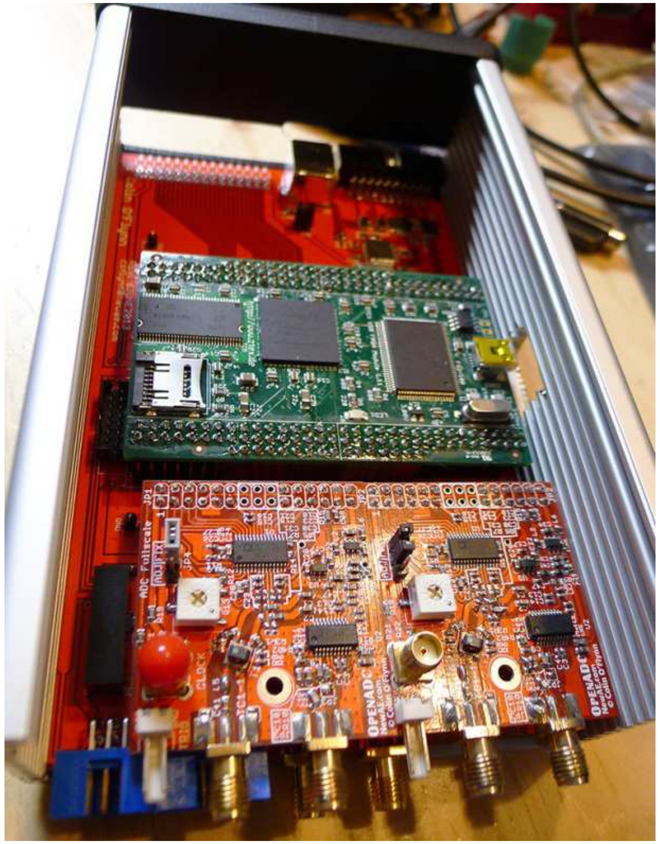
OpenADC Features (ADC Board)

- 105 MSPS, 10 bits (can be overclocked)
- Low Noise Amplifier (LNA) for adjustable -5 to +55 dB gain
  - ~120 MHz input bandwidth
- Transformer input for higher bandwidth (500MHz+)
- Clock Input

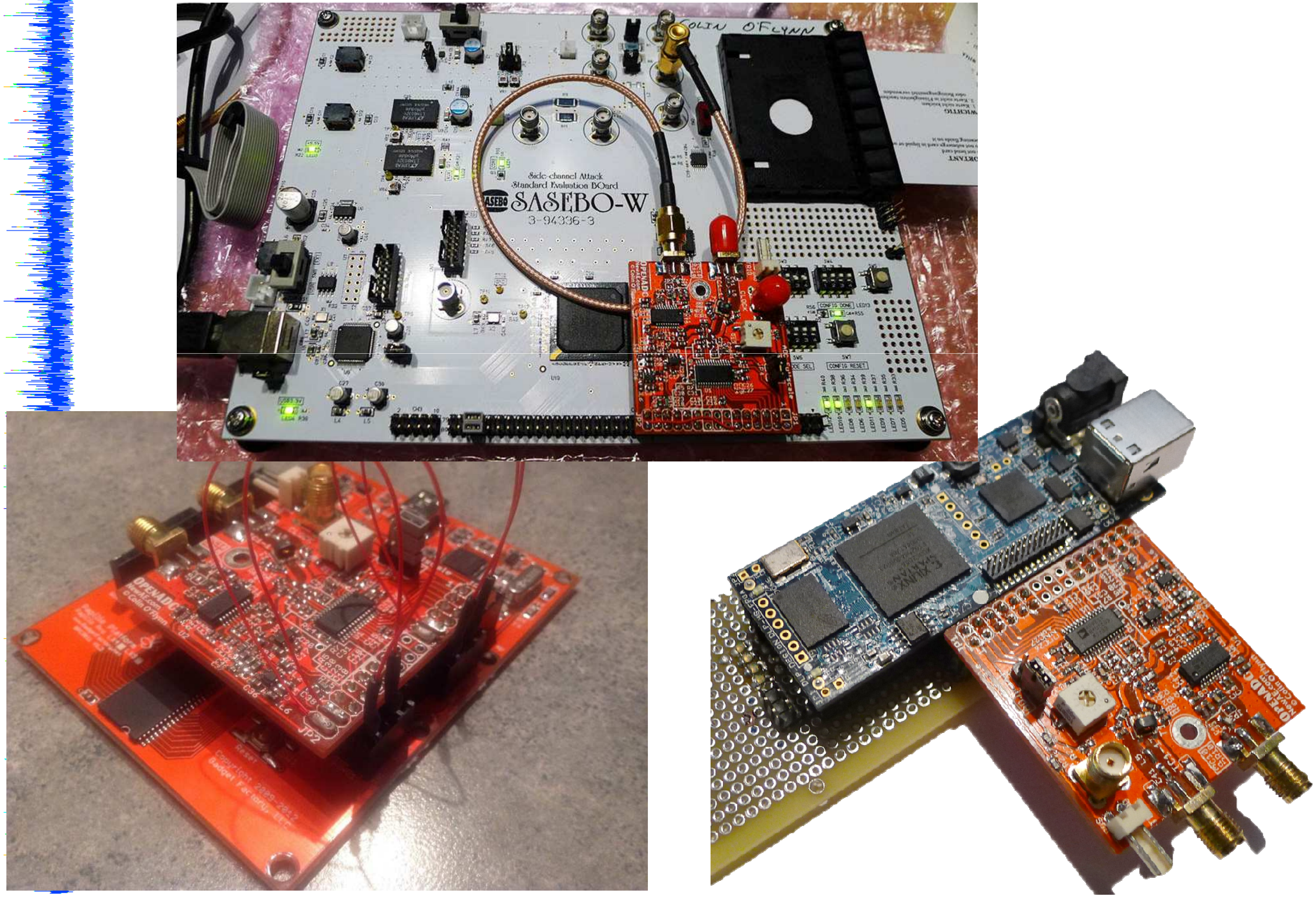
ChipWhisperer Capture v2

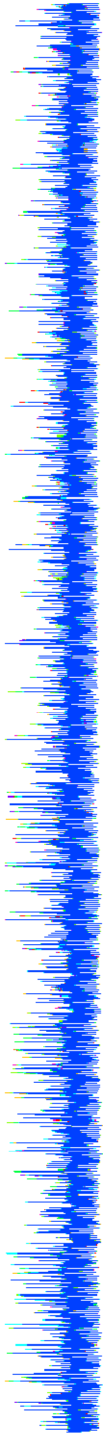


ChipWhisperer Capture v2



Other FPGAs?


COLIN O'FLYNN
Side-channel Attack
Standard Evaluation BOard
SASEBO-W
3-94336-3
OPENADC

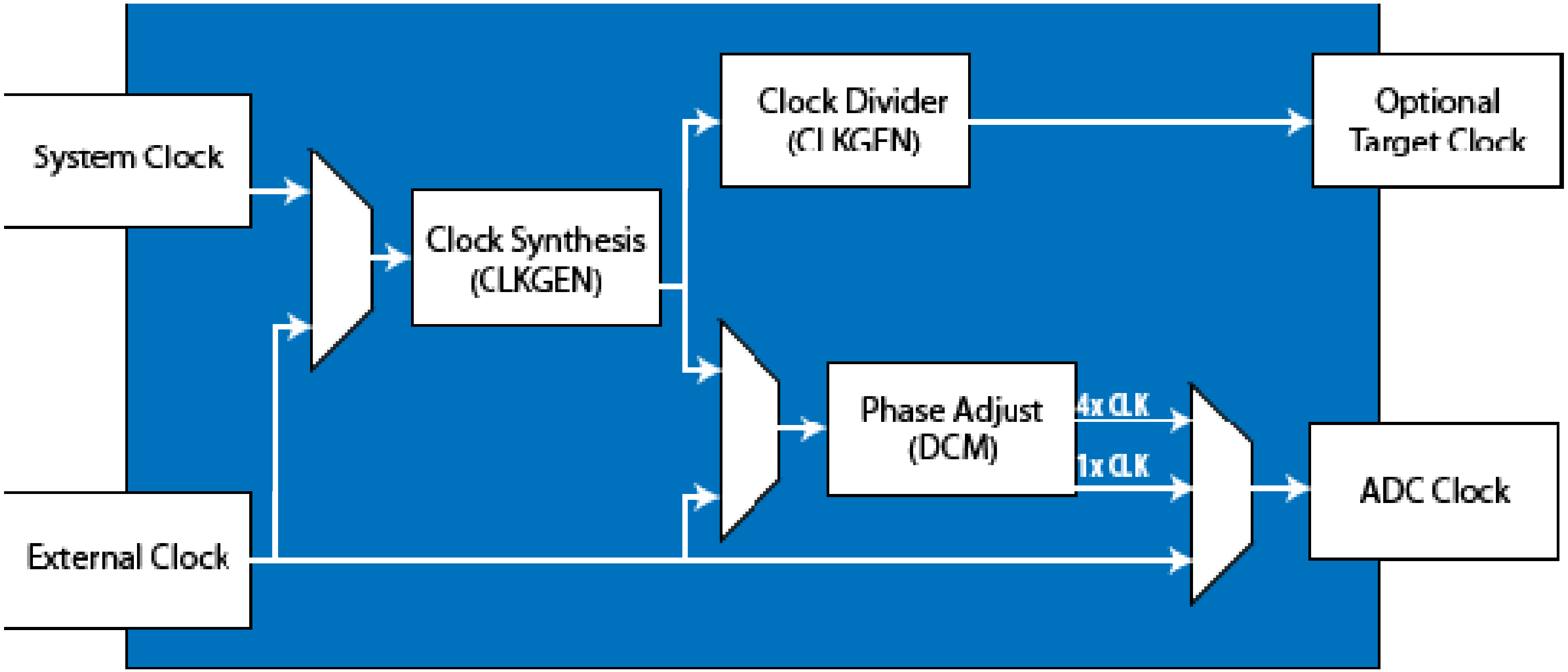


Modules available for FPGA

- Base System:
  - Cypress EZ-USB FX2 Interface (USB 2.0 high-speed)
  - FTDI FT2232H Interface (USB 2.0 high-speed)
  - Serial Interface (slow)
  - Main registers + register interface
- Clock Generator:
  - Phase Adjust
  - Clock routing
  - Phase-locked 4x generator
  - Phase-unlocked variable generator (NOT DONE YET)
  - Target clock generator
- Triggers:
  - Routing Module
  - Basic Trigger Module
  - I/O Pattern Trigger Module
  - Correlation Trigger Module (NOT DONE YET)
- Interfaces
  - Serial Interface (8-N-1 fixed baud rate)
  - SmartCard Module (basic messages only)
  - Universal Serial Interface (BETA)

Base System

Clock Generator



System Clock

External Clock

Clock Synthesis (CLKGEN)

Clock Divider (CLKGEN)

Optional Target Clock

Phase Adjust (DCM)

4x CLK

1x CLK

ADC Clock

WARNING: Confirm clocks LOCKED before operating...

Using the DCM (Phase Adjust)

- DCM provides a phase locked reference.
- Variably adjust the phase of the signal passing through this block to sample at a specific moment relative to the external clock edge.
- The DCM block provides a 1x and 4x clock

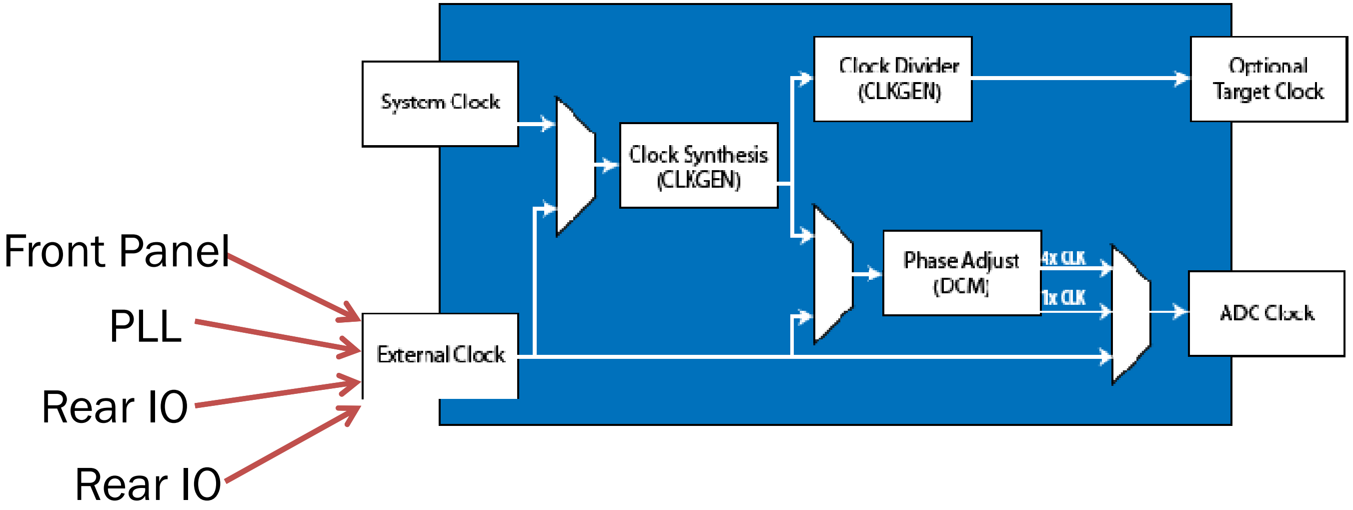
- Input Range: 5 - 250 MHz (-2 speed grade)
- Output Range (1x output): 5 - 250 MHz

Using CLKGEN

- CLKGEN block provides a clock synthesis, which can generate a range of frequencies from either the external or system clock. (NB: Not Complete)
- Be warned the CLKGEN block provides **no phase reference** between the input and output.

- **CLKGEN Output:** 5-333 MHz (-2 speed grade)
- **CLKGEN Input:** 0.5 - 333 MHz (-2 speed grade)

Total Clocking System



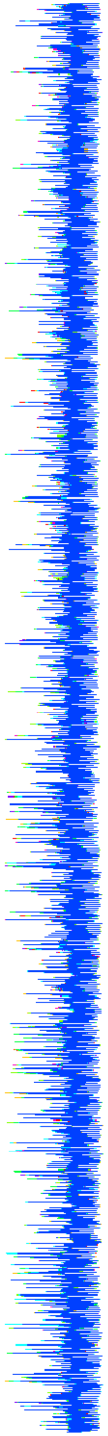
Front Panel

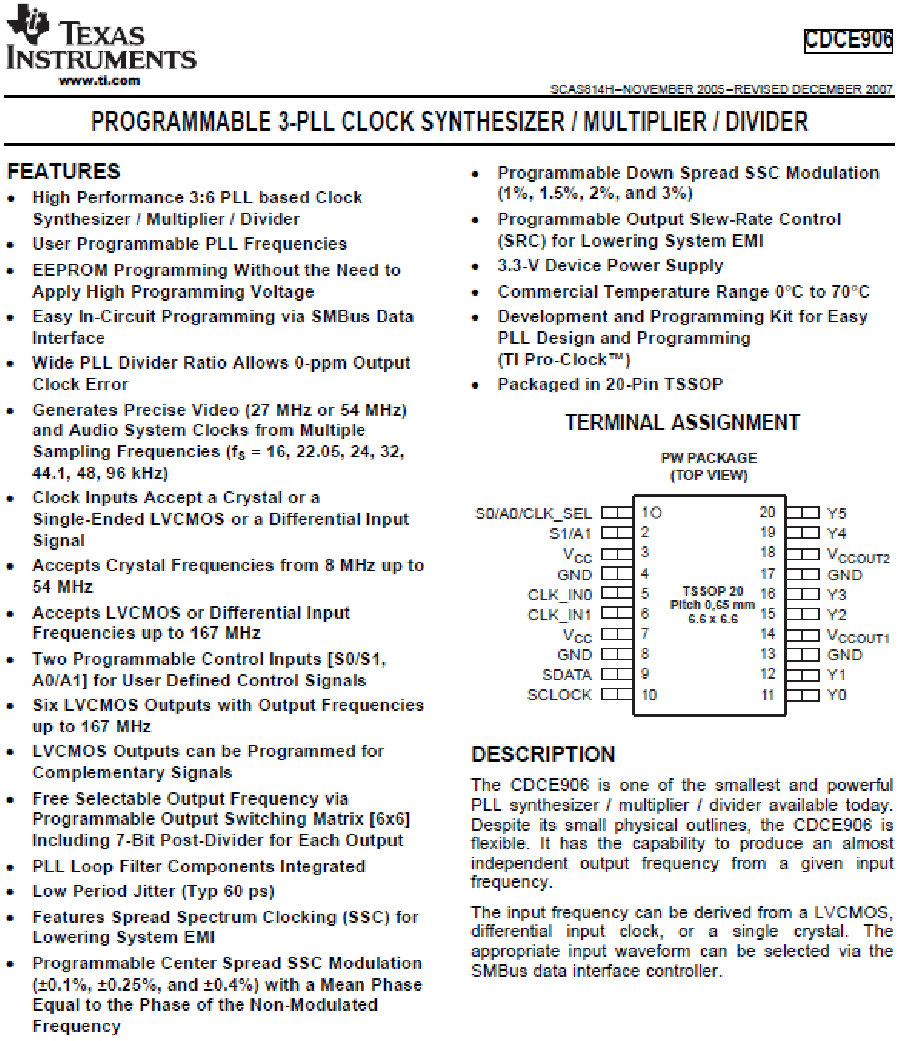
PLL

Rear IO

Rear IO

PLL Input





TEXAS INSTRUMENTS
www.ti.com

CDCE906

SCAS814H–NOVEMBER 2005–REVISED DECEMBER 2007

PROGRAMMABLE 3-PLL CLOCK SYNTHESIZER / MULTIPLIER / DIVIDER

FEATURES

- High Performance 3:6 PLL based Clock Synthesizer / Multiplier / Divider
- User Programmable PLL Frequencies
- EEPROM Programming Without the Need to Apply High Programming Voltage
- Easy In-Circuit Programming via SMBus Data Interface
- Wide PLL Divider Ratio Allows 0-ppm Output Clock Error
- Generates Precise Video (27 MHz or 54 MHz) and Audio System Clocks from Multiple Sampling Frequencies (f_S = 16, 22.05, 24, 32, 44.1, 48, 96 kHz)
- Clock Inputs Accept a Crystal or a Single-Ended LVCMOS or a Differential Input Signal
- Accepts Crystal Frequencies from 8 MHz up to 54 MHz
- Accepts LVCMOS or Differential Input Frequencies up to 167 MHz
- Two Programmable Control Inputs [S0/S1, A0/A1] for User Defined Control Signals
- Six LVCMOS Outputs with Output Frequencies up to 167 MHz
- LVCMOS Outputs can be Programmed for Complementary Signals
- Free Selectable Output Frequency via Programmable Output Switching Matrix [6x6] Including 7-Bit Post-Divider for Each Output
- PLL Loop Filter Components Integrated
- Low Period Jitter (Typ 60 ps)
- Features Spread Spectrum Clocking (SSC) for Lowering System EMI
- Programmable Center Spread SSC Modulation (±0.1%, ±0.25%, and ±0.4%) with a Mean Phase Equal to the Phase of the Non-Modulated Frequency
- Programmable Down Spread SSC Modulation (1%, 1.5%, 2%, and 3%)
- Programmable Output Slew-Rate Control (SRC) for Lowering System EMI
- 3.3-V Device Power Supply
- Commercial Temperature Range 0°C to 70°C
- Development and Programming Kit for Easy PLL Design and Programming (TI Pro-Clock™)
- Packaged in 20-Pin TSSOP

TERMINAL ASSIGNMENT

PW PACKAGE
(TOP VIEW)

TSSOP 20
Pitch 0,65 mm
6.6 x 6.6

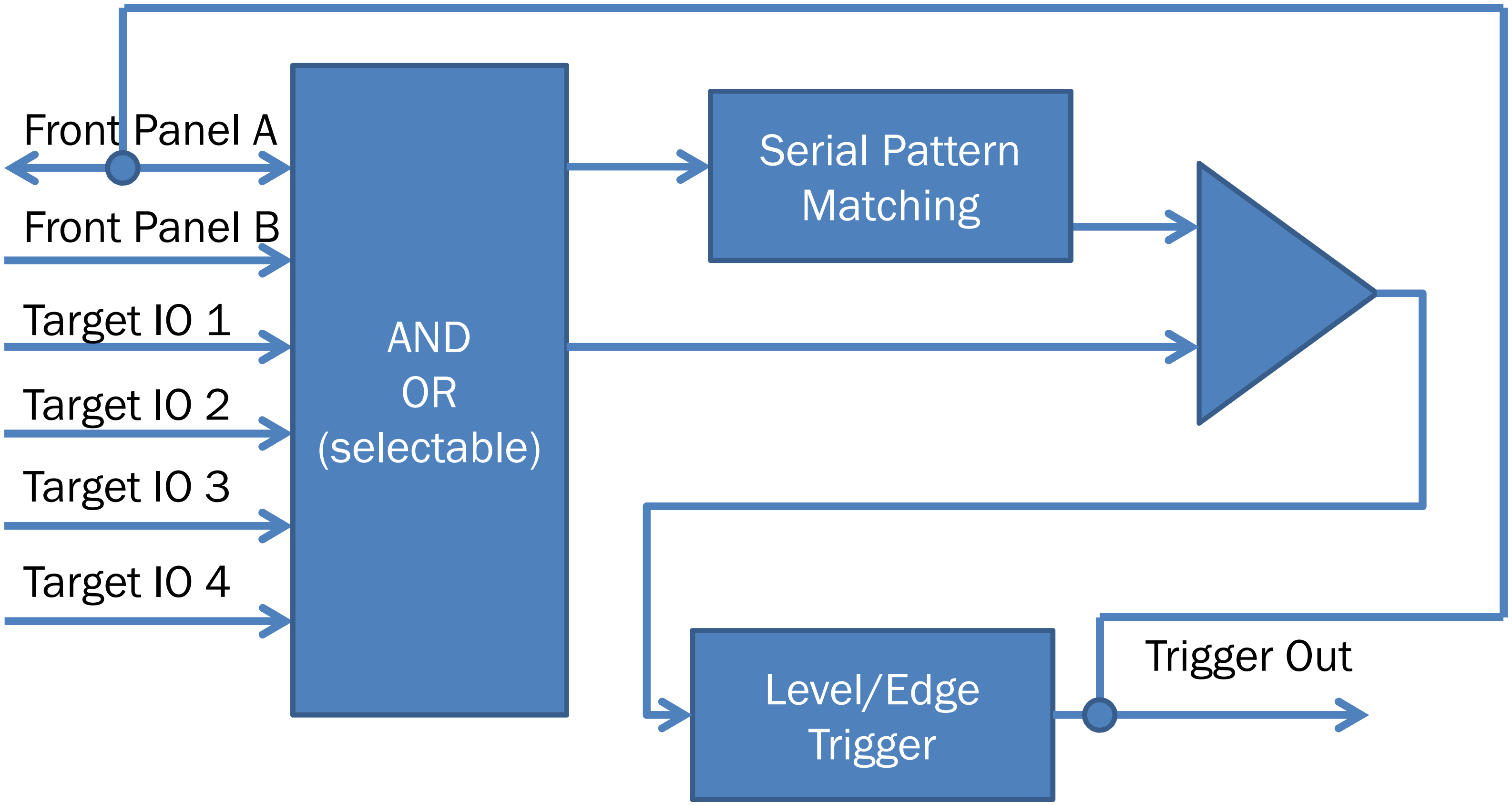
S0/A0/CLK_SEL	1	20	Y5
S1/A1	2	19	Y4
V_{CC}	3	18	V_{CCOUT2}
GND	4	17	GND
CLK_IN0	5	16	Y3
CLK_IN1	6	15	Y2
V_{CC}	7	14	V_{CCOUT1}
GND	8	13	GND
SDATA	9	12	Y1
SCLOCK	10	11	Y0

DESCRIPTION

The CDCE906 is one of the smallest and powerful PLL synthesizer / multiplier / divider available today. Despite its small physical outlines, the CDCE906 is flexible. It has the capability to produce an almost independent output frequency from a given input frequency.

The input frequency can be derived from a LVCMOS, differential input clock, or a single crystal. The appropriate input waveform can be selected via the SMBus data interface controller.

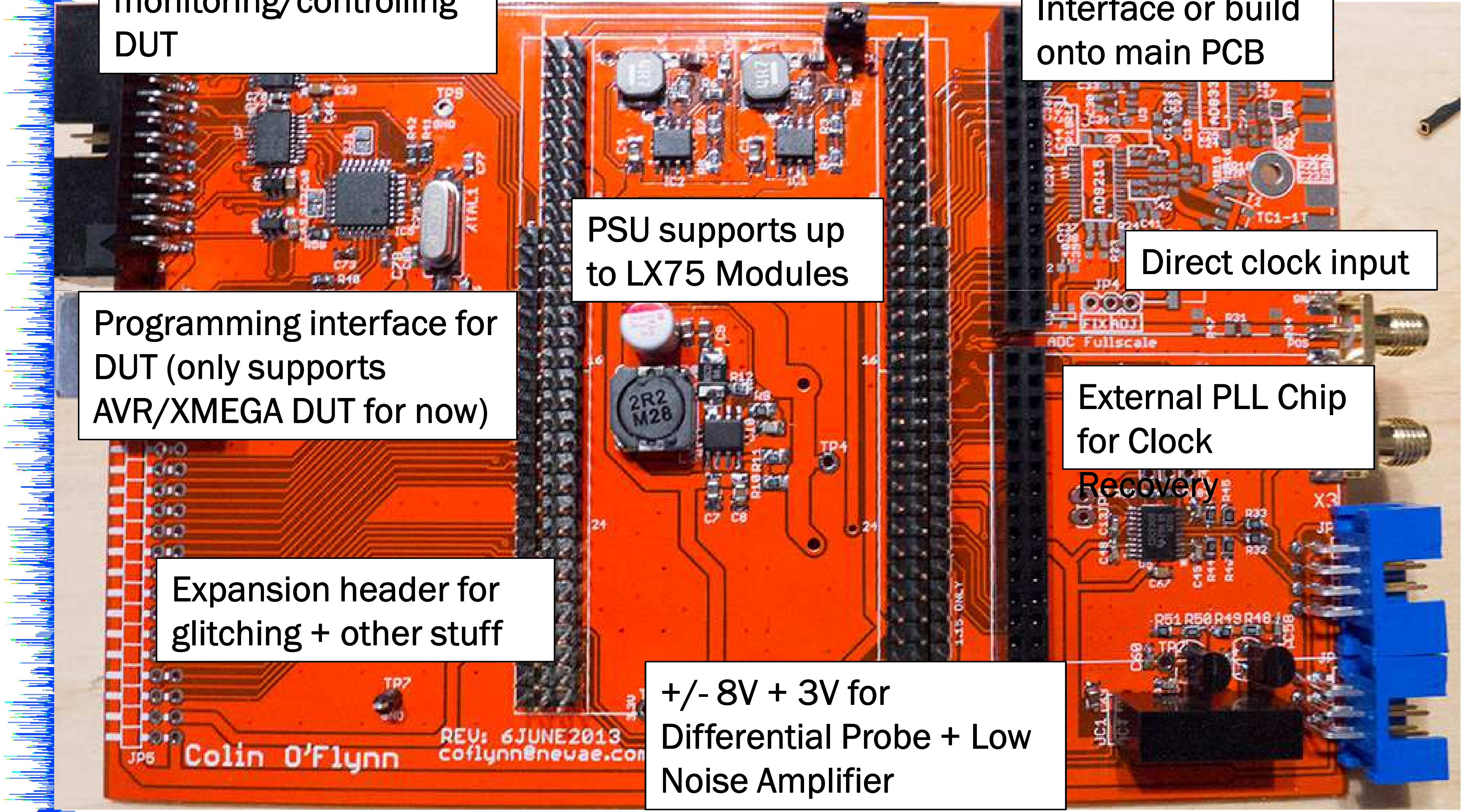
Trigger Routing



Use of AND/OR

Interfaces

ChipWhisperer Capture Rev2 Hardware



Bidirectional voltage translators for monitoring/controlling DUT

OpenADC Interface or build onto main PCB

PSU supports up to LX75 Modules

Direct clock input

Programming interface for DUT (only supports AVR/XMEGA DUT for now)

External PLL Chip for Clock Recovery

Expansion header for glitching + other stuff

+/- 8V + 3V for Differential Probe + Low Noise Amplifier

WHY HAVE A STANDARD?

- Accessible, Open Hardware
  - Easily modifiable for special probes
  - Beyond side-channel, can be used for glitch attacks
  - High Performance
- Accessible, Open Software
  - Easy to get new participants quickly “up to speed”

ATTACKING PRACTICAL SYSTEMS

Getting the Clock

Varying Clocks

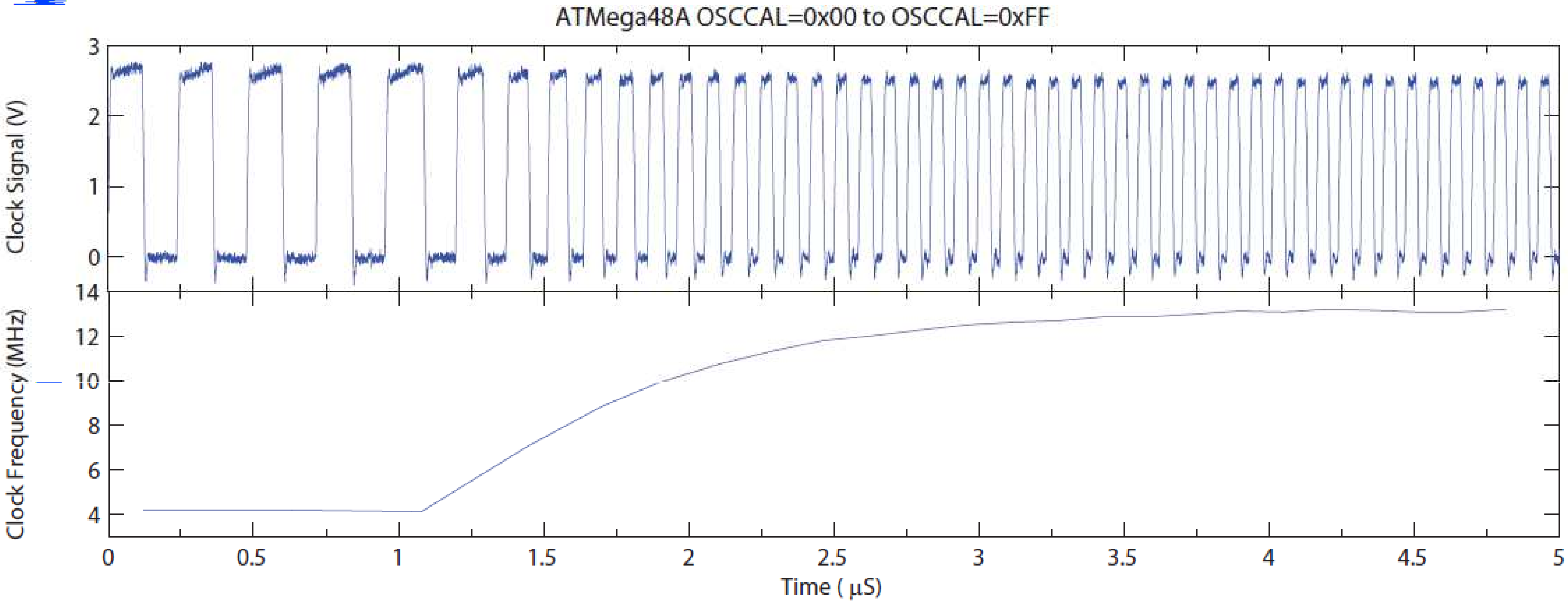


Fig. 3. Atmel AtMega48A internal clock frequency change as OSCCAL changes from 0 to 255.

O'Flynn, C. and Chen, Z. Synchronous Sampling and Clock Recovery of Internal Oscillators for Side Channel Analysis. Cryptography ePrint Archive Report 2013/294

Varying Clocks

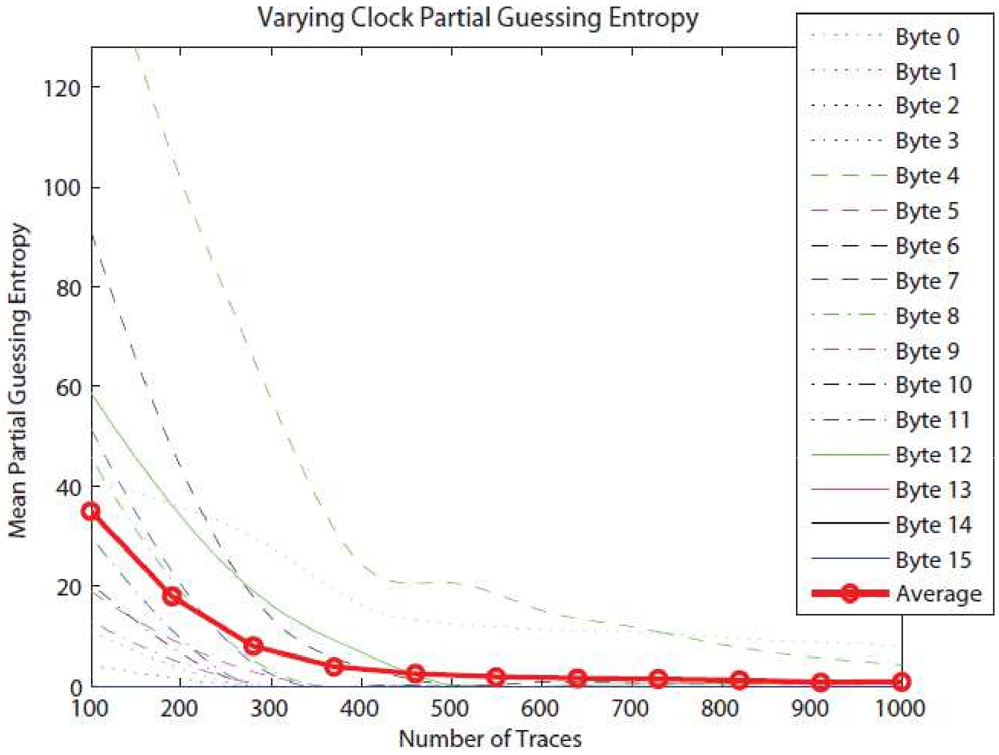
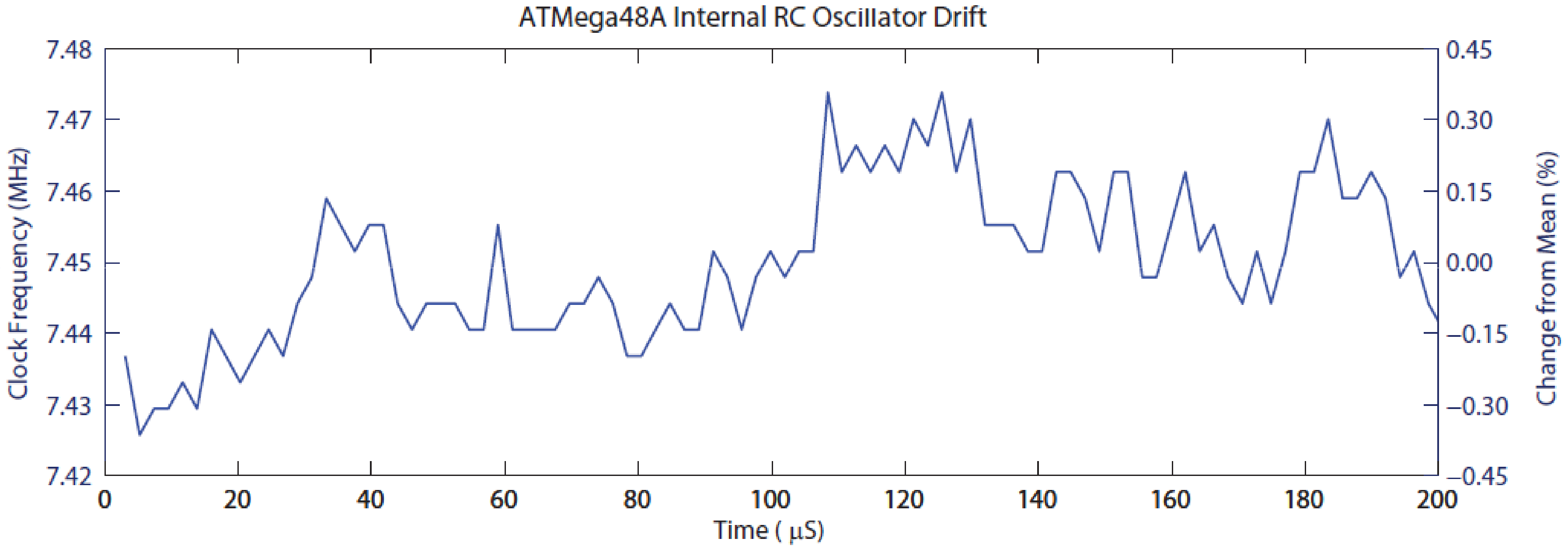


Fig. 5. Results of a CPA attack on a device with oscillator frequency randomly varying between 4.5 MHz–12.7 MHz on each encryption, and no trace synchronization being performed.

O'Flynn, C. and Chen, Z. Synchronous Sampling and Clock Recovery of Internal Oscillators for Side Channel Analysis. Cryptography ePrint Archive Report 2013/294

Internal Oscillators



O'Flynn, C. and Chen, Z. Synchronous Sampling and Clock Recovery of Internal Oscillators for Side Channel Analysis. Cryptography ePrint Archive Report 2013/294

Clock Recovery

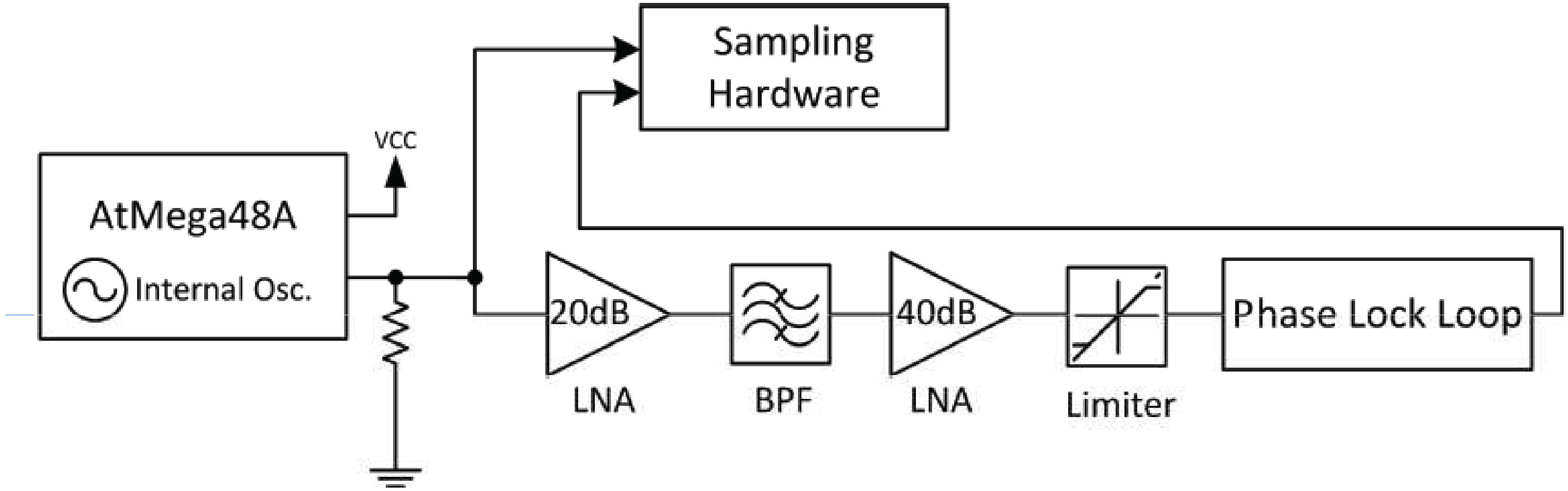
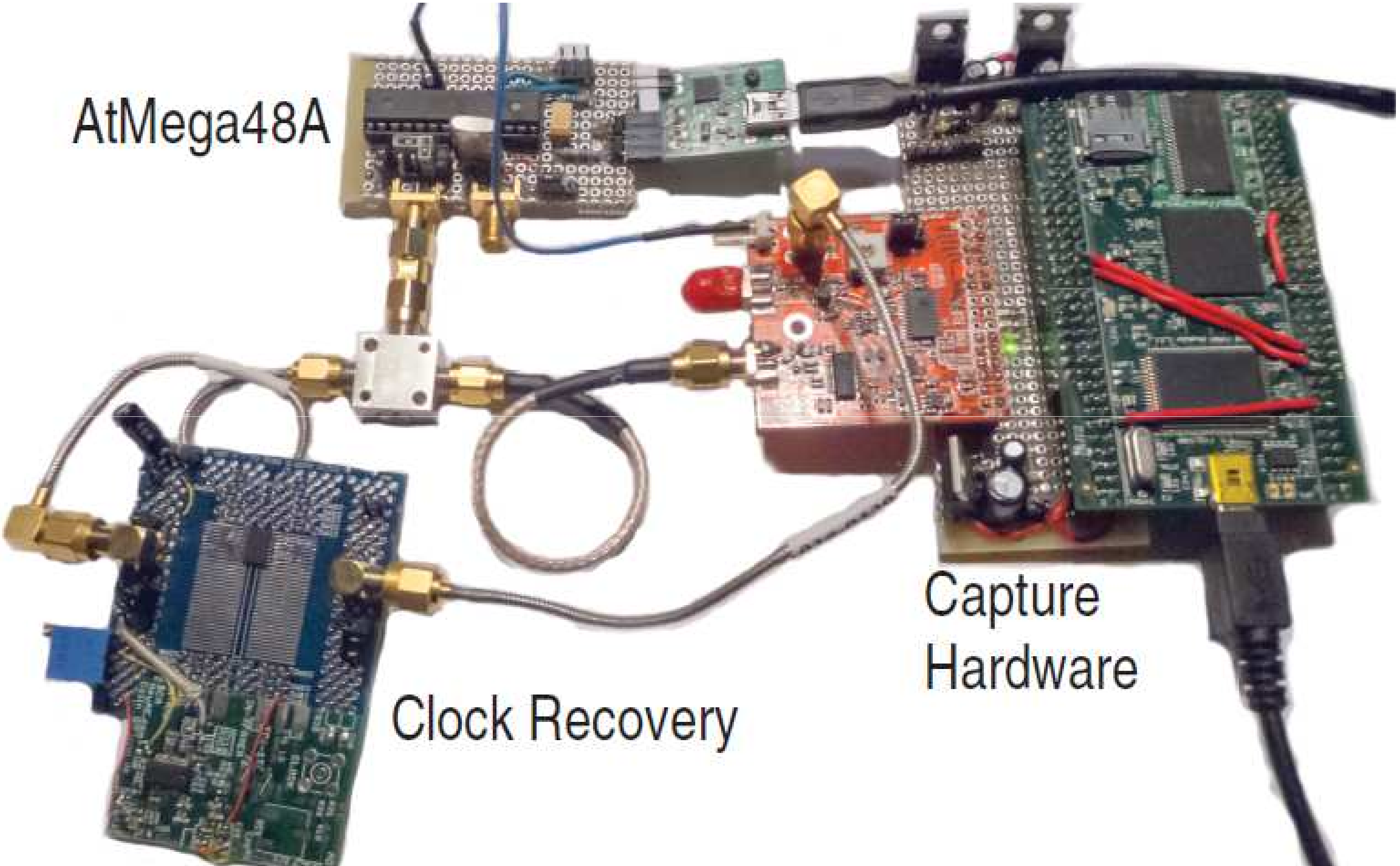


Fig. 6. Clock Recovery Block Diagram.

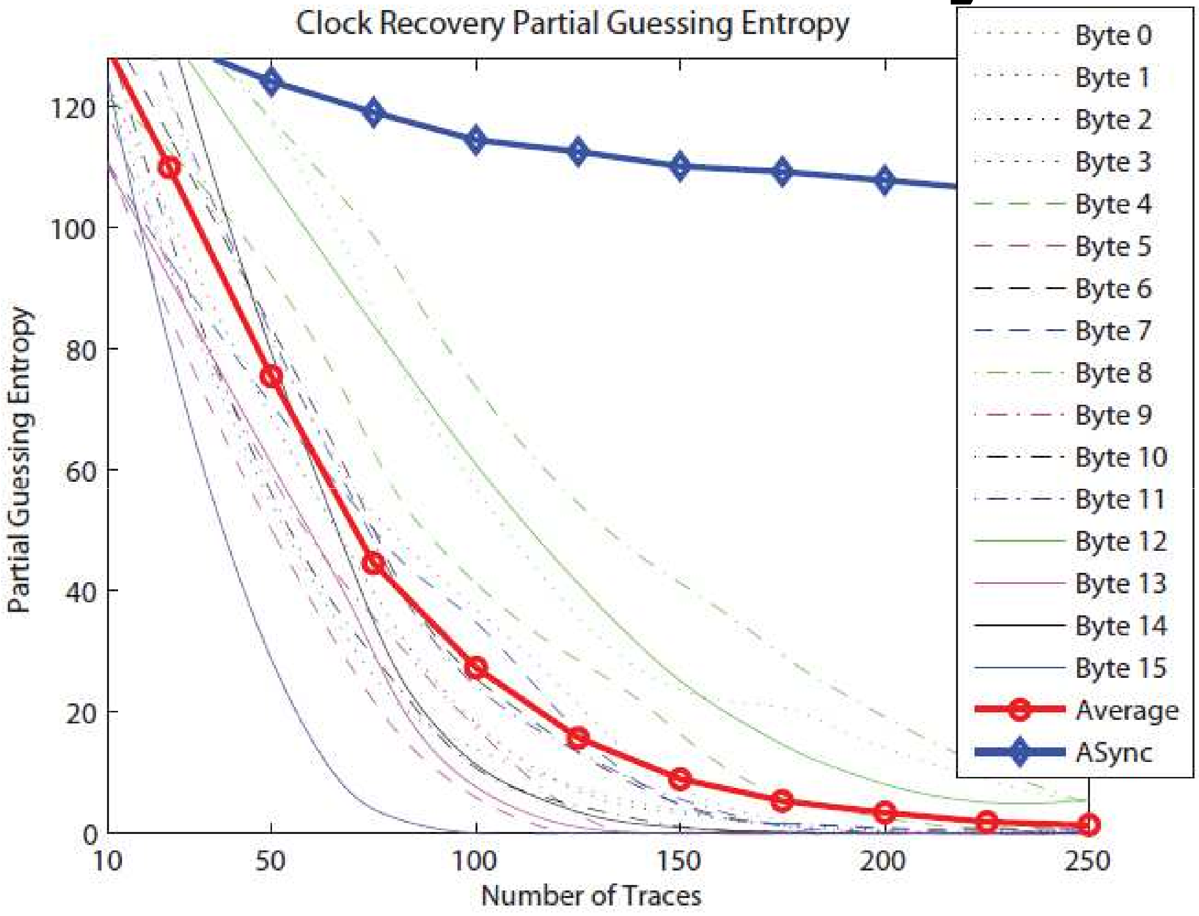
O'Flynn, C. and Chen, Z. Synchronous Sampling and Clock Recovery of Internal Oscillators for Side Channel Analysis. Cryptography ePrint Archive Report 2013/294

Clock Recovery



O'Flynn, C. and Chen, Z. Synchronous Sampling and Clock Recovery of Internal Oscillators for Side Channel Analysis. Cryptography ePrint Archive Report 2013/294

Clock Recovery



O'Flynn, C. and Chen, Z. Synchronous Sampling and Clock Recovery of Internal Oscillators for Side Channel Analysis. Cryptography ePrint Archive Report 2013/294

Clock Recovery

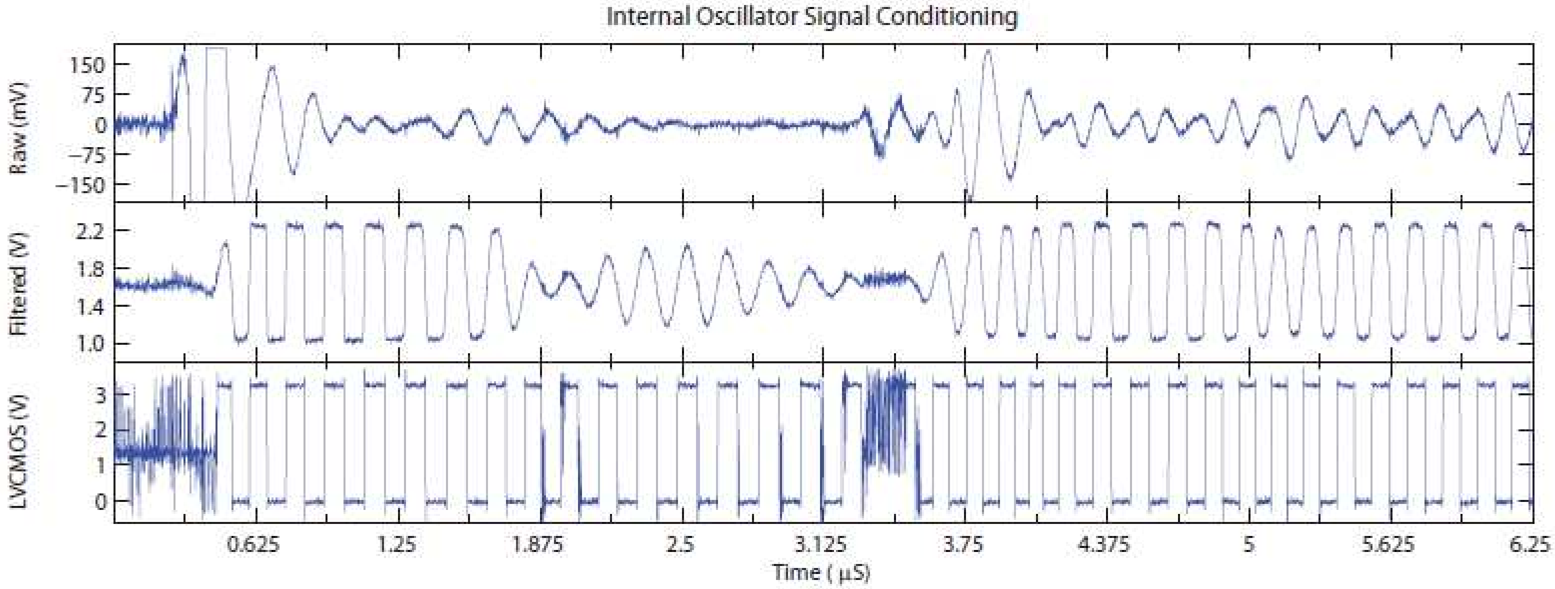
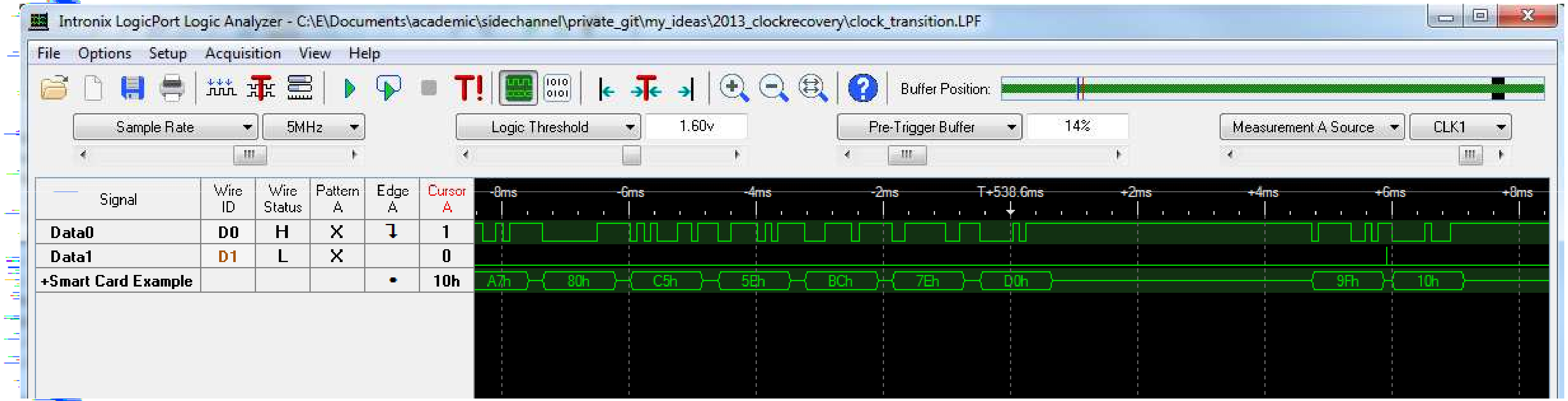


Fig. 7. Recovery of 1.3 MHz Internal RC Oscillator on KeeLoq single-chip hardware.

Trigger Timing



Intronix LogicPort Logic Analyzer - C:\E\Documents\academic\sidechannel\private_git\my_ideas\2013_clockrecovery\clock_transition.LPF

Buffer Position:

Sample Rate: 5MHz

Logic Threshold: 1.60v

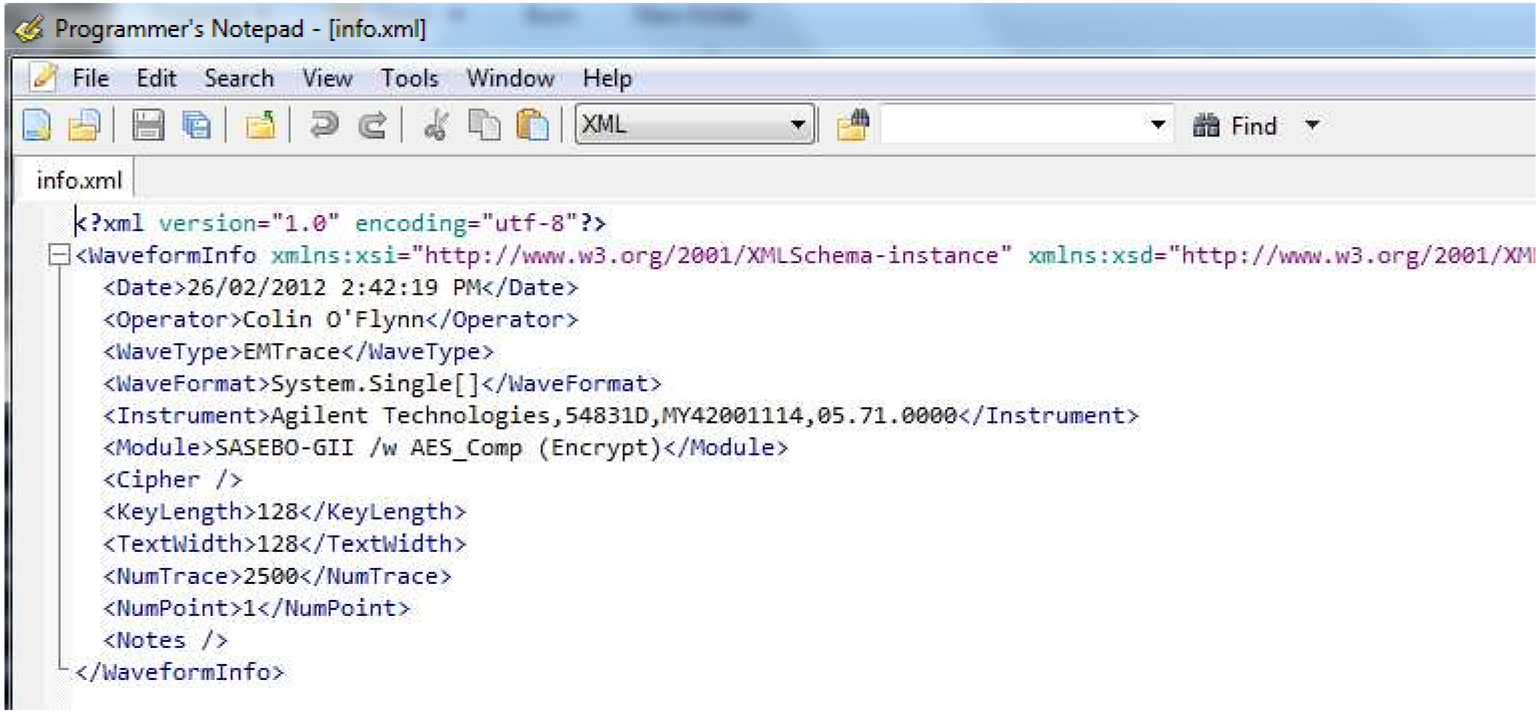
Pre-Trigger Buffer: 14%

Measurement A Source: CLK1

Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data0	D0	H	X	↴	1
Data1	D1	L	X		0
+Smart Card Example				•	10h

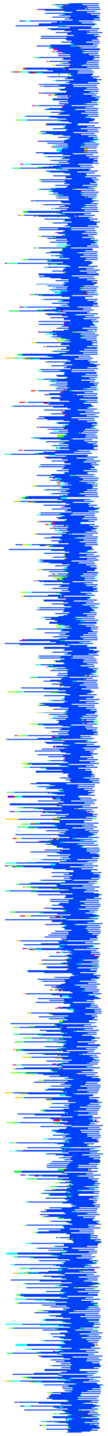
Does Sample Rate = Clock Rate?

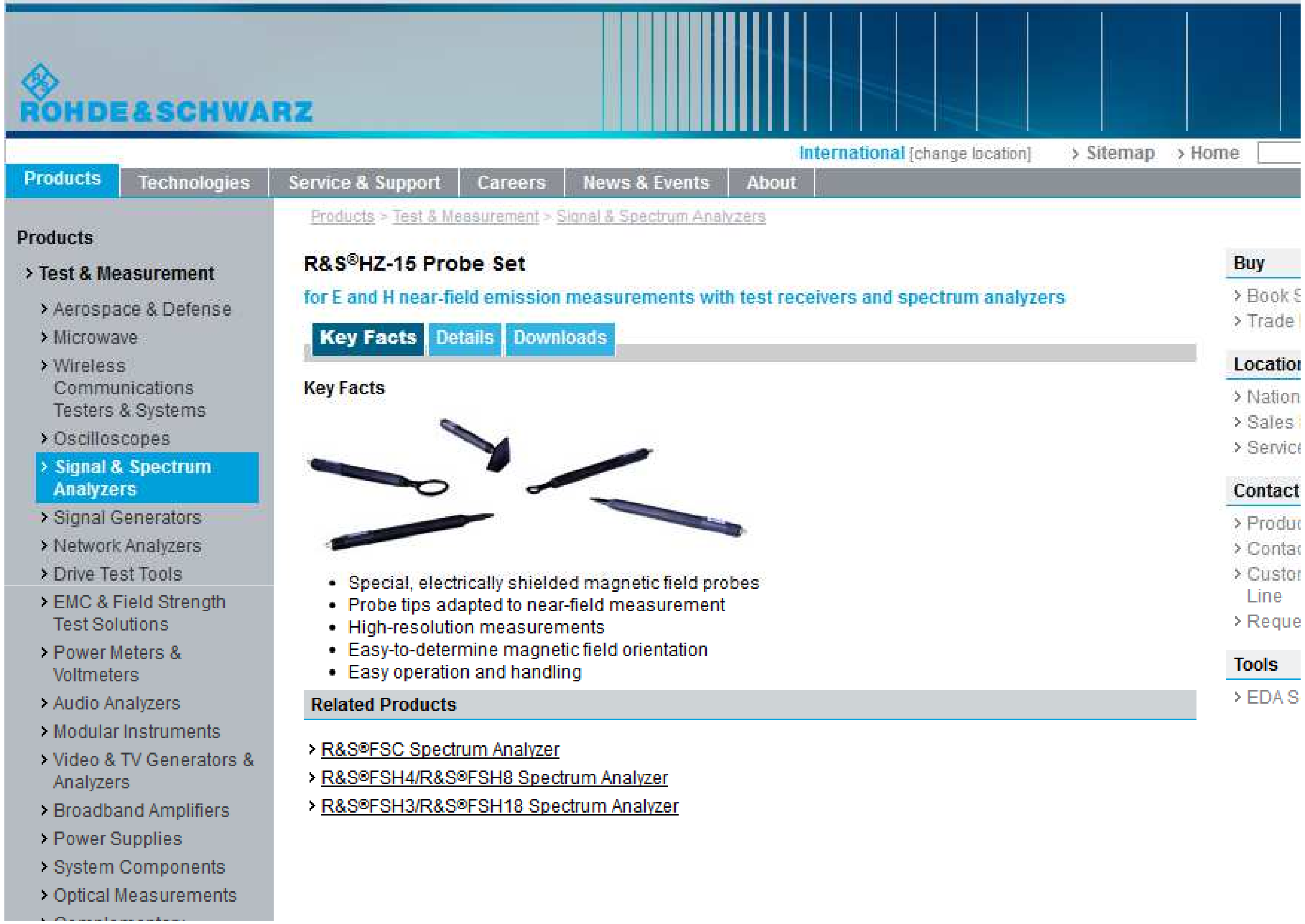
- Makes life easier… but
  - A single point may be enough (especially for hardware crypto)



```
<?xml version="1.0" encoding="utf-8"?>
<WaveformInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XM
  <Date>26/02/2012 2:42:19 PM</Date>
  <Operator>Colin O'Flynn</Operator>
  <WaveType>EMTrace</WaveType>
  <WaveFormat>System.Single[]</WaveFormat>
  <Instrument>Agilent Technologies,54831D,MY42001114,05.71.0000</Instrument>
  <Module>SASEBO-GII /w AES_Comp (Encrypt)</Module>
  <Cipher />
  <KeyLength>128</KeyLength>
  <TextWidth>128</TextWidth>
  <NumTrace>2500</NumTrace>
  <NumPoint>1</NumPoint>
  <Notes />
</WaveformInfo>
```

MAGNETIC FIELD PROBES





Products > Test & Measurement > Signal & Spectrum Analyzers

R&S®HZ-15 Probe Set

for E and H near-field emission measurements with test receivers and spectrum analyzers

Key Facts

- Special, electrically shielded magnetic field probes
- Probe tips adapted to near-field measurement
- High-resolution measurements
- Easy-to-determine magnetic field orientation
- Easy operation and handling

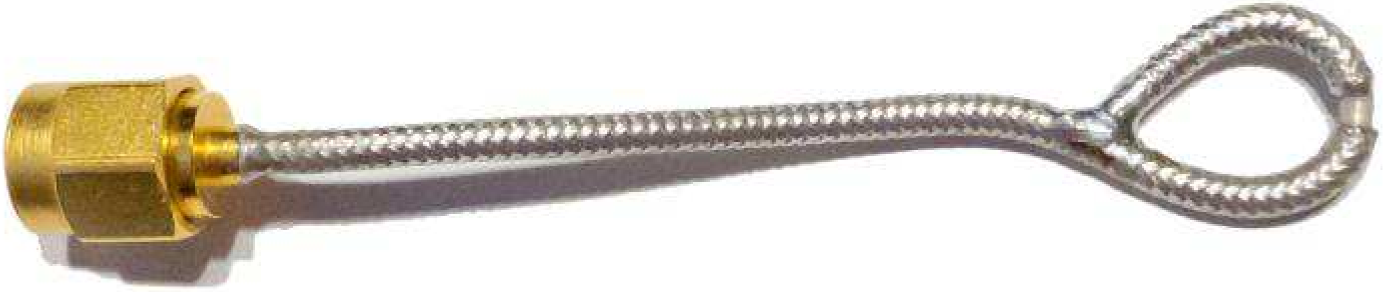
Related Products

› R&S®FSC Spectrum Analyzer

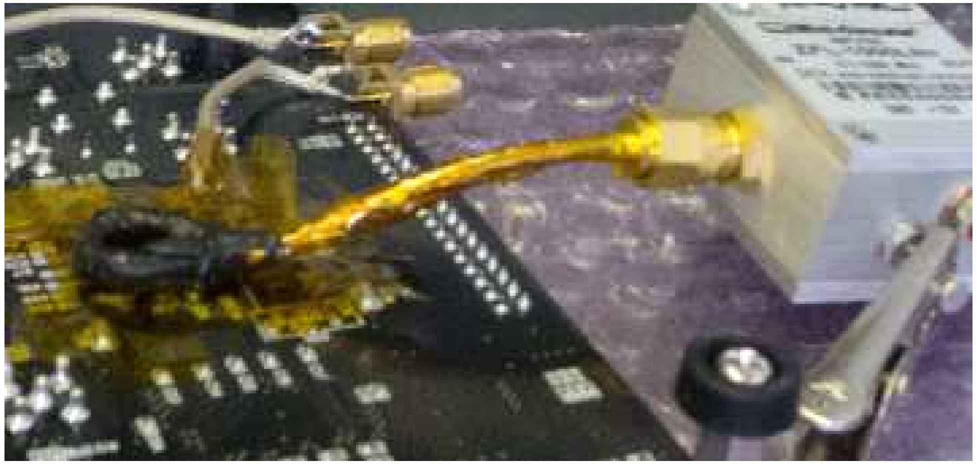
› R&S®FSH4/R&S®FSH8 Spectrum Analyzer

› R&S®FSH3/R&S®FSH18 Spectrum Analyzer

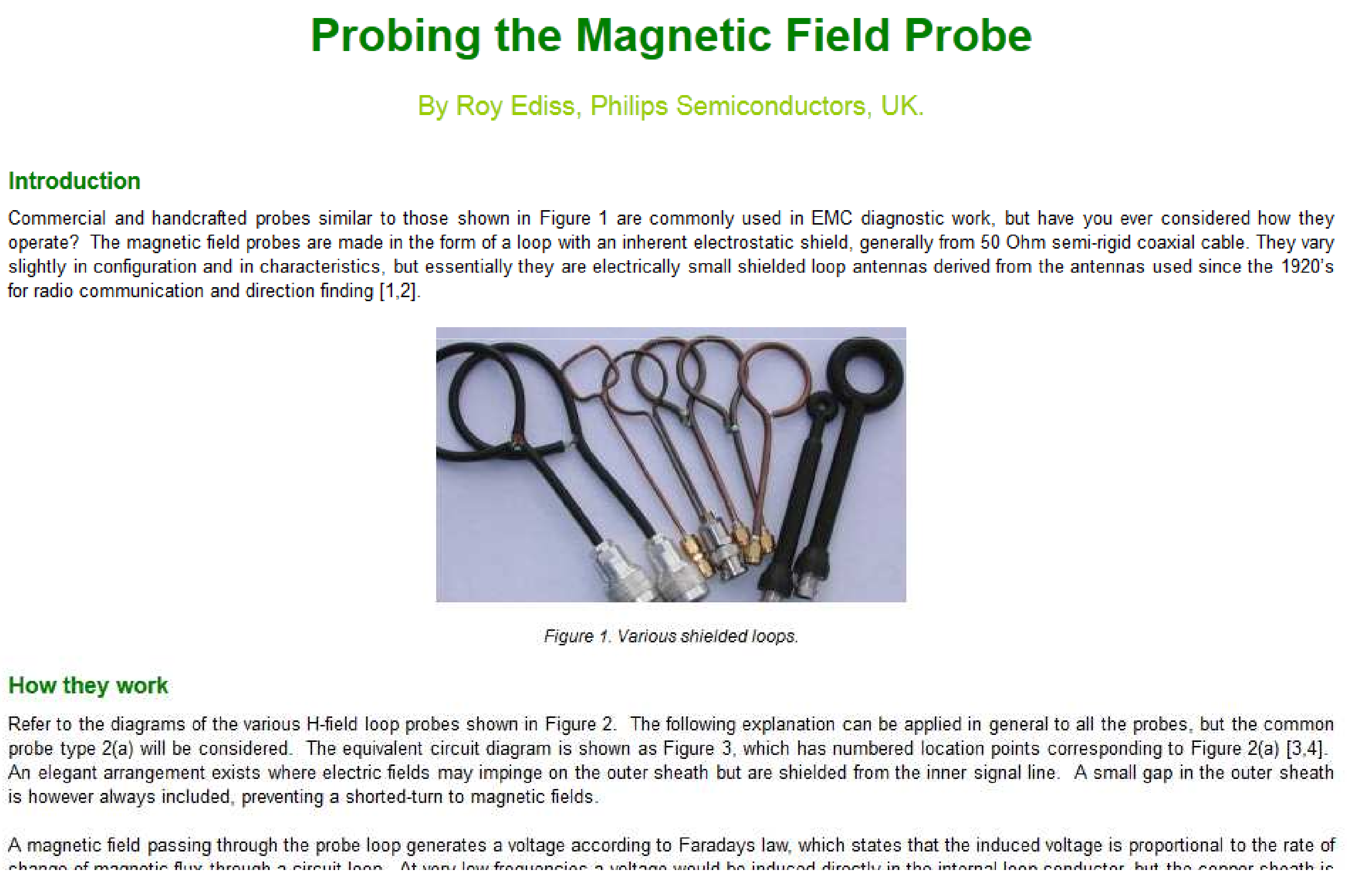
Length of Semi-Rigid cable with SMA Connectors ($3 surplus) can be turned into a simple magnetic loop:



Wrap entire thing in non-conductive tape (here I used self-fusing + polyimide) to avoid shorting out anything:



Additional References



Probing the Magnetic Field Probe

By Roy Ediss, Philips Semiconductors, UK.

Introduction

Commercial and handcrafted probes similar to those shown in Figure 1 are commonly used in EMC diagnostic work, but have you ever considered how they operate? The magnetic field probes are made in the form of a loop with an inherent electrostatic shield, generally from 50 Ohm semi-rigid coaxial cable. They vary slightly in configuration and in characteristics, but essentially they are electrically small shielded loop antennas derived from the antennas used since the 1920's for radio communication and direction finding [1,2].

Figure 1. Various shielded loops.

How they work

Refer to the diagrams of the various H-field loop probes shown in Figure 2. The following explanation can be applied in general to all the probes, but the common probe type 2(a) will be considered. The equivalent circuit diagram is shown as Figure 3, which has numbered location points corresponding to Figure 2(a) [3,4]. An elegant arrangement exists where electric fields may impinge on the outer sheath but are shielded from the inner signal line. A small gap in the outer sheath is however always included, preventing a shorted-turn to magnetic fields.

A magnetic field passing through the probe loop generates a voltage according to Faradays law, which states that the induced voltage is proportional to the rate of change of magnetic flux through a circuit loop. At very low frequencies a voltage would be induced directly in the internal loop conductor, but the copper sheath is

http://www.compliance-club.com/archive/old_archive/030718.htm

Additional References

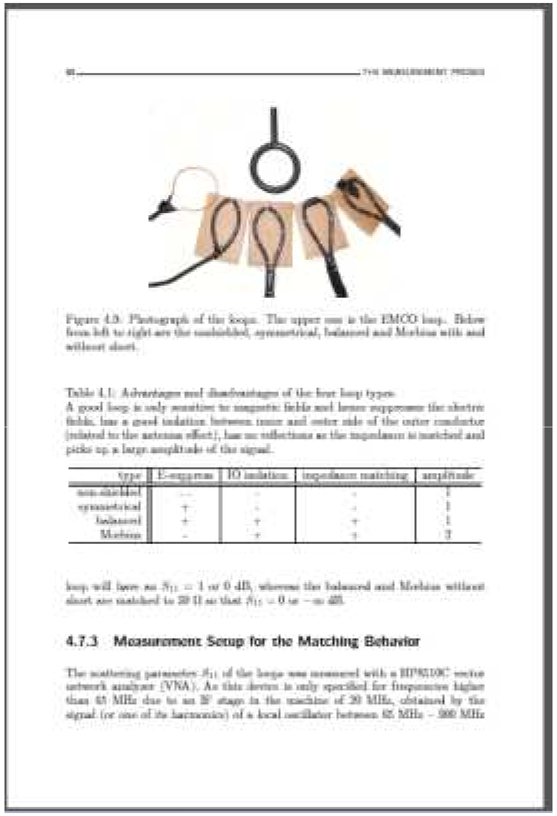


Figure 4.9: Photograph of the loops. The upper one is the EMCO loop. Below from left to right are the unshielded, symmetrical, balanced and Moebius with and without short.

Table 4.1: Advantages and disadvantages of the four loop types. A good loop is only sensitive to magnetic fields and hence suppresses the electric fields, has a good isolation between inner and outer side of the outer conductor (related to the antenna effect), has no reflections as the impedance is matched and picks up a large amplitude of the signal.

type	E-suppress	IO isolation	impedance matching	amplitude
non-shielded	--	-	-	1
symmetrical	+	-	-	1
balanced	+	+	+	1
Moebius	-	+	+	2

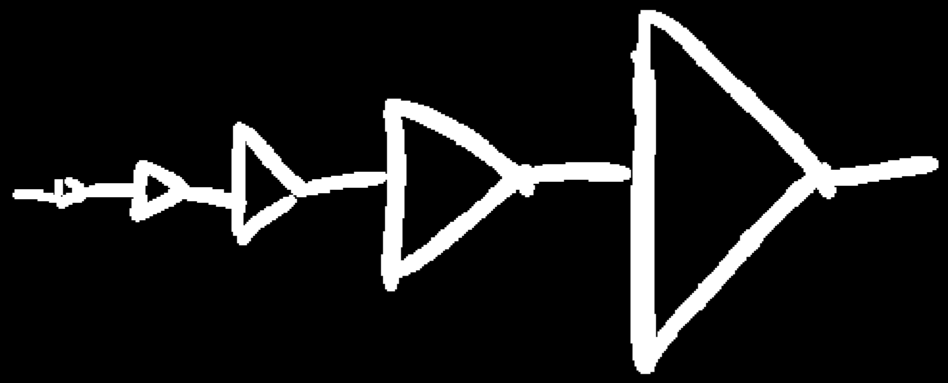
loop will have an $S_{11} = 1$ or 0 dB, whereas the balanced and Moebius without short are matched to 50 Ω so that $S_{11} = 0$ or $-\infty$ dB.

4.7.3 Measurement Setup for the Matching Behavior

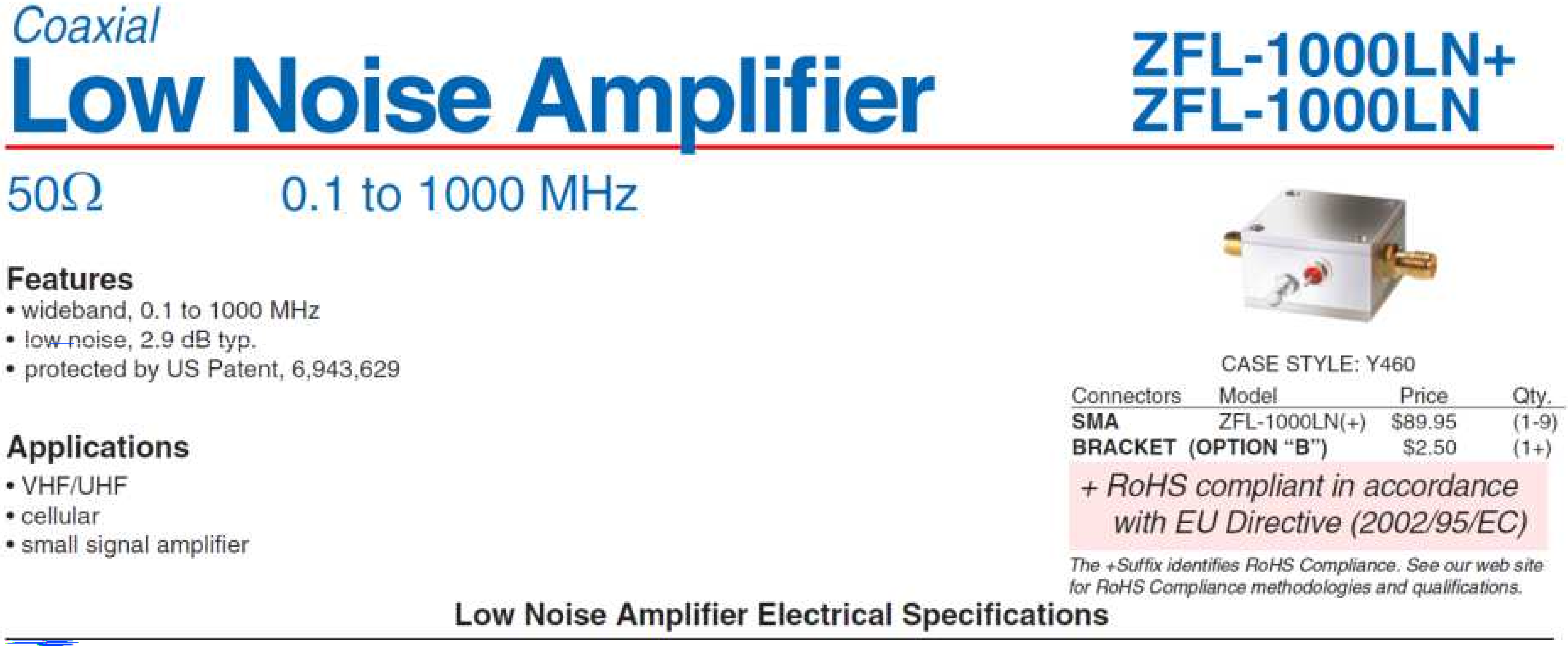
The scattering parameter S_{11} of the loops was measured with a HP8510C vector network analyzer (VNA). As this device is only specified for frequencies higher than 45 MHz due to an IF stage in the machine of 20 MHz, obtained by the signal (or one of its harmonics) of a local oscillator between 65 MHz – 300 MHz

Elke De Mulder: Electromagnetic Techniques and Probes for Side-Channel Analysis on Cryptographic Devices
http://www.cosic.esat.kuleuven.be/publications/thesis-182.pdf

PRE-AMPLIFIER



Low Noise Amplifier



Coaxial

Low Noise Amplifier

ZFL-1000LN+
ZFL-1000LN

50Ω 0.1 to 1000 MHz

Features

- wideband, 0.1 to 1000 MHz
- low noise, 2.9 dB typ.
- protected by US Patent, 6,943,629

Applications

- VHF/UHF
- cellular
- small signal amplifier

CASE STYLE: Y460

Connectors	Model	Price	Qty.
SMA	ZFL-1000LN(+)	$89.95	(1-9)
BRACKET	**(OPTION "B")**	$2.50	(1+)

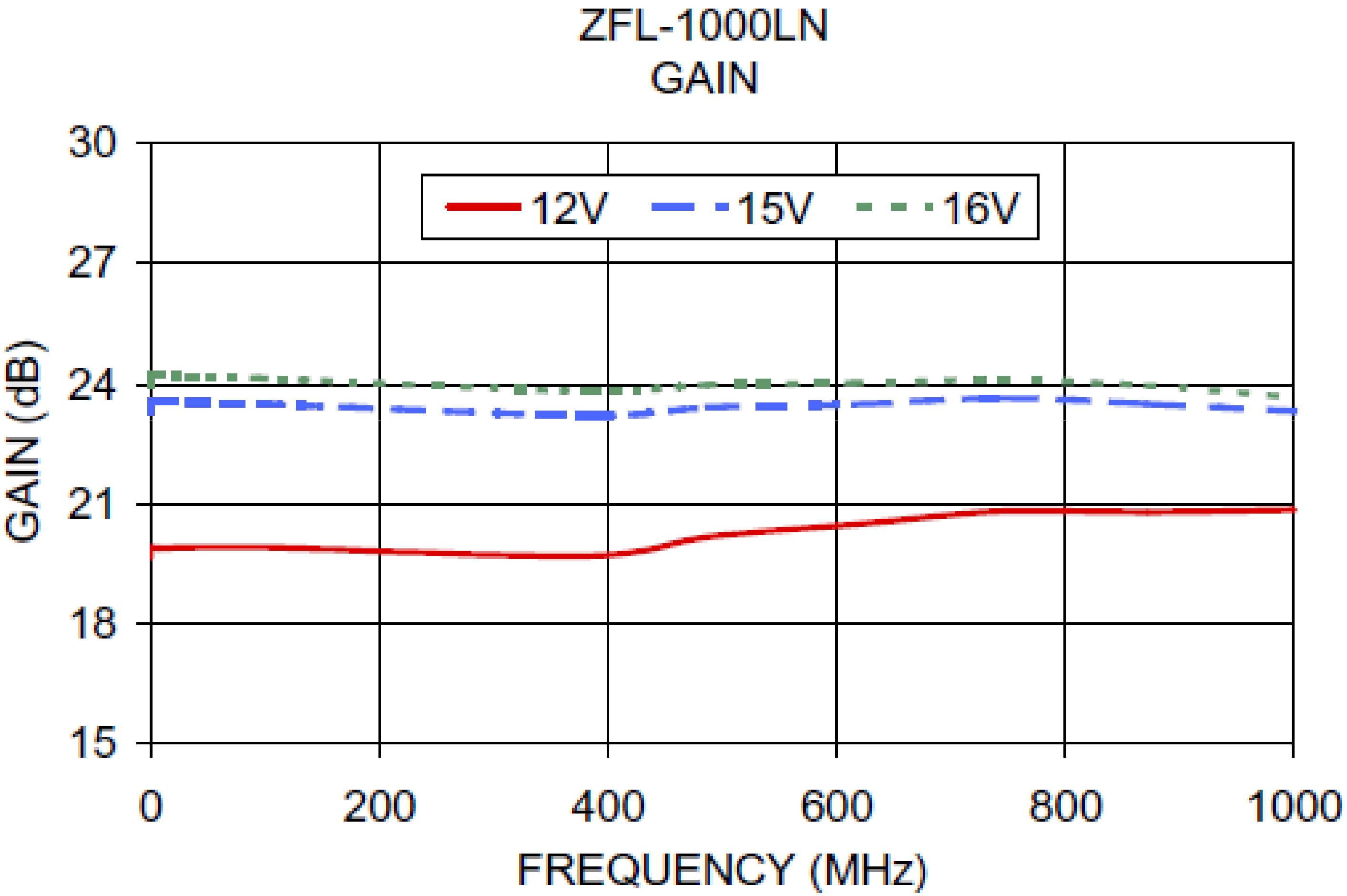
+ RoHS compliant in accordance with EU Directive (2002/95/EC)

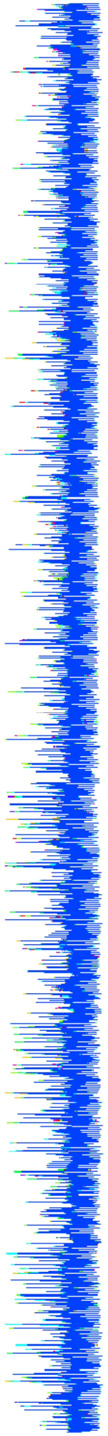
The +Suffix identifies RoHS Compliance. See our web site for RoHS Compliance methodologies and qualifications.

Low Noise Amplifier Electrical Specifications

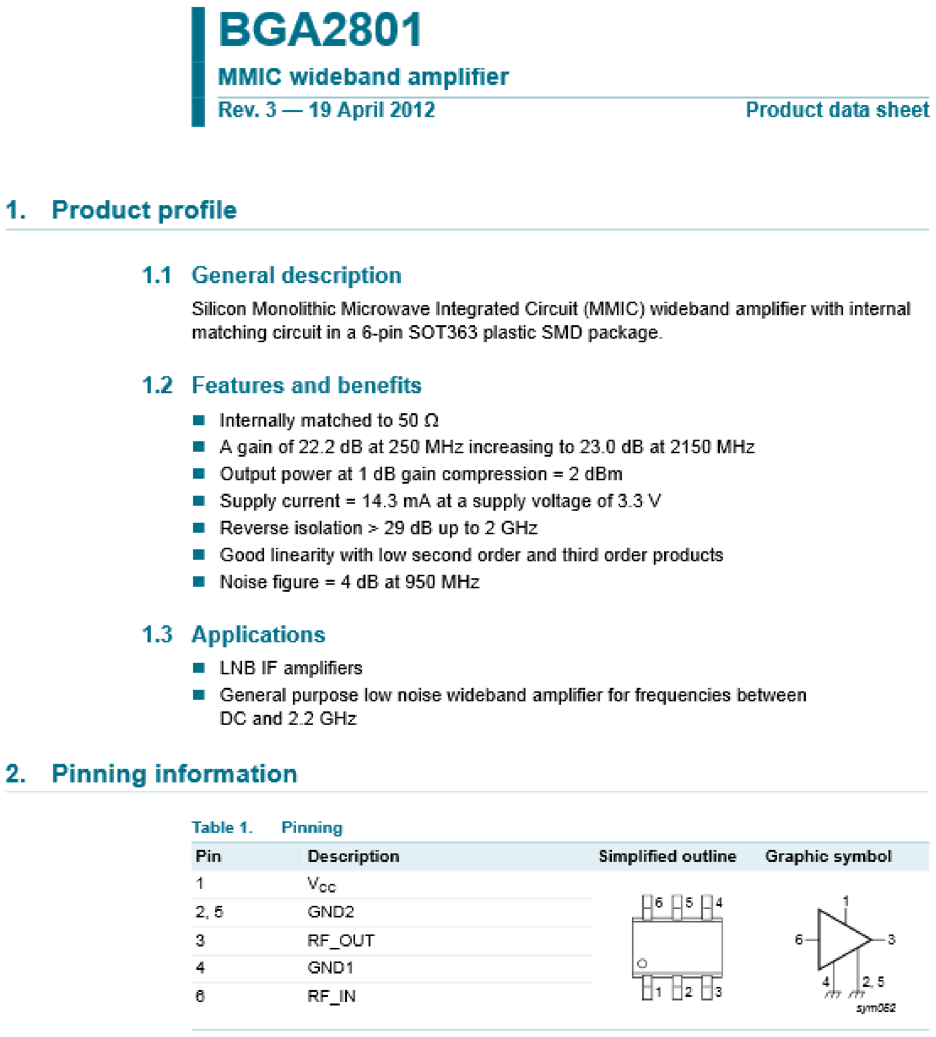
Assuming we are making a probe, there is no need to purchase the expensive pre-amplifier offered by that manufacture. Here is a 20 dB amplifier for $90, it was shown being used in another photo.

Low Noise Amplifier





Even Cheaper...



BGA2801

MMIC wideband amplifier

Rev. 3 — 19 April 2012 Product data sheet

1. Product profile

1.1 General description

Silicon Monolithic Microwave Integrated Circuit (MMIC) wideband amplifier with internal matching circuit in a 6-pin SOT363 plastic SMD package.

1.2 Features and benefits

- Internally matched to 50 Ω
- A gain of 22.2 dB at 250 MHz increasing to 23.0 dB at 2150 MHz
- Output power at 1 dB gain compression = 2 dBm
- Supply current = 14.3 mA at a supply voltage of 3.3 V
- Reverse isolation > 29 dB up to 2 GHz
- Good linearity with low second order and third order products
- Noise figure = 4 dB at 950 MHz

1.3 Applications

- LNB IF amplifiers
- General purpose low noise wideband amplifier for frequencies between DC and 2.2 GHz

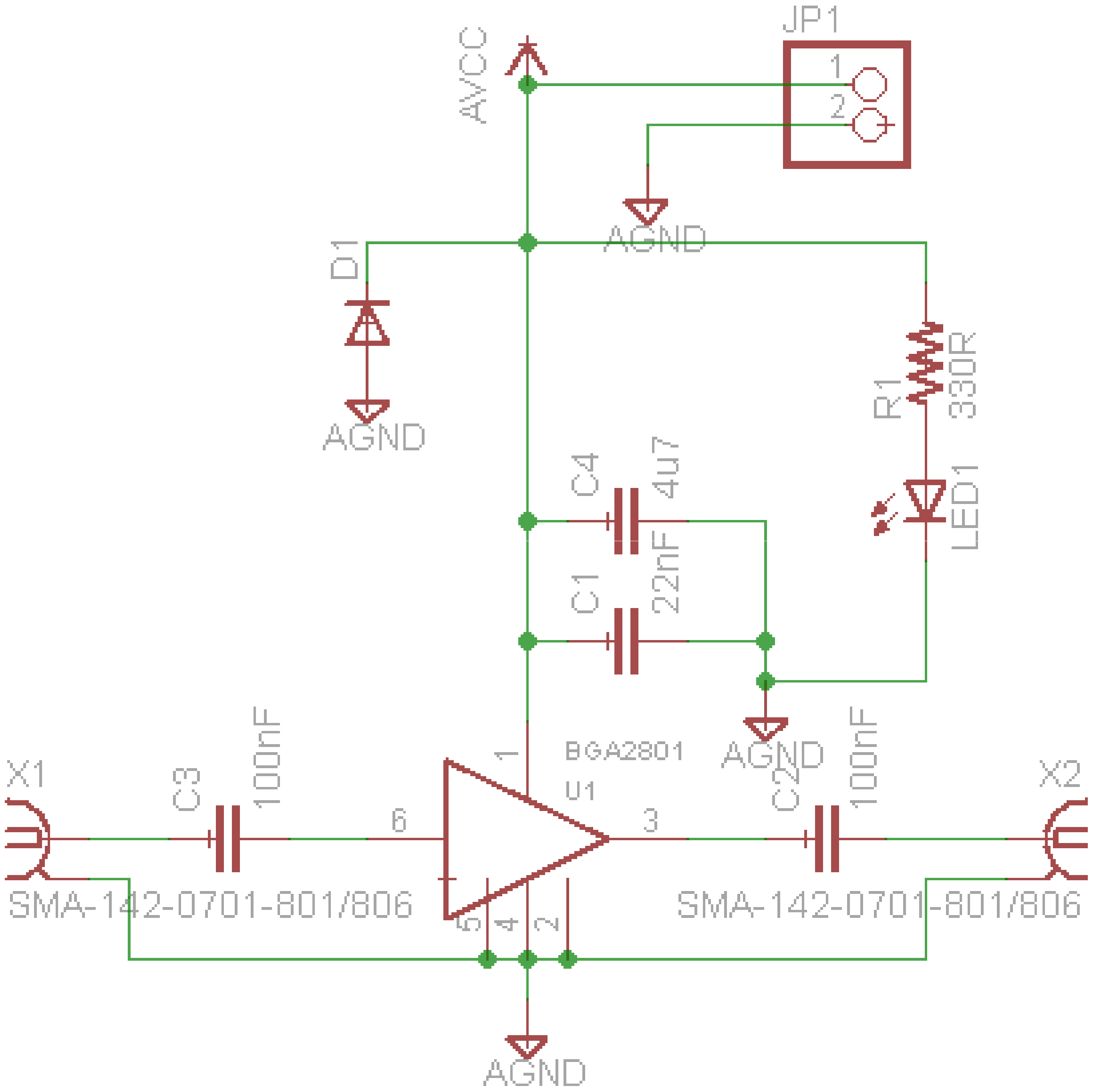
2. Pinning information

Table 1. Pinning

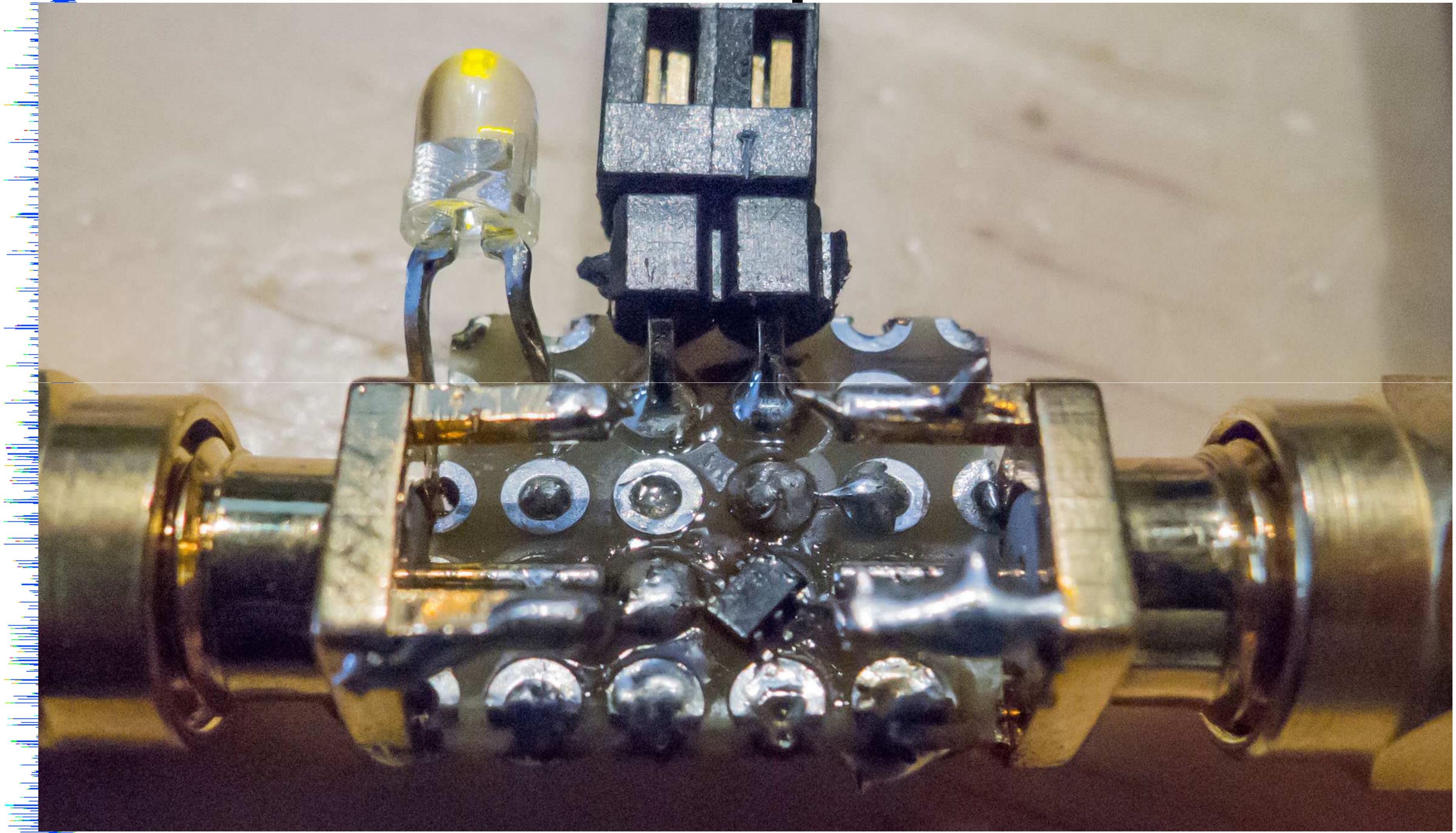
Pin	Description	Simplified outline	Graphic symbol
1	V_{CC}		
2, 5	GND2		
3	RF_OUT		
4	GND1		
6	RF_IN		

~

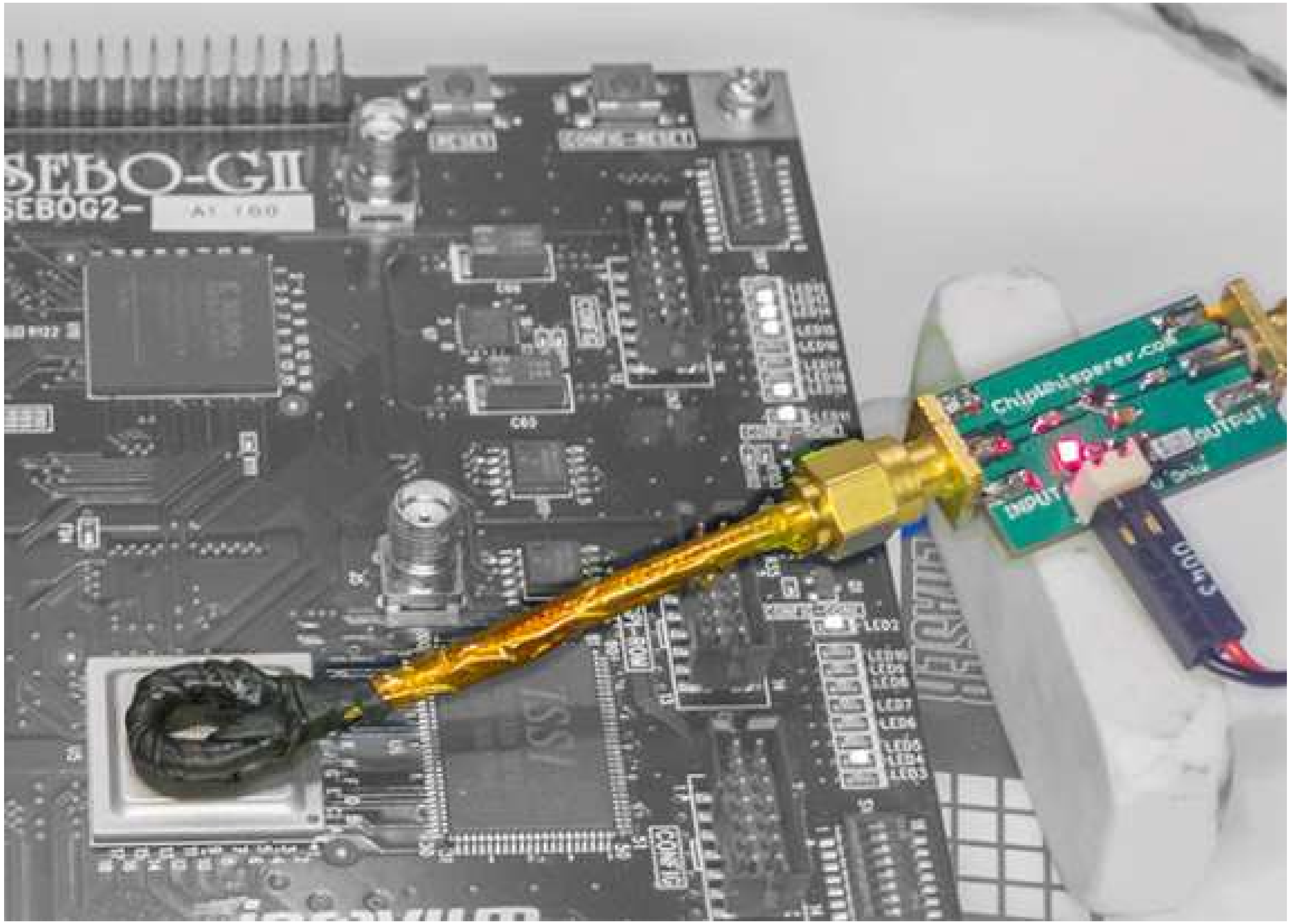
$0.60

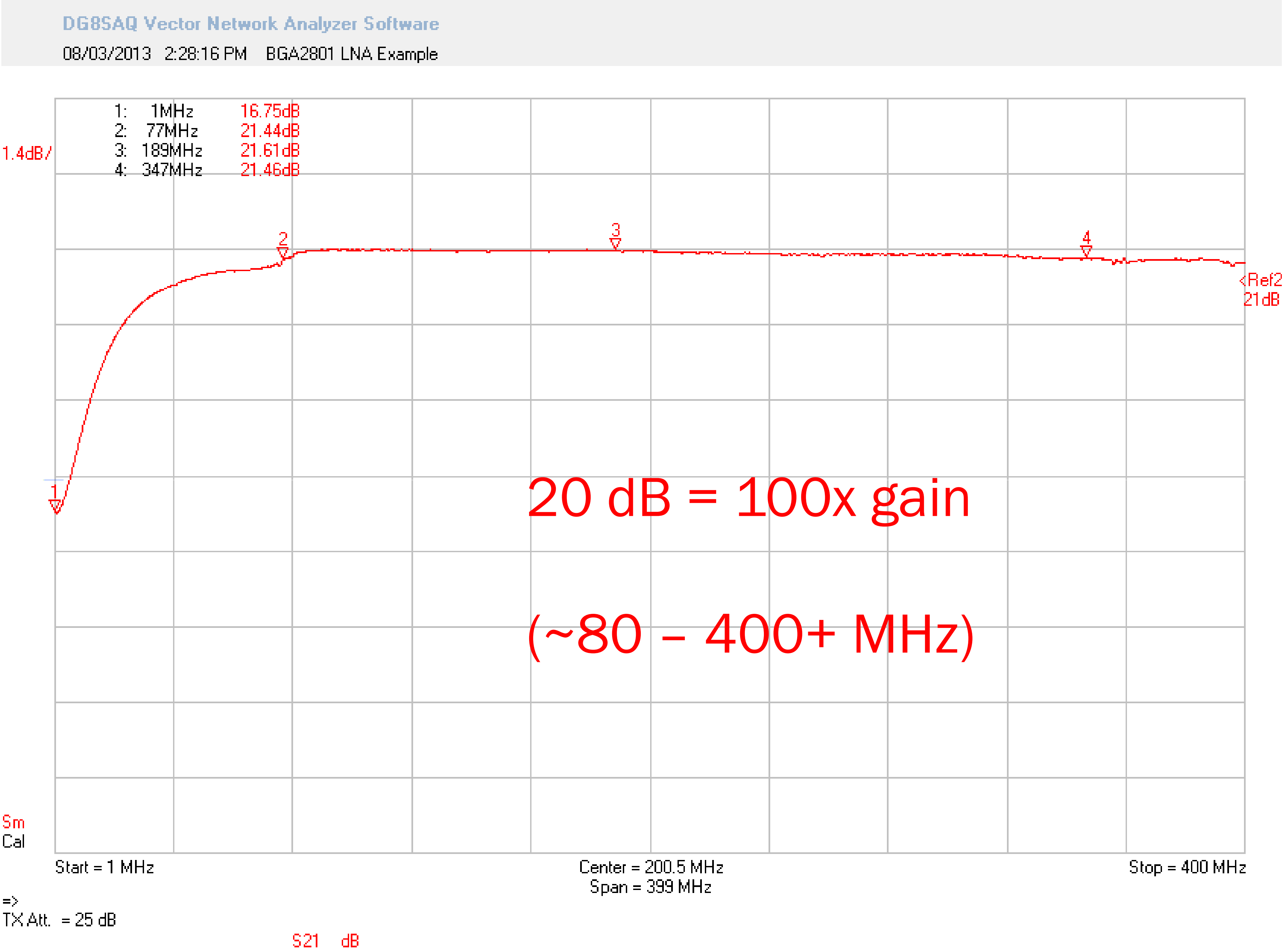

JP1
1
2
AVCC
AGND
D1
AGND
R1
330R
LED1
C4
4u7
C1
22nF
AGND
BGA2801
U1
1
6
3
5
4
2
X1
C3
100nF
SMA-142-0701-801/806
C2
100nF
X2
SMA-142-0701-801/806
AGND

Pre-Amplifier



Pre-Amplifier





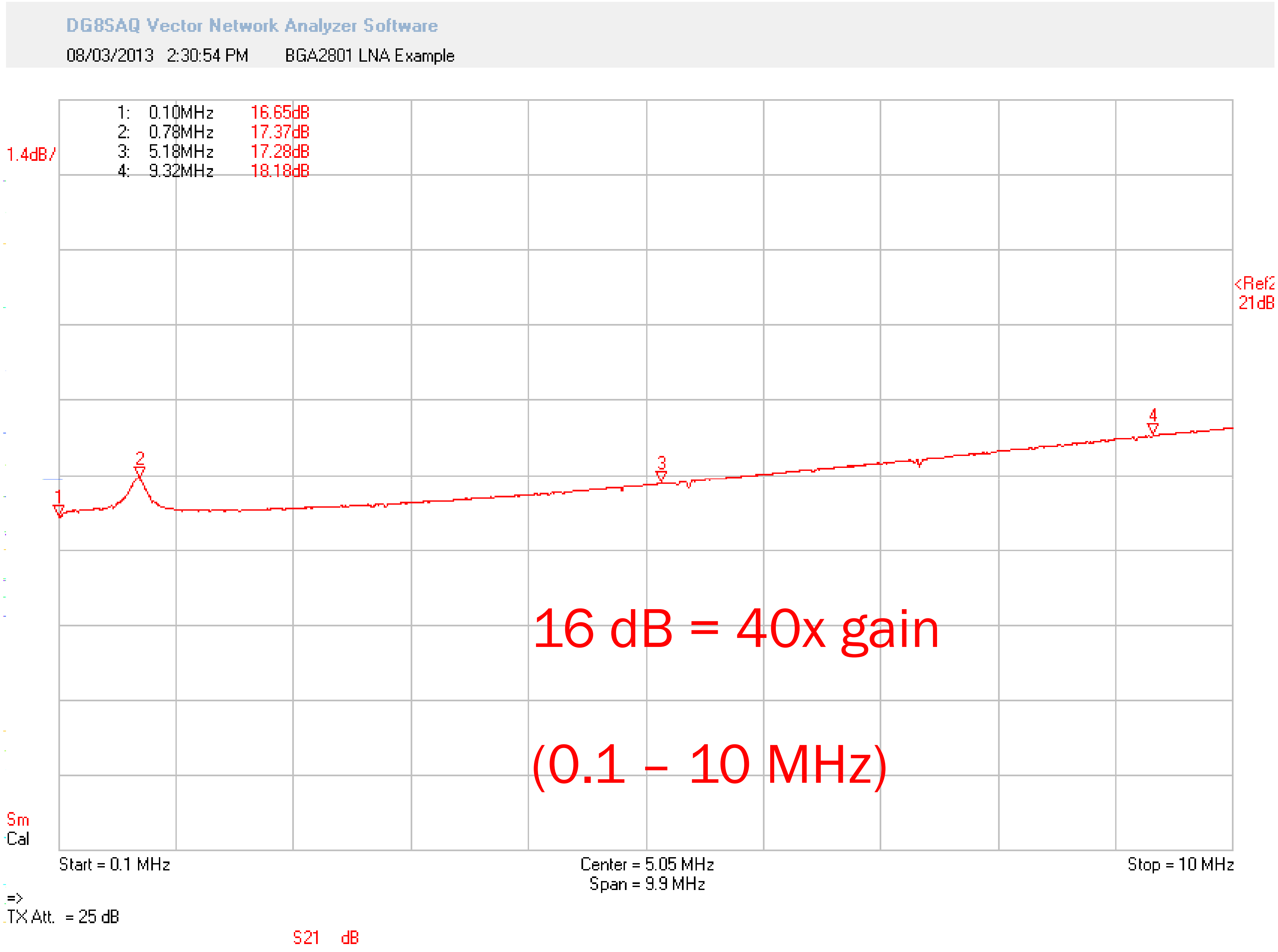
DG8SAQ Vector Network Analyzer Software

08/03/2013 2:28:16 PM BGA2801 LNA Example

1: 1MHz 16.75dB
2: 77MHz 21.44dB
3: 189MHz 21.61dB
4: 347MHz 21.46dB

20 dB = 100x gain

(~80 – 400+ MHz)



DG8SAQ Vector Network Analyzer Software

08/03/2013 2:30:54 PM BGA2801 LNA Example

1:	0.10MHz	16.65dB
2:	0.78MHz	17.37dB
3:	5.18MHz	17.28dB
4:	9.32MHz	18.18dB

Start = 0.1 MHz

Center = 5.05 MHz
Span = 9.9 MHz

Stop = 10 MHz

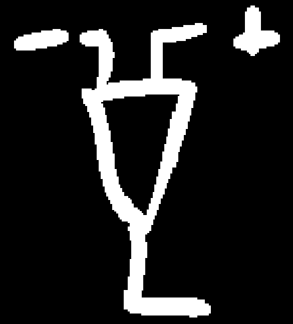
=>
TX Att. = 25 dB

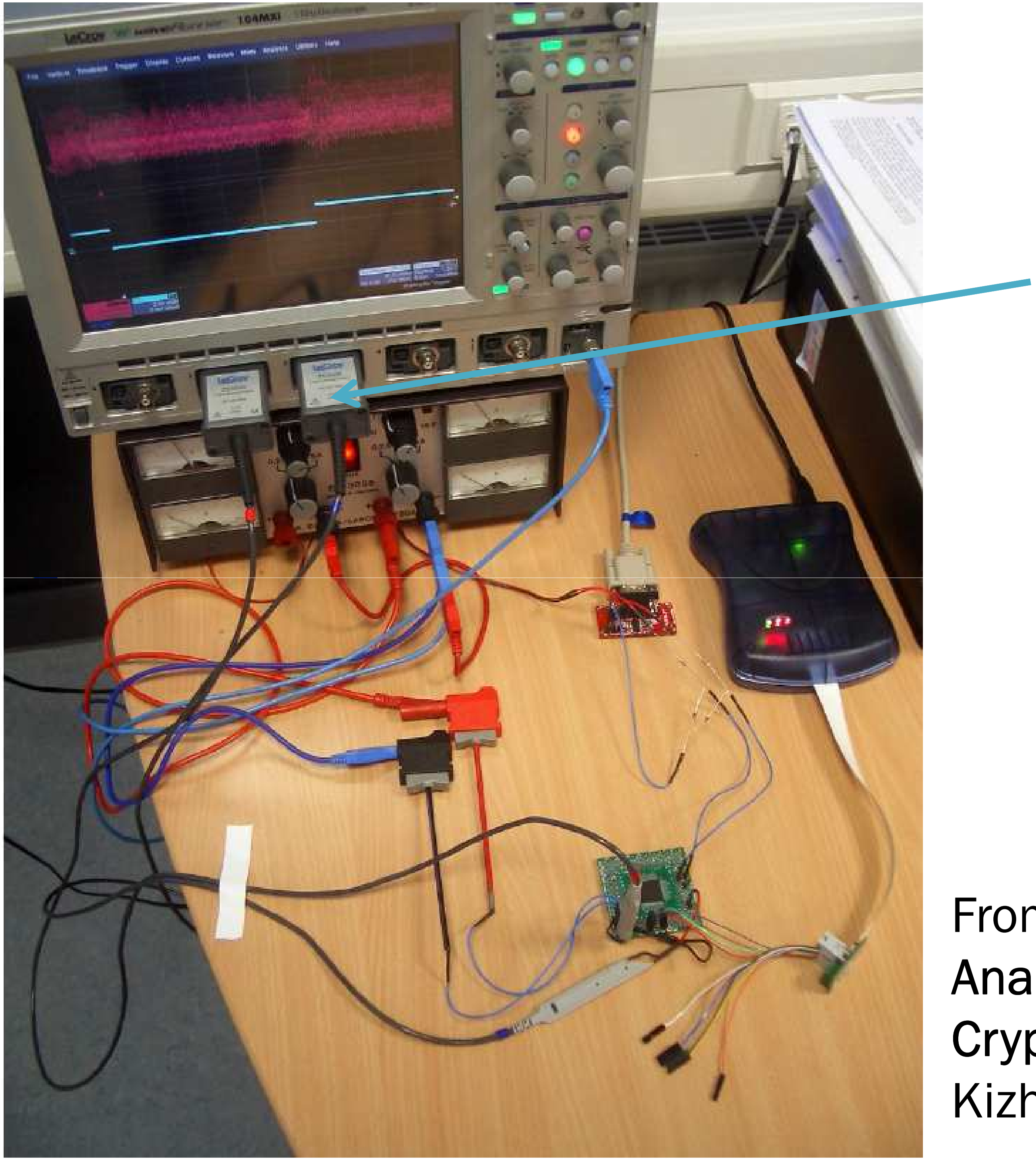
S21 dB

16 dB = 40x gain

(0.1 – 10 MHz)

DIFFERENTIAL PROBE





Differential Probe

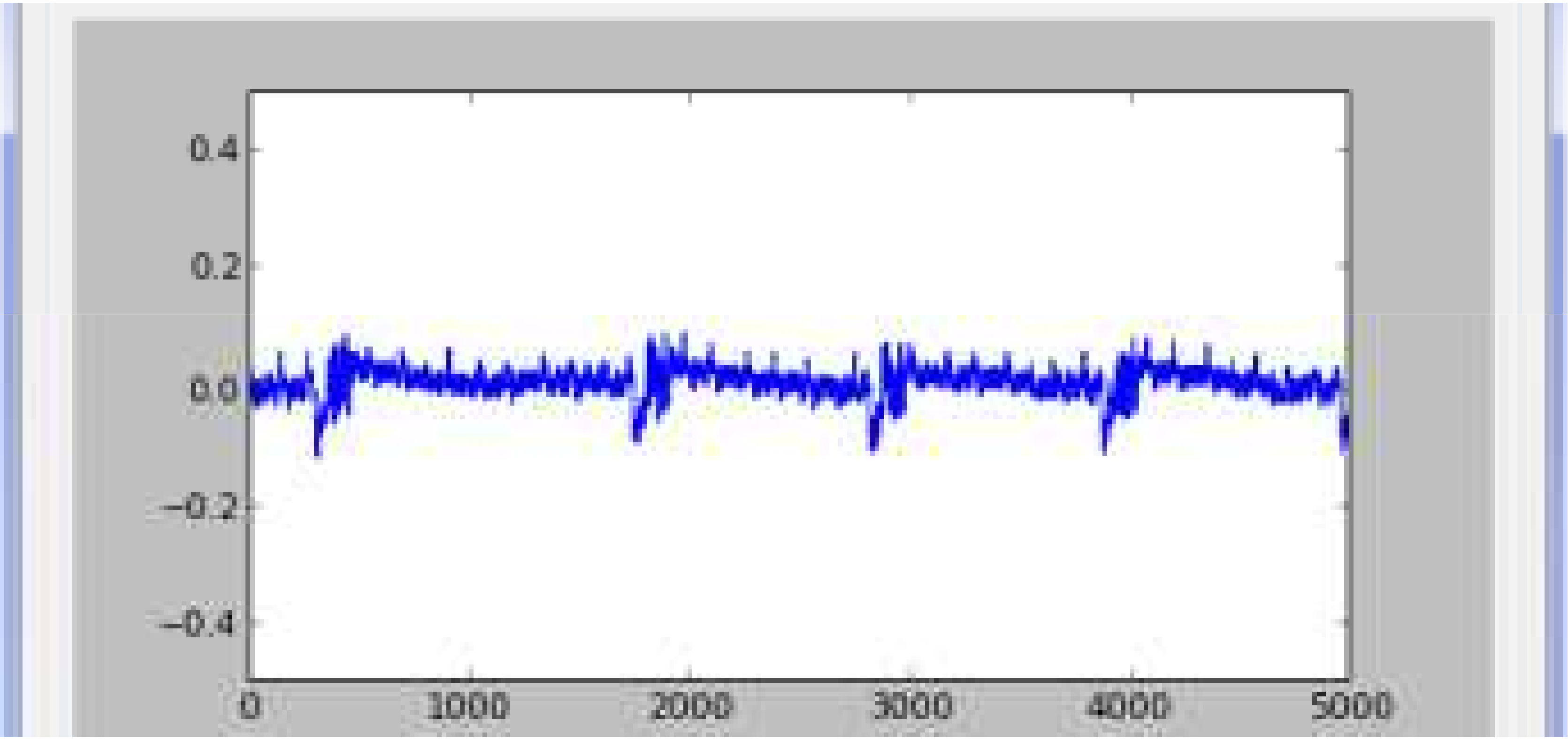
From "**Side Channel Analysis of AVR XMEGA Crypto Engine**" by Ilya Kizhvatov

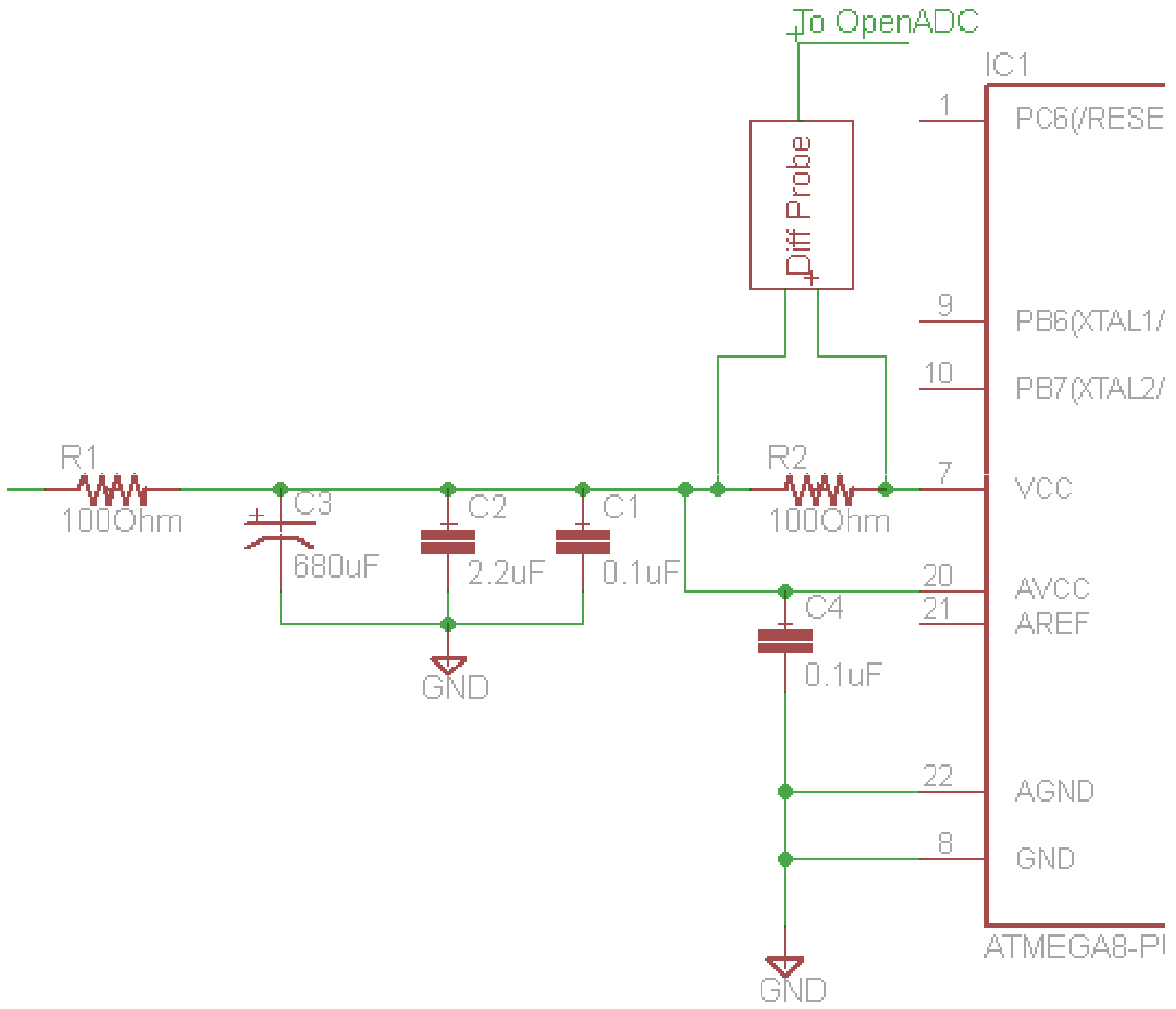
V = I R

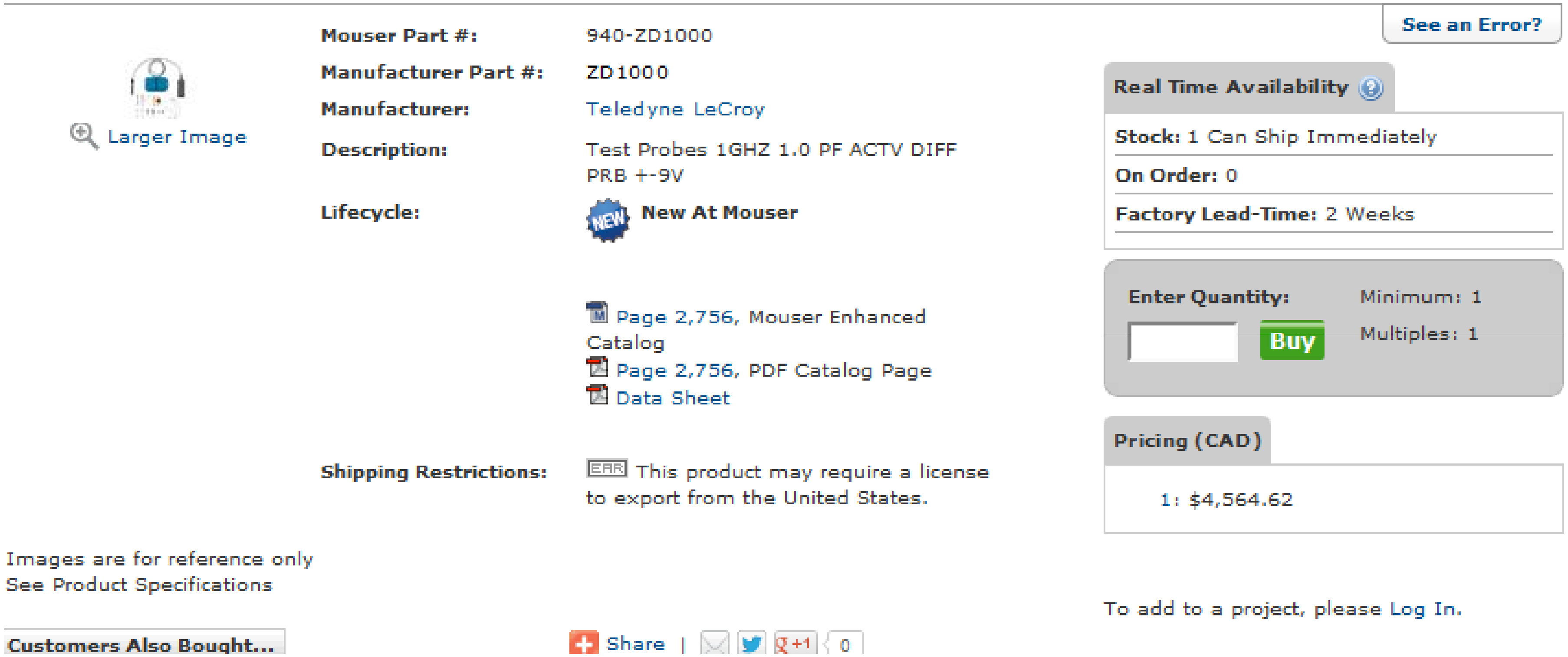
i.e. say signature was 0.2 mA, shunt was 75 ohms

0.0002 x 75 = 0.015 = 15 mV

Common-Mode Noise




To OpenADC
Diff Probe
+
IC1
1
PC6(/RESE
9
PB6(XTAL1/
10
PB7(XTAL2/
7
VCC
20
AVCC
21
AREF
22
AGND
8
GND
ATMEGA8-PI
R1
100Ohm
C3
680uF
+
C2
2.2uF
C1
0.1uF
R2
100Ohm
C4
0.1uF
GND
GND



See an Error?

Mouser Part #:	940-ZD1000
Manufacturer Part #:	ZD1000
Manufacturer:	Teledyne LeCroy
Description:	Test Probes 1GHZ 1.0 PF ACTV DIFF PRB +-9V
Lifecycle:	**New At Mouser**
	Page 2,756, Mouser Enhanced Catalog Page 2,756, PDF Catalog Page Data Sheet
Shipping Restrictions:	EAR This product may require a license to export from the United States.

Images are for reference only
See Product Specifications

Real Time Availability

Stock: 1 Can Ship Immediately

On Order: 0

Factory Lead-Time: 2 Weeks

Enter Quantity: Minimum: 1
Multiples: 1

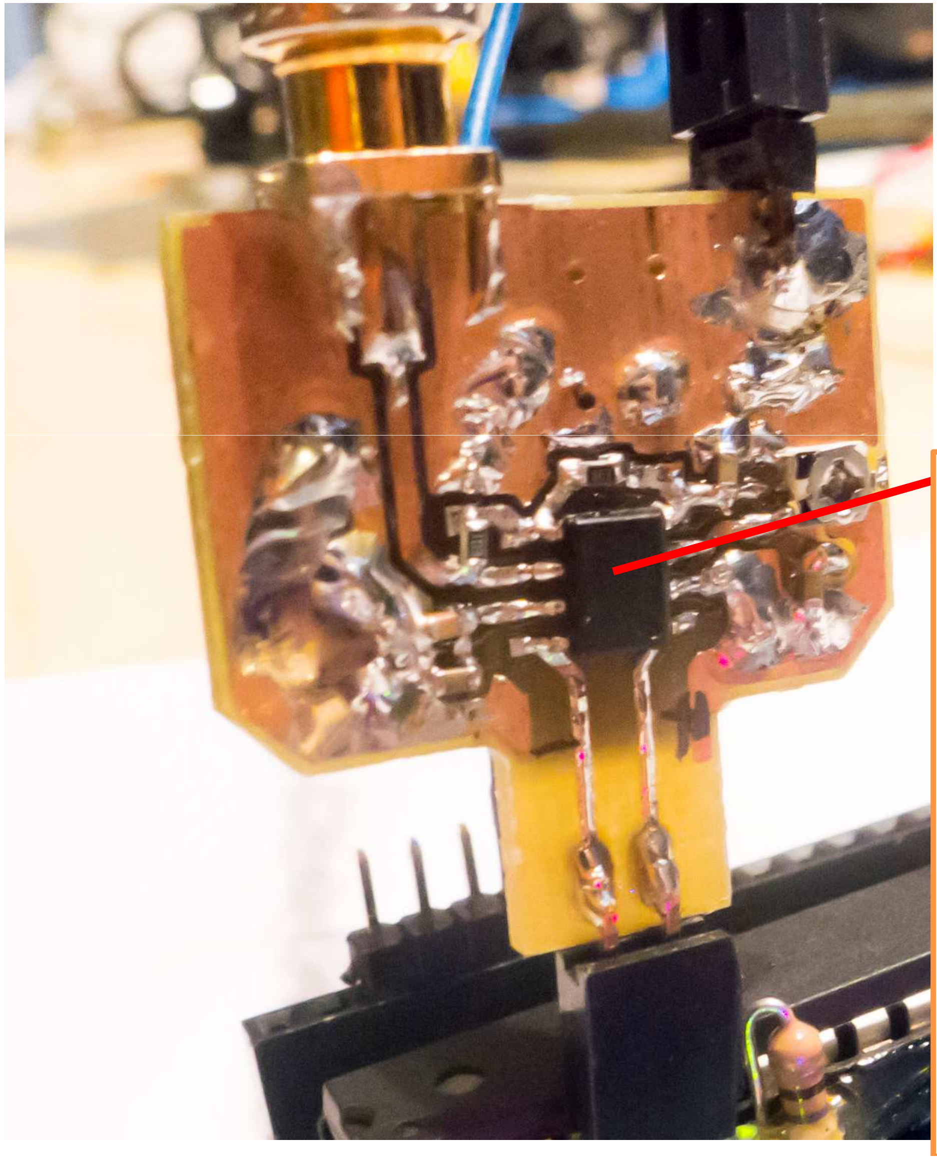
Pricing (CAD)

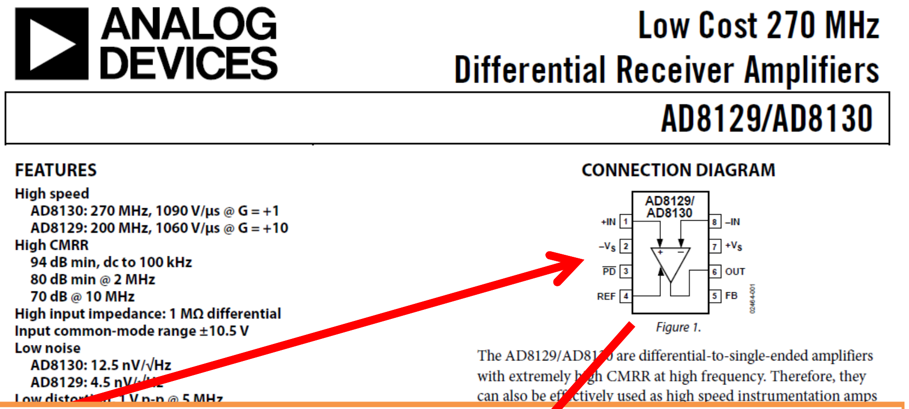
1: $4,564.62

To add to a project, please Log In.

Customers Also Bought...

Be A Cheapskate!





ANALOG DEVICES

Low Cost 270 MHz Differential Receiver Amplifiers

AD8129/AD8130

FEATURES

High speed
- AD8130: 270 MHz, 1090 V/μs @ G = +1
- AD8129: 200 MHz, 1060 V/μs @ G = +10

High CMRR
- 94 dB min, dc to 100 kHz
- 80 dB min @ 2 MHz
- 70 dB @ 10 MHz

High input impedance: 1 MΩ differential

Input common-mode range ±10.5 V

Low noise
- AD8130: 12.5 nV/√Hz
- AD8129: 4.5 nV/√Hz

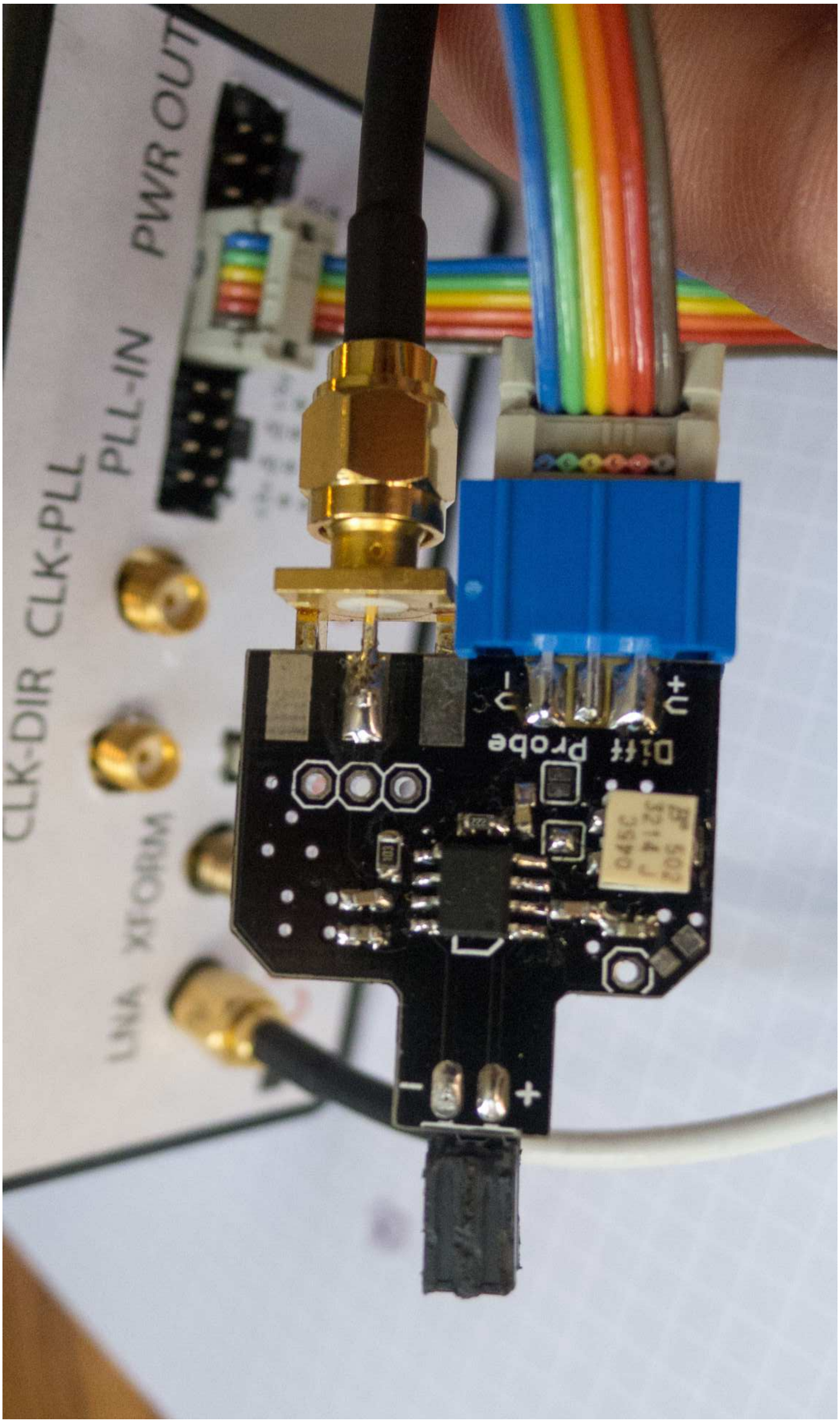
Low distortion: 1 V p-p @ 5 MHz

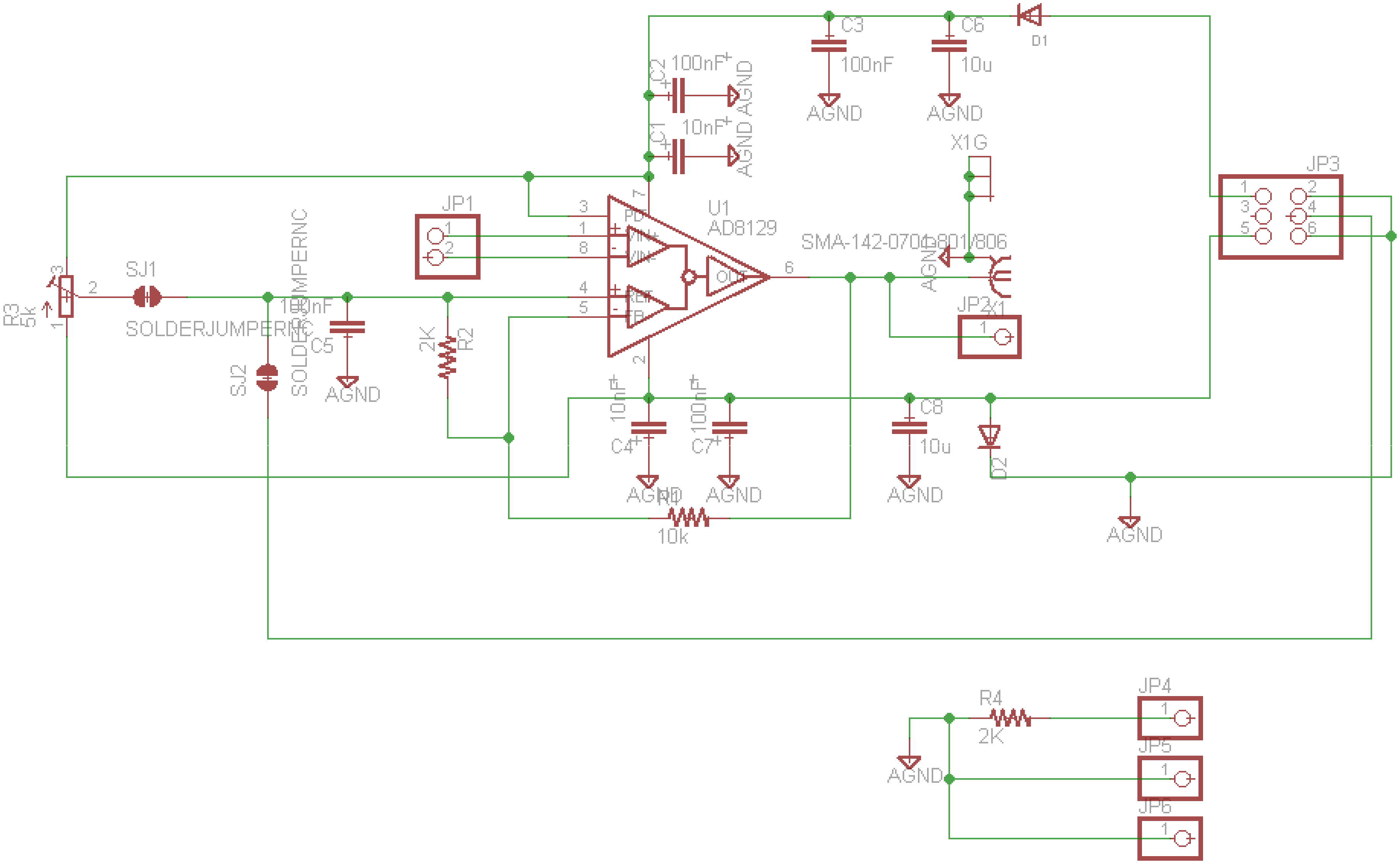
CONNECTION DIAGRAM

Figure 1.

The AD8129/AD8130 are differential-to-single-ended amplifiers with extremely high CMRR at high frequency. Therefore, they can also be effectively used as high speed instrumentation amps

This chip is < $5 in single-unit quantities! Add a voltage supply & a few resistors/capacitors and you've got a pretty good probe.


PWR OUT
PLL-IN
CLK-PLL
CLK-DIR
XFORM
LNA
Diff Probe
-U
+U
-
+



C3
100nF
AGND
C6
10u
AGND
D1
C2
100nF
C1
10nF
AGND AGND
X1G
JP3
1
2
3
4
5
6
U1
AD8129
PD
+VIN
-VIN
REF
FB
OUT
3
1
8
4
5
7
2
6
JP1
1
2
SMA-142-0701-801/806
AGND
JP2
X1
1
R3
5k
1
2
3
SJ1
SOLDERJUMPERNC
100nF
C5
AGND
SJ2
SOLDERJUMPERNC
2K
R2
10nF
C4
100nF
C7
AGND
AGND
R1
10k
C8
10u
AGND
D2
AGND
R4
2K
JP4
JP5
JP6
AGND

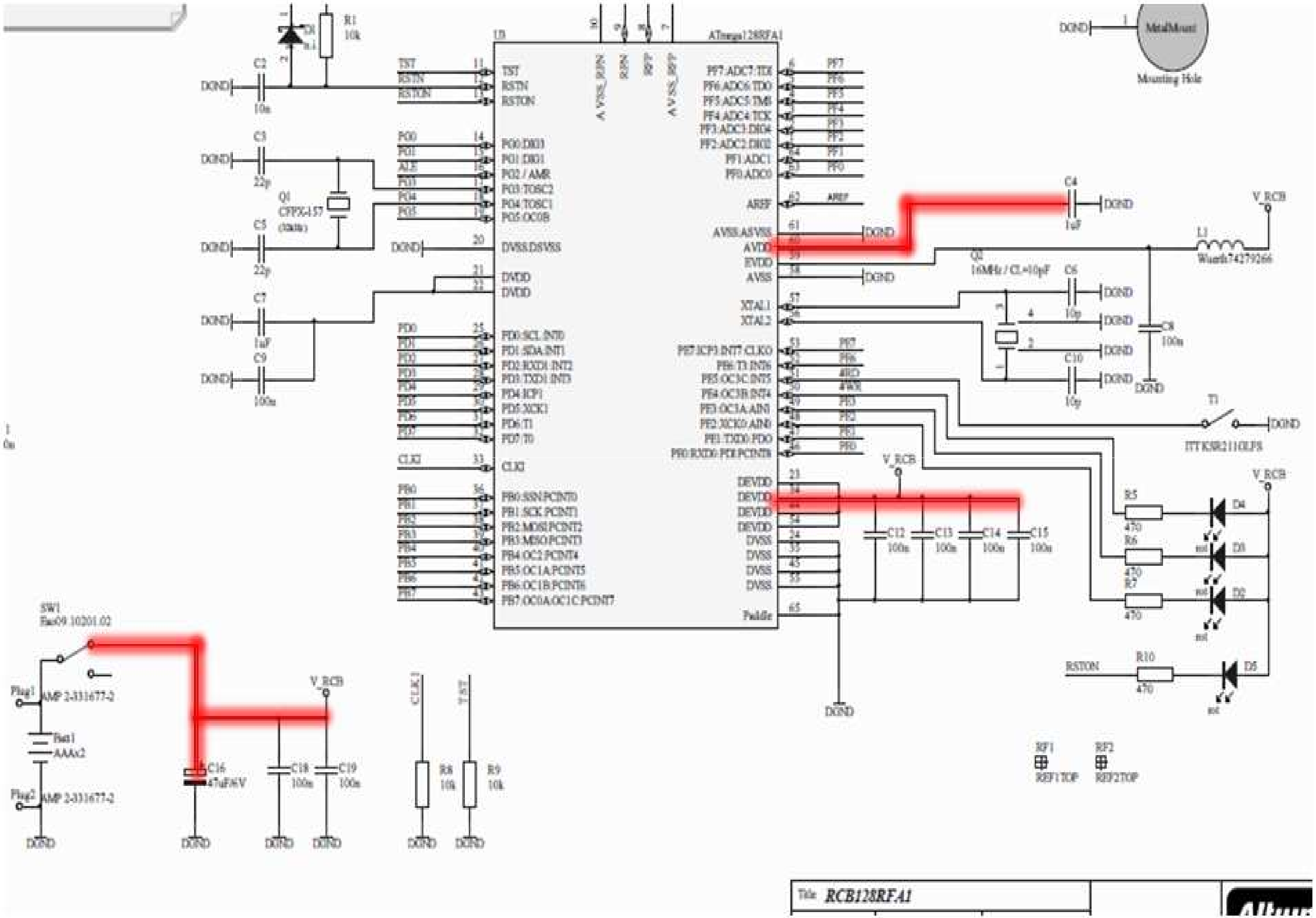
Brief Notes

- The AD8129 must have supply voltages with at least about 1.3V of headroom.
- –VS can be connected to ground (using jumper), and you only need to supply a single positive voltage.
  - COULD NOT use this on a shunt in the ground path, since the lowest common mode voltage it could measure is around 1.3V.
  - ChipWhisperer provides +/-8V Rails

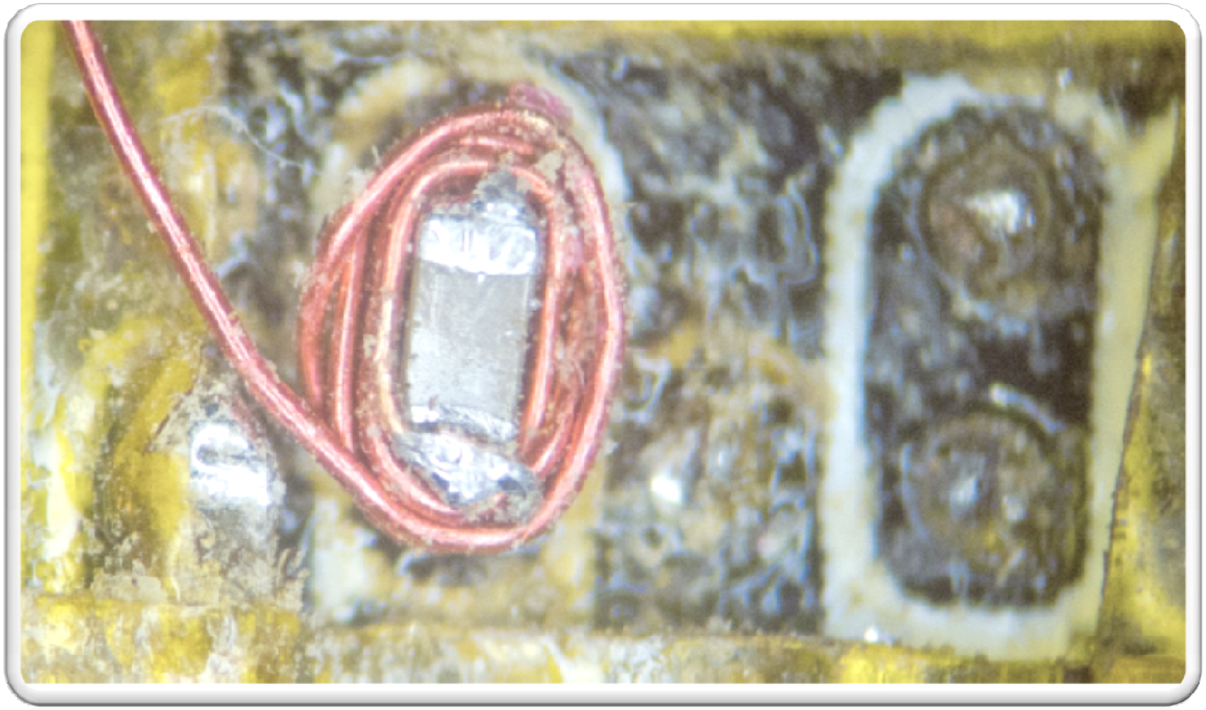
Brief Notes

- 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 VCC-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.
  - If +VS is 5V and –VS is 0V (GND), this means you want the output to be around 2.5V
  - If +VS is 7V and –VS is 0V (GND), this means you want the output to be around 3.5V
  - If +VS is 5V and –VS is -2V, this means you want the output to be around 1.5V

DECOUPLING CAPACITOR MEASUREMENT

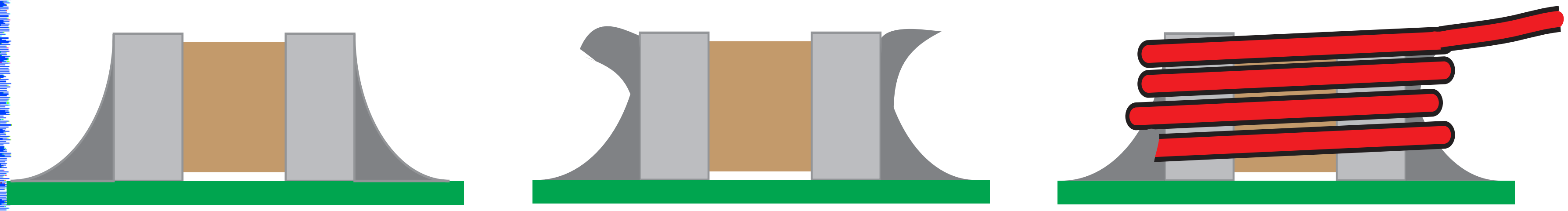

U3
ATmega128RFA1
TST
RSTN
RSTON
PG0:DIG3
PG1:DIG1
PG2 / AMR
PG3:TOSC2
PG4:TOSC1
PG5:OC0B
DVSS:DSVSS
DVDD
DVDD
PD0:SCL:INT0
PD1:SDA:INT1
PD2:RXD1:INT2
PD3:TXD1:INT3
PD4:ICP1
PD5:XCK1
PD6:T1
PD7:T0
CLKI
PB0:SSN:PCINT0
PB1:SCK:PCINT1
PB2:MOSI:PCINT2
PB3:MISO:PCINT3
PB4:OC2:PCINT4
PB5:OC1A:PCINT5
PB6:OC1B:PCINT6
PB7:OC0A:OC1C:PCINT7
AVSS_RFN
RFN
RFP
AVSS_RFP
PF7:ADC7:TDI
PF6:ADC6:TDO
PF5:ADC5:TMS
PF4:ADC4:TCK
PF3:ADC3:DIG4
PF2:ADC2:DIG2
PF1:ADC1
PF0:ADC0
AREF
AVSS:ASVSS
AVDD
EVDD
AVSS
XTAL1
XTAL2
PE7:ICP3:INT7:CLKO
PE6:T3:INT6
PE5:OC3C:INT5
PE4:OC3B:INT4
PE3:OC3A:AIN1
PE2:XCK0:AIN0
PE1:TXD0:PDO
PE0:RXD0:PDI:PCINT8
DEVDD
DEVDD
DEVDD
DEVDD
DVSS
DVSS
DVSS
DVSS
Paddle
R1
10k
D1
C2
10n
C3
22p
Q1
CFPX-157
(32kHz)
C5
22p
C7
1uF
C9
100n
DGND
MetalMount
Mounting Hole
C4
1uF
V_RCB
L1
Wuerth74279266
Q2
16MHz / CL=10pF
C6
10p
C8
100n
C10
10p
T1
ITT KSR211OLFS
R5
470
R6
470
R7
470
R10
470
D4
D3
D2
D5
red
RSTON
C12
100n
C13
100n
C14
100n
C15
100n
RF1
REF1TOP
RF2
REF2TOP
SW1
Ear09 10201.02
Plug1
AMP 2-331677-2
Bat1
AAAx2
Plug2
AMP 2-331677-2
C16
47uF/6V
C18
100n
C19
100n
CLKI
TST
R8
10k
R9
10k
Title RCB128RFA1

Decoupling Capacitor Measurement

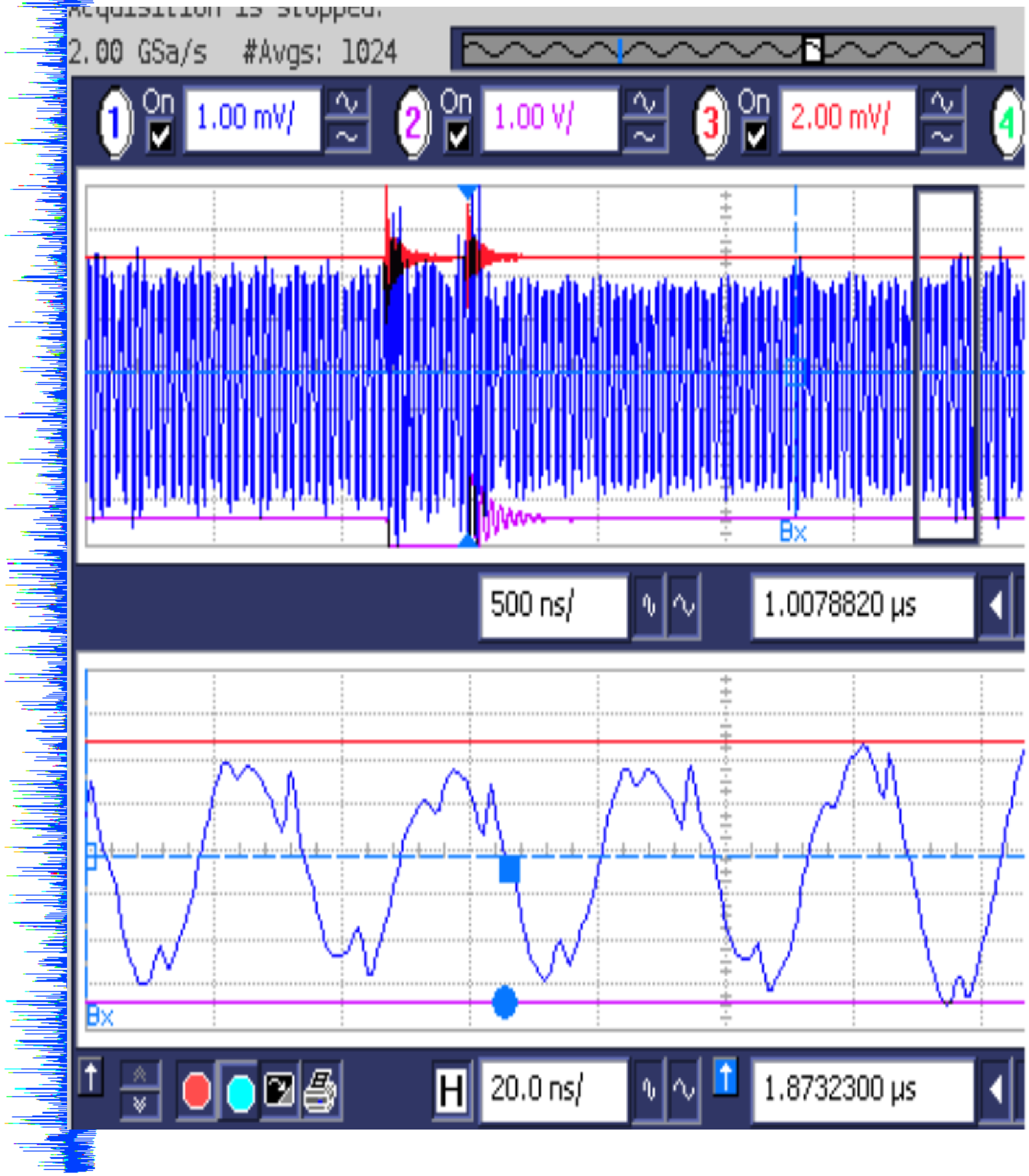


0402 Capacitor on SASEBO-GII

Decoupling Capacitor Measurement



Decoupling Capacitor Measurement



Acquisition is stopped.

2.00 GSa/s #Avgs: 1024

1 On ☑ 1.00 mV/

2 On ☑ 1.00 V/

3 On ☑ 2.00 mV/

4

500 ns/ 1.0078820 µs

H 20.0 ns/ 1.8732300 µs

Current Shunt

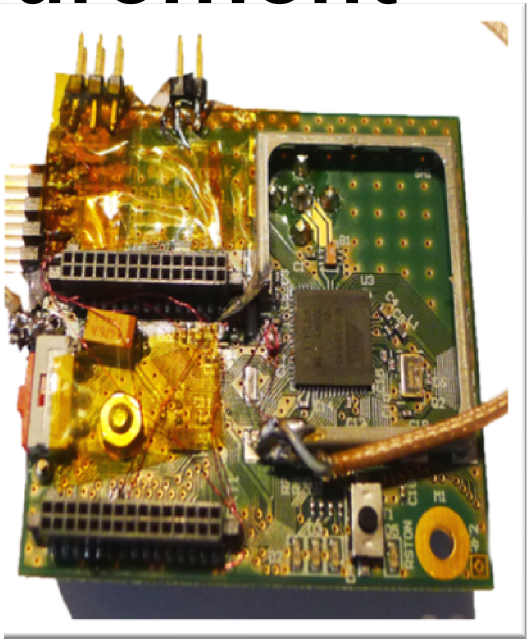
Inductive Pickup

Sync

Current Shunt

Inductive Pickup

Sync



CHIP WHISPERER

Listen to your Inner Hardware

INTRODUCING CHIPWHISPERER

www.ChipWhisperer.com

- GIT Repository for tools shown here
- GIT Repository for hardware designs
- Mailing List for discussion
- Wiki for Documentation
- Tools Licensed via GPL-V3 (aggressively enforced)

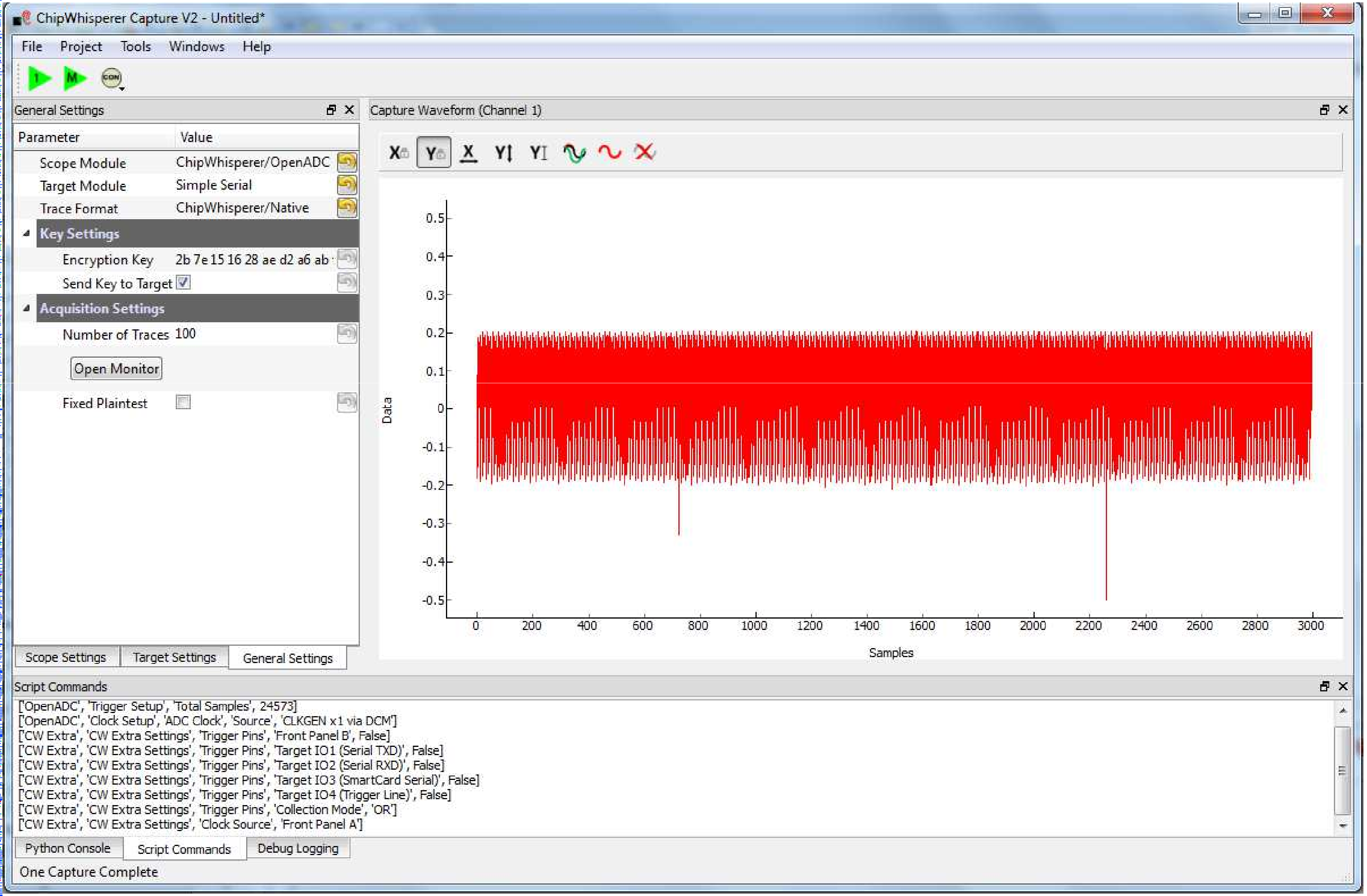
Design “Principles”

- Avoids forcing users into a corner
  - Can run tools from a script (no need to use GUI)
  - Can export data to MATLAB or anything else (no need to use my format)
  - Supports various other hardware

Why Python?

- Easy to understand/modify
- Good scientific libraries (scipy/numpy)
- Cross-Platform (Linux/Mac/Windows)
- Simple GUI programming
- Scriptable & Can use Interpreter
  - Will demonstrate how useful for debugging
- Good support for interfacing to other languages
  - Write high-performance code in C, send data to MATLAB, etc

CW-Capture v2 Features



Parameter	Value
Scope Module	ChipWhisperer/OpenADC
Target Module	Simple Serial
Trace Format	ChipWhisperer/Native
Key Settings	
Encryption Key	2b 7e 15 16 28 ae d2 a6 ab
Send Key to Target	☑
Acquisition Settings	
Number of Traces	100
Open Monitor	
Fixed Plaintest	□

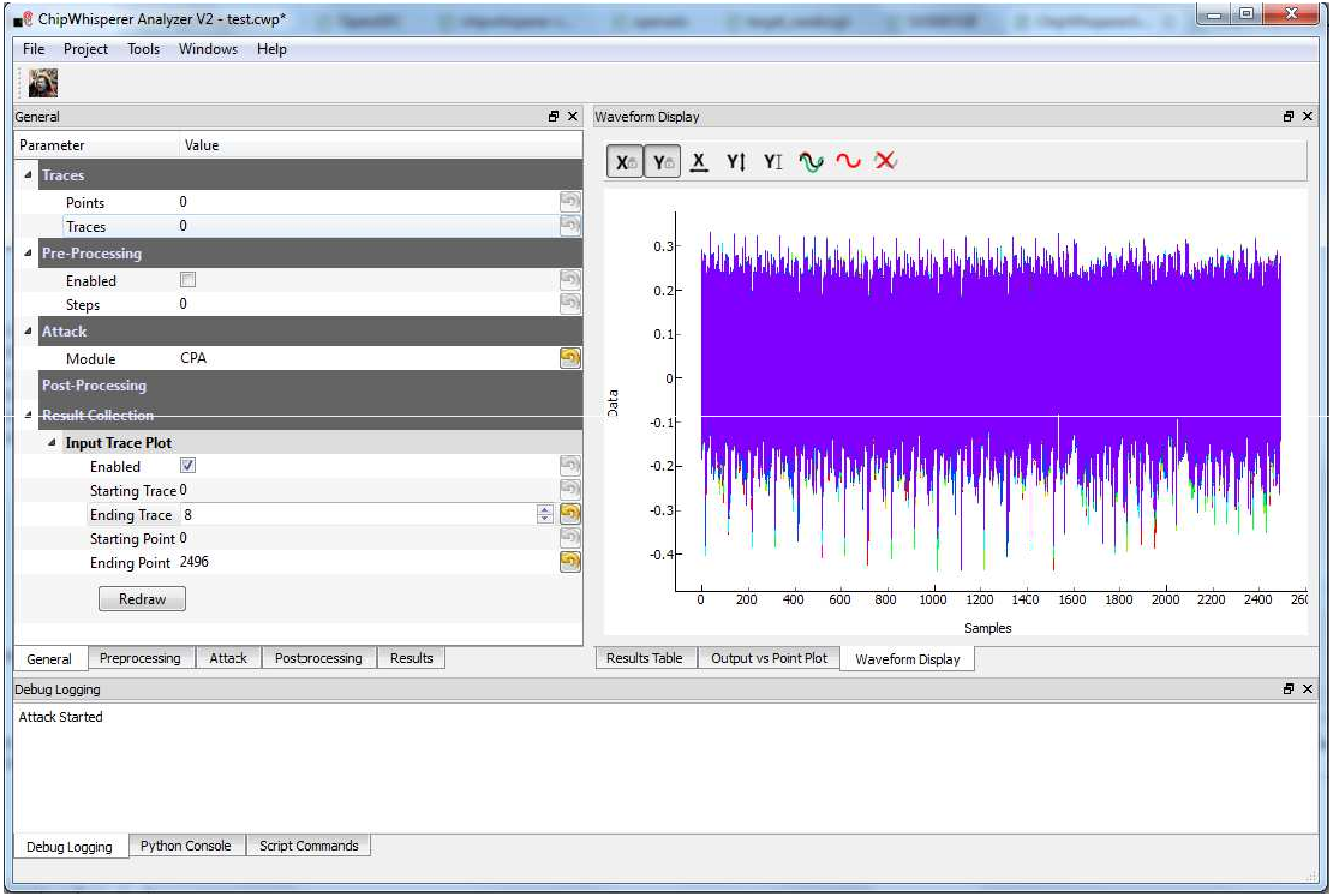
```
['OpenADC', 'Trigger Setup', 'Total Samples', 24573]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'CLKGEN x1 via DCM']
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel B', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO1 (Serial TXD)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO2 (Serial RXD)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO3 (SmartCard Serial)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Collection Mode', 'OR']
['CW Extra', 'CW Extra Settings', 'Clock Source', 'Front Panel A']
```

One Capture Complete

CW-Capture v2 Features

- Supported Scopes:
  - OpenADC via SASEBO-W ($$$)
  - ChipWhisperer Rev2 Capture HW ($$$)
  - Avnet LX9 Microboard ($)
- Supported Targets:
  - PC/SC SmartCard Readers
  - System Serial Port
  - ChipWhisperer-specific extensions (incl. SASEBO-W)
  - SASEBO-GII Board

CW-Analyzer V2 Features



ChipWhisperer Analyzer V2 - test.cwp*

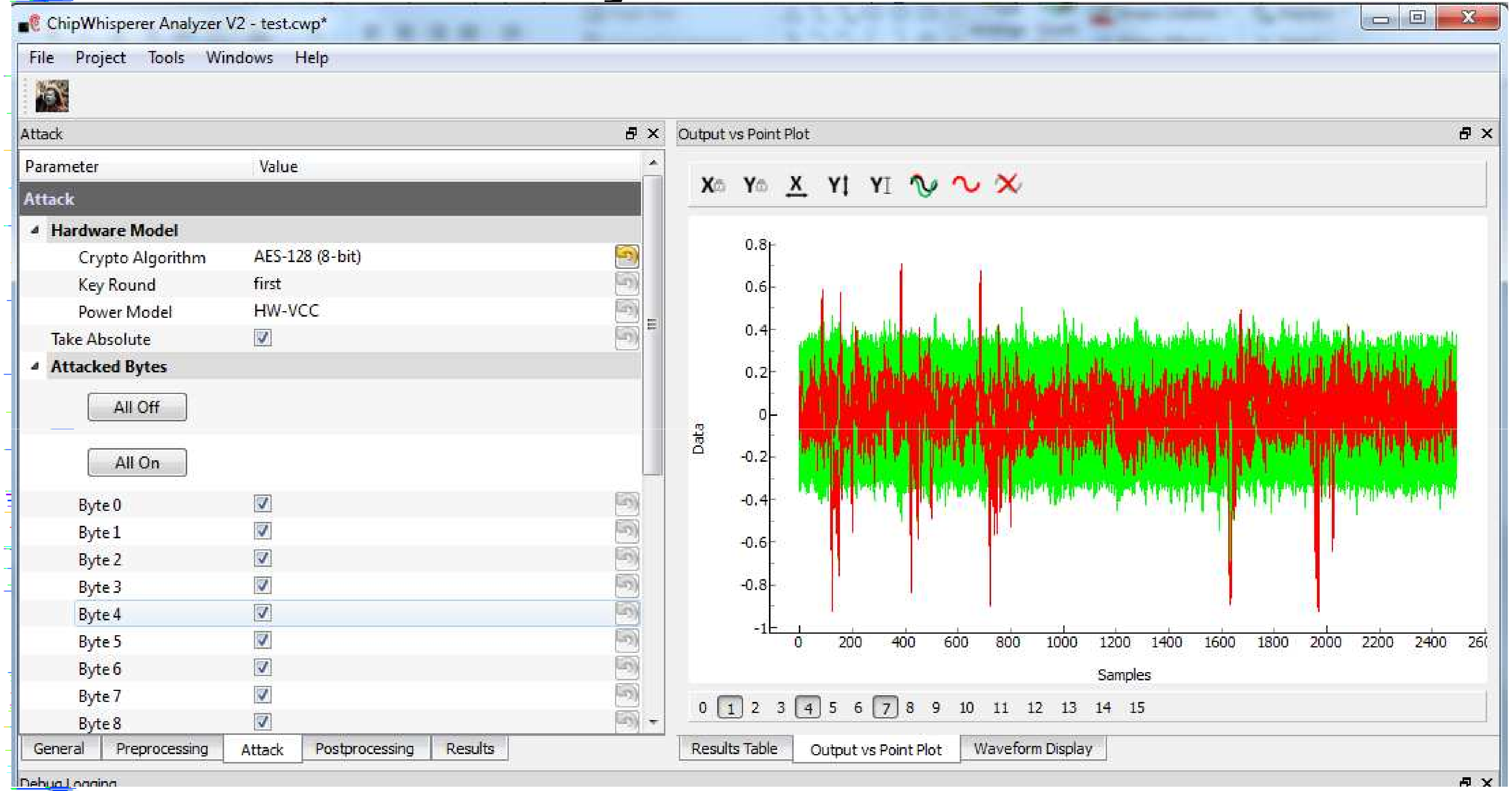
Parameter	Value
Traces	
Points	0
Traces	0
Pre-Processing	
Enabled	□
Steps	0
Attack	
Module	CPA
Post-Processing	
Result Collection	
Input Trace Plot	
Enabled	☑
Starting Trace	0
Ending Trace	8
Starting Point	0
Ending Point	2496

Debug Logging

```
Attack Started
```

Plot saved traces including performing averaging, FFTs

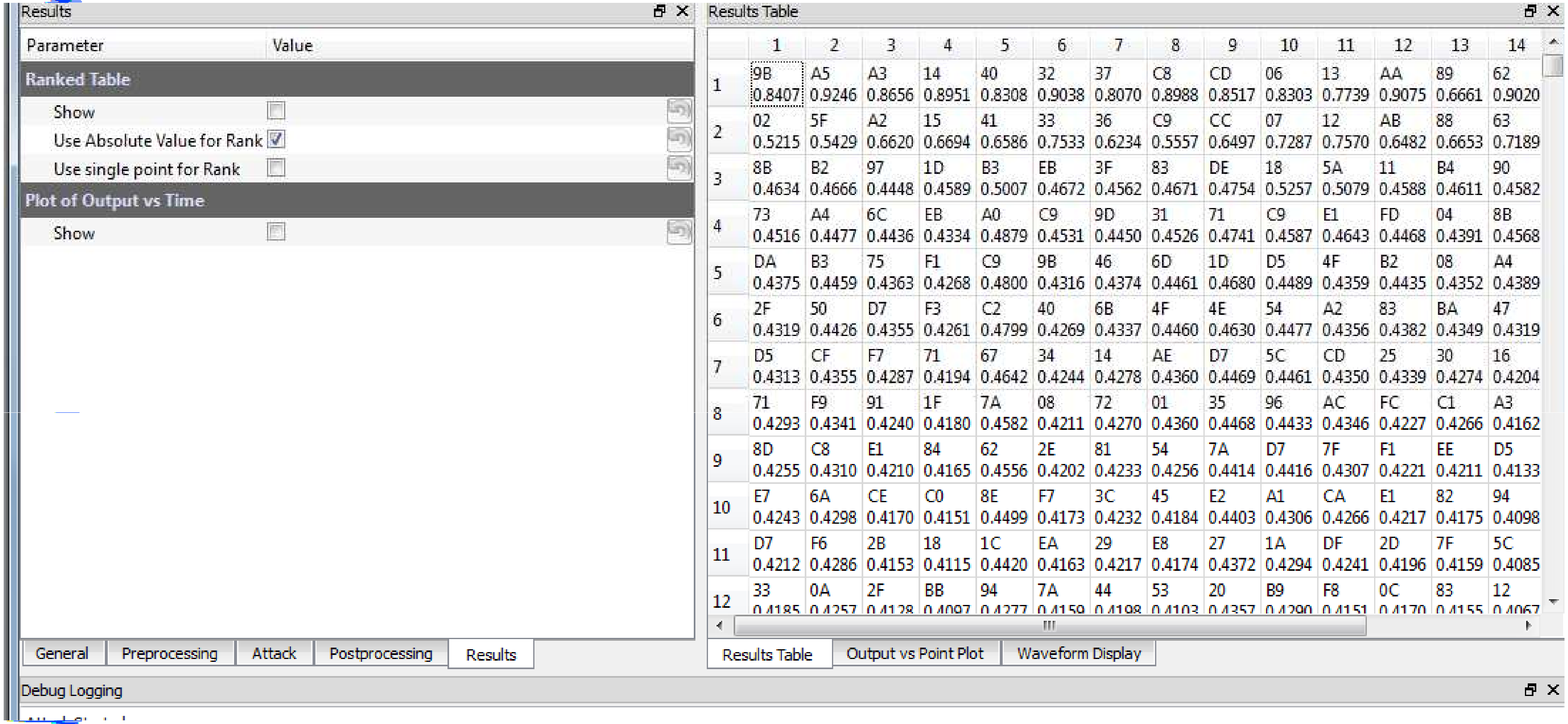
CW-Analyzer V2 Features



Parameter	Value
Attack	
Hardware Model	
Crypto Algorithm	AES-128 (8-bit)
Key Round	first
Power Model	HW-VCC
Take Absolute	☑
Attacked Bytes	
Byte 0	☑
Byte 1	☑
Byte 2	☑
Byte 3	☑
Byte 4	☑
Byte 5	☑
Byte 6	☑
Byte 7	☑
Byte 8	☑

Run attack(s), plot outputs in different formats. Include/exclude bytes in plot. Narrow down on areas of interest, transfer that back to capture for more efficiency.

CW-Analyzer V2 Features



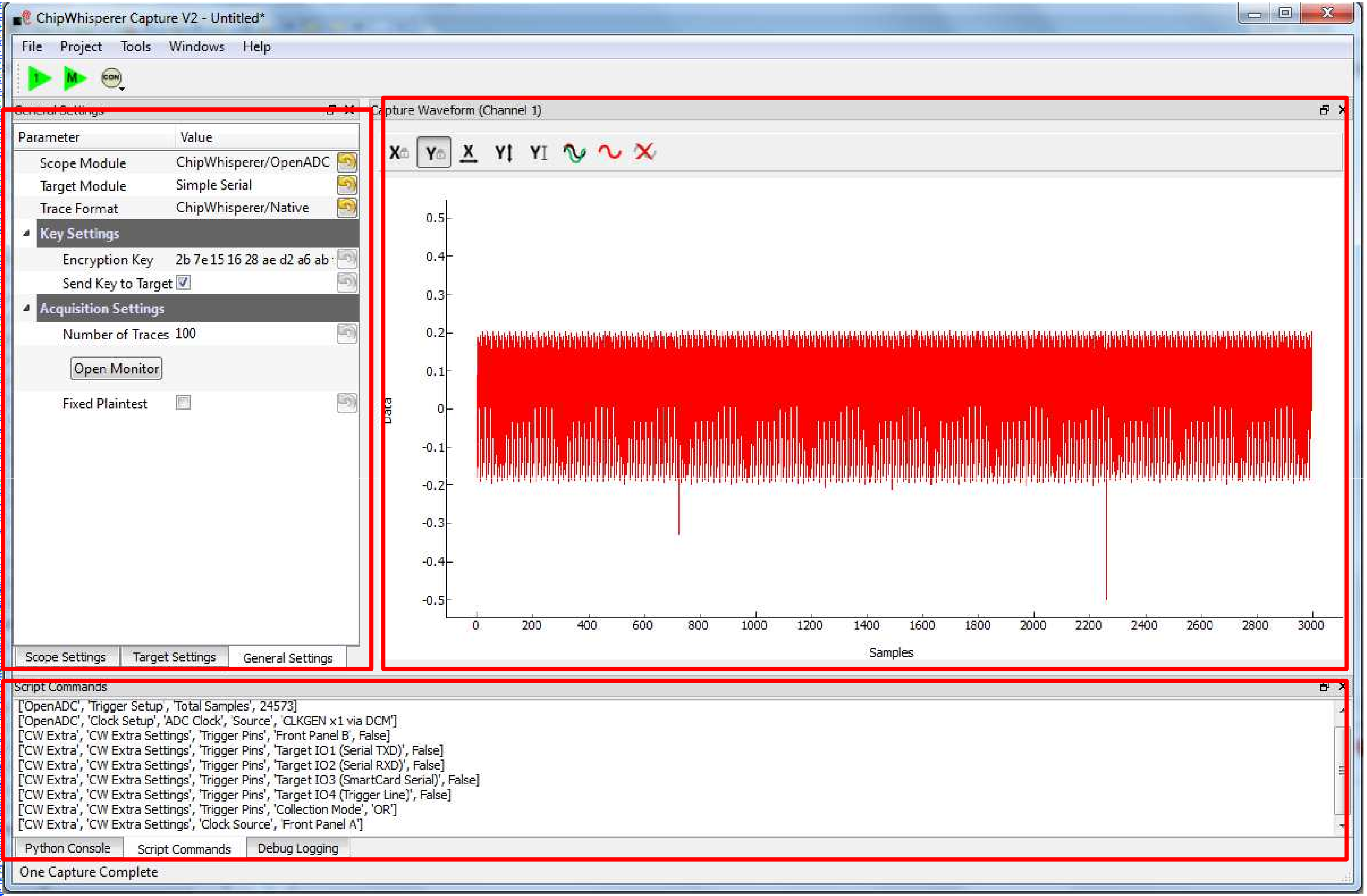
Parameter	Value
Ranked Table	
Show	□
Use Absolute Value for Rank	☑
Use single point for Rank	□
Plot of Output vs Time	
Show	□

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	9B 0.8407	A5 0.9246	A3 0.8656	14 0.8951	40 0.8308	32 0.9038	37 0.8070	C8 0.8988	CD 0.8517	06 0.8303	13 0.7739	AA 0.9075	89 0.6661	62 0.9020
2	02 0.5215	5F 0.5429	A2 0.6620	15 0.6694	41 0.6586	33 0.7533	36 0.6234	C9 0.5557	CC 0.6497	07 0.7287	12 0.7570	AB 0.6482	88 0.6653	63 0.7189
3	8B 0.4634	B2 0.4666	97 0.4448	1D 0.4589	B3 0.5007	EB 0.4672	3F 0.4562	83 0.4671	DE 0.4754	18 0.5257	5A 0.5079	11 0.4588	B4 0.4611	90 0.4582
4	73 0.4516	A4 0.4477	6C 0.4436	EB 0.4334	A0 0.4879	C9 0.4531	9D 0.4450	31 0.4526	71 0.4741	C9 0.4587	E1 0.4643	FD 0.4468	04 0.4391	8B 0.4568
5	DA 0.4375	B3 0.4459	75 0.4363	F1 0.4268	C9 0.4800	9B 0.4316	46 0.4374	6D 0.4461	1D 0.4680	D5 0.4489	4F 0.4359	B2 0.4435	08 0.4352	A4 0.4389
6	2F 0.4319	50 0.4426	D7 0.4355	F3 0.4261	C2 0.4799	40 0.4269	6B 0.4337	4F 0.4460	4E 0.4630	54 0.4477	A2 0.4356	83 0.4382	BA 0.4349	47 0.4319
7	D5 0.4313	CF 0.4355	F7 0.4287	71 0.4194	67 0.4642	34 0.4244	14 0.4278	AE 0.4360	D7 0.4469	5C 0.4461	CD 0.4350	25 0.4339	30 0.4274	16 0.4204
8	71 0.4293	F9 0.4341	91 0.4240	1F 0.4180	7A 0.4582	08 0.4211	72 0.4270	01 0.4360	35 0.4468	96 0.4433	AC 0.4346	FC 0.4227	C1 0.4266	A3 0.4162
9	8D 0.4255	C8 0.4310	E1 0.4210	84 0.4165	62 0.4556	2E 0.4202	81 0.4233	54 0.4256	7A 0.4414	D7 0.4416	7F 0.4307	F1 0.4221	EE 0.4211	D5 0.4133
10	E7 0.4243	6A 0.4298	CE 0.4170	C0 0.4151	8E 0.4499	F7 0.4173	3C 0.4232	45 0.4184	E2 0.4403	A1 0.4306	CA 0.4266	E1 0.4217	82 0.4175	94 0.4098
11	D7 0.4212	F6 0.4286	2B 0.4153	18 0.4115	1C 0.4420	EA 0.4163	29 0.4217	E8 0.4174	27 0.4372	1A 0.4294	DF 0.4241	2D 0.4196	7F 0.4159	5C 0.4085
12	33 0.4185	0A 0.4257	2F 0.4128	BB 0.4097	94 0.4277	7A 0.4159	44 0.4198	53 0.4103	20 0.4357	B9 0.4290	F8 0.4151	0C 0.4170	83 0.4155	12 0.4067

Tabular results display.

ChipWhispererAnalyzer v2 Software

- Supported Scopes:
  - OpenADC via SASEBO-W ($$$)
  - ChipWhisperer Rev2 Capture HW ($$$)
  - Avnet LX9 Microboard ($)
- Supported Targets:
  - PC/SC SmartCard Readers
  - System Serial Port
  - ChipWhisperer-specific extensions (incl. SASEBO-W)
  - SASEBO-GII Board

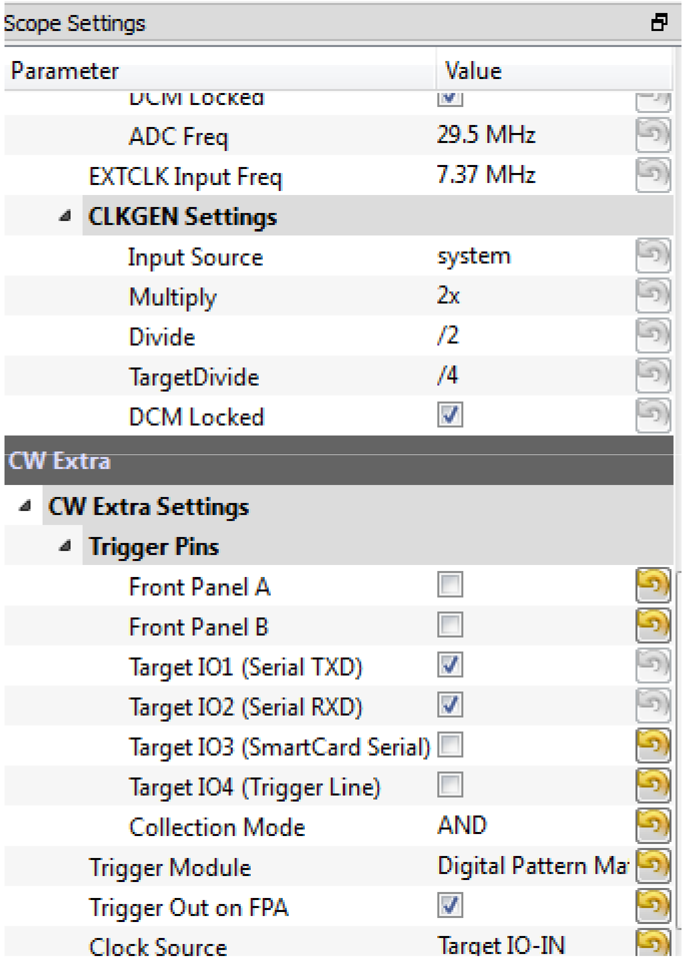


Parameter	Value
Scope Module	ChipWhisperer/OpenADC
Target Module	Simple Serial
Trace Format	ChipWhisperer/Native
Key Settings	
Encryption Key	2b 7e 15 16 28 ae d2 a6 ab
Send Key to Target	☑
Acquisition Settings	
Number of Traces	100
Fixed Plaintest	□

```
['OpenADC', 'Trigger Setup', 'Total Samples', 24573]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'CLKGEN x1 via DCM']
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel B', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO1 (Serial TXD)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO2 (Serial RXD)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO3 (SmartCard Serial)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Collection Mode', 'OR']
['CW Extra', 'CW Extra Settings', 'Clock Source', 'Front Panel A']
```

One Capture Complete

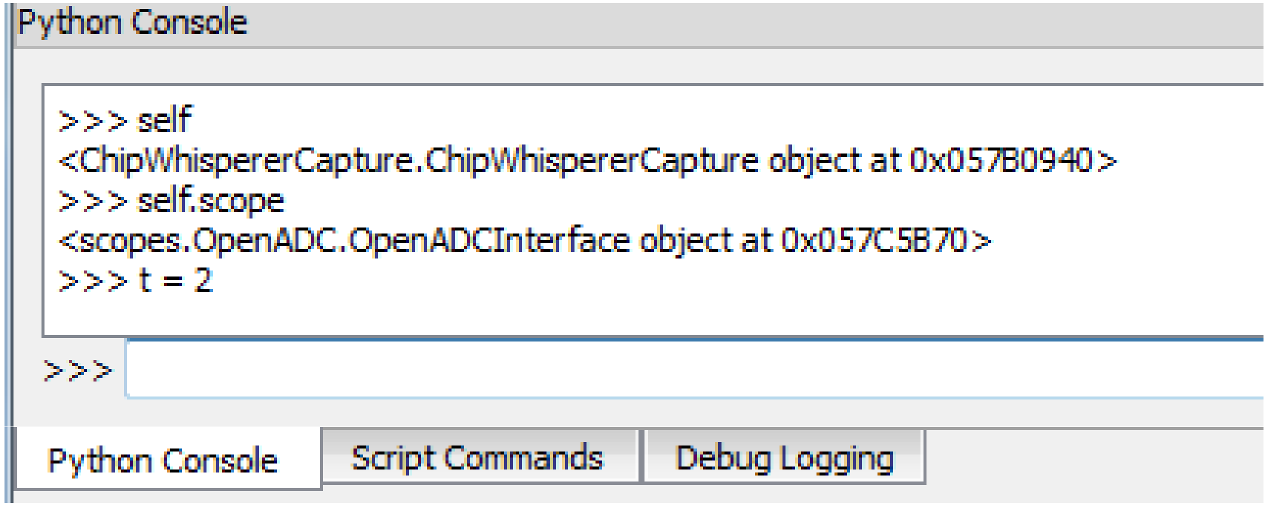
GUI Features



Scope Settings

Parameter	Value
DCM Locked	☑
ADC Freq	29.5 MHz
EXTCLK Input Freq	7.37 MHz
CLKGEN Settings	
Input Source	system
Multiply	2x
Divide	/2
TargetDivide	/4
DCM Locked	☑
CW Extra	
CW Extra Settings	
Trigger Pins	
Front Panel A	□
Front Panel B	□
Target IO1 (Serial TXD)	☑
Target IO2 (Serial RXD)	☑
Target IO3 (SmartCard Serial)	□
Target IO4 (Trigger Line)	□
Collection Mode	AND
Trigger Module	Digital Pattern Ma
Trigger Out on FPA	☑
Clock Source	Target IO-IN

GUI Features

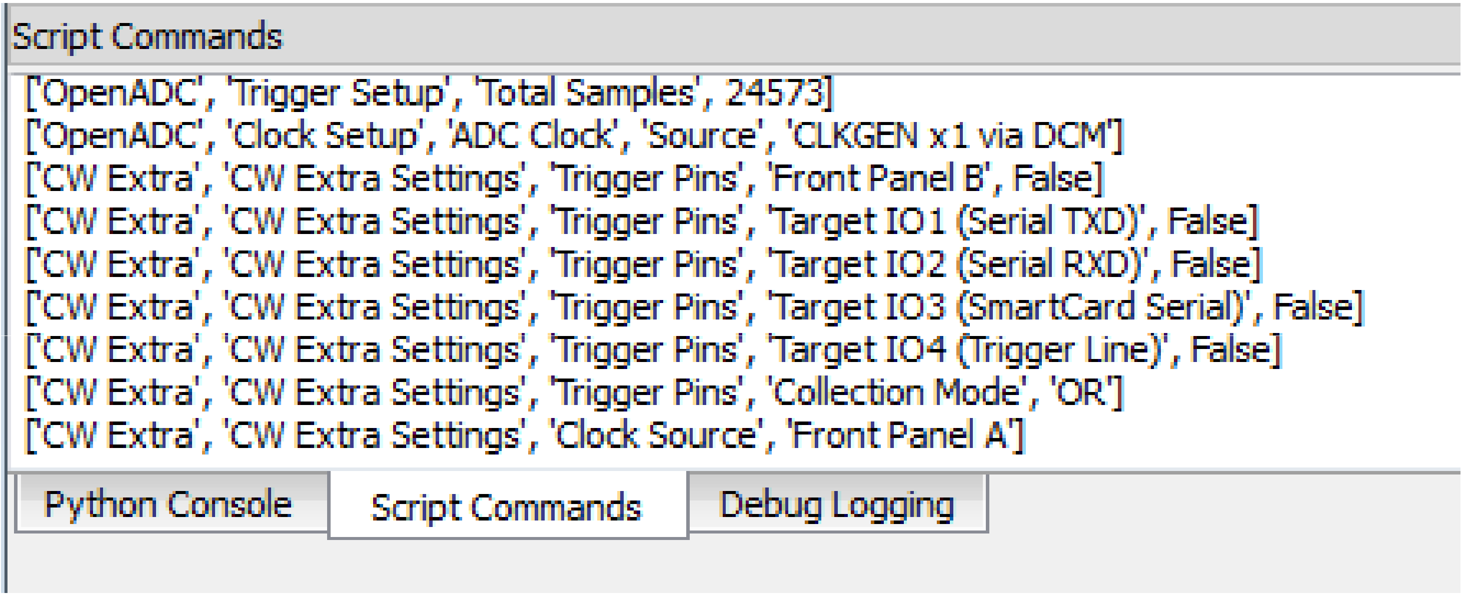


```
>>> self
<ChipWhispererCapture.ChipWhispererCapture object at 0x057B0940>
>>> self.scope
<scopes.OpenADC.OpenADCInterface object at 0x057C5B70>
>>> t = 2
```

```
>>>
```

- Access/change ANYTHING in running program!
- Load new/experimental modules without proper interface

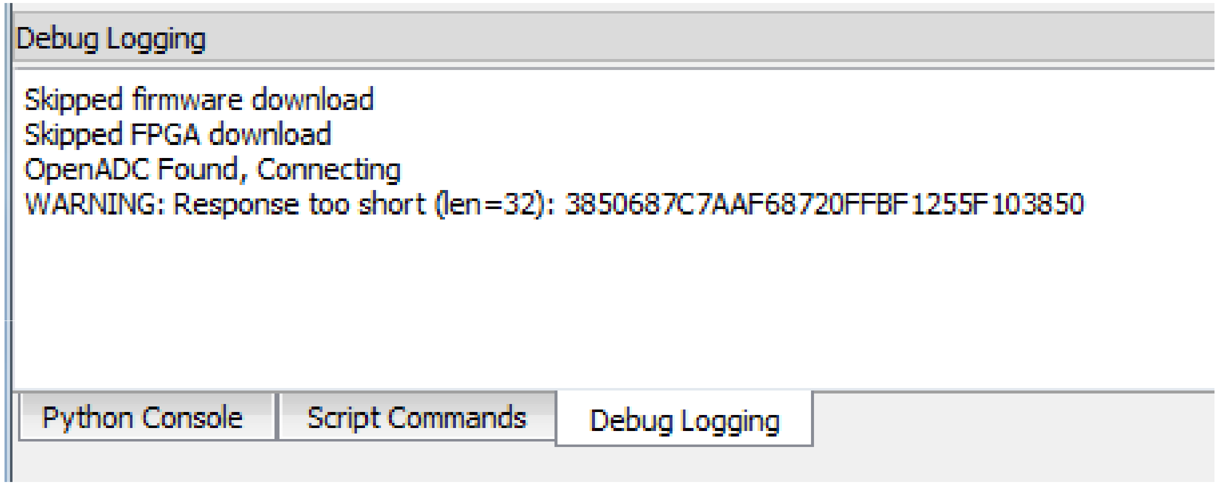
GUI Features



```
['OpenADC', 'Trigger Setup', 'Total Samples', 24573]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'CLKGEN x1 via DCM']
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel B', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO1 (Serial TXD)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO2 (Serial RXD)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO3 (SmartCard Serial)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', False]
['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Collection Mode', 'OR']
['CW Extra', 'CW Extra Settings', 'Clock Source', 'Front Panel A']
```

- Keeps a record of your clicks, used in scripting

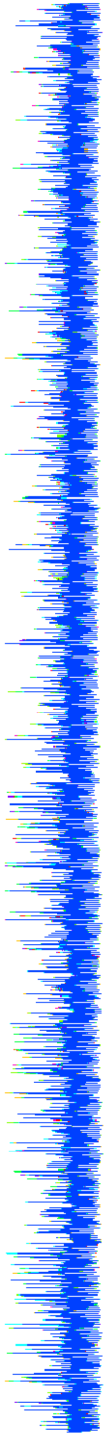
GUI Features



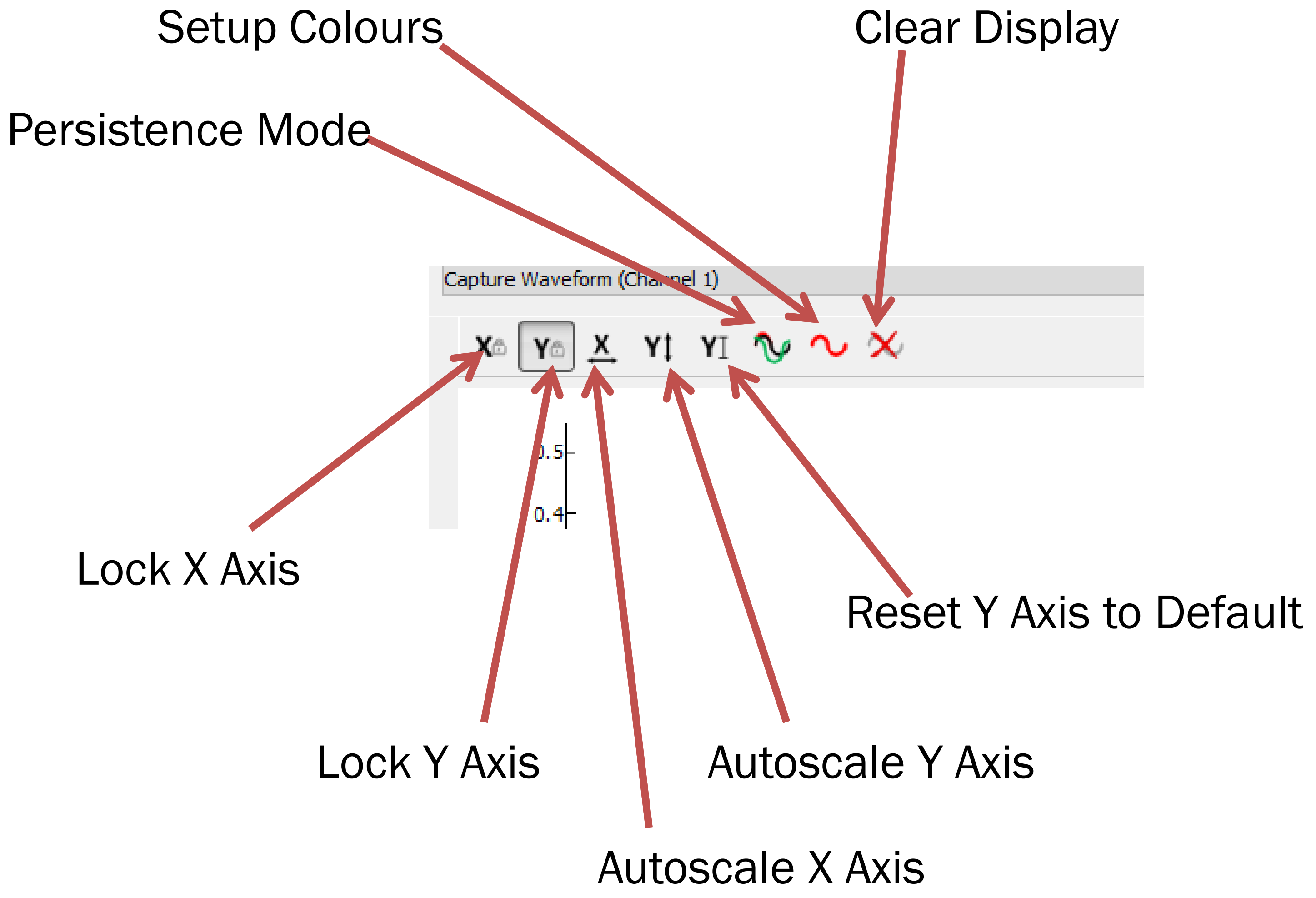
Debug Logging

```
Skipped firmware download
Skipped FPGA download
OpenADC Found, Connecting
WARNING: Response too short (len=32): 3850687C7AAF68720FFBF1255F103850
```

- Random output goes here (or to command line)



Waveform Display Toolbar



Setup Colours

Clear Display

Persistence Mode

Lock X Axis

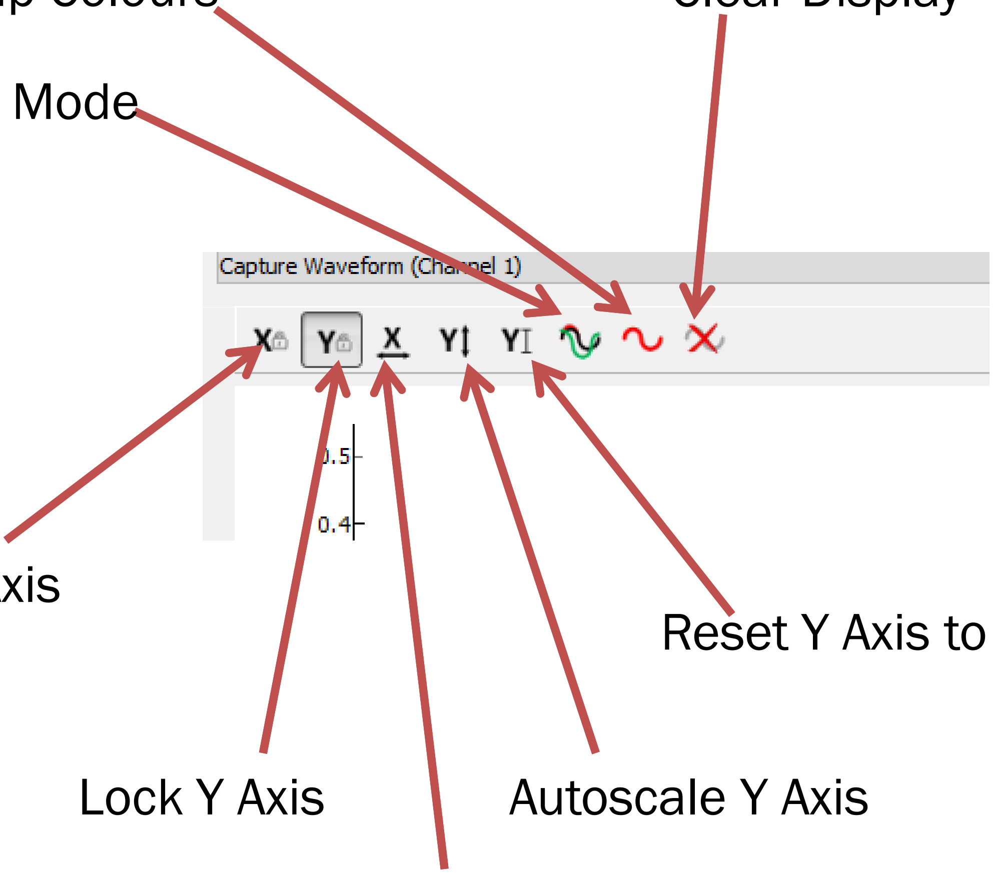
Reset Y Axis to Default

Lock Y Axis

Autoscale Y Axis

Autoscale X Axis

Waveform Display Toolbar



Setup Colours

Clear Display

Persistence Mode

Lock X Axis

Reset Y Axis to Default

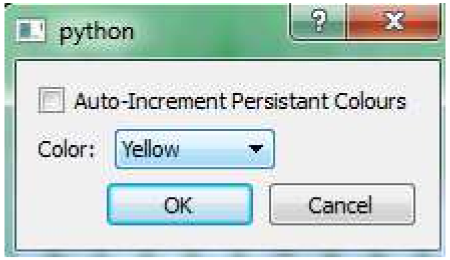
Lock Y Axis

Autoscale Y Axis

Autoscale X Axis

Using Average Mode

1. In Colour Menu, Disable auto-increment, set colour to something bland (like yellow)

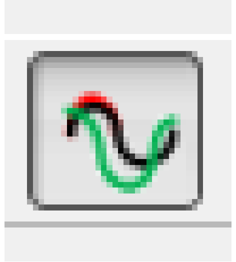


python

□ Auto-Increment Persistant Colours

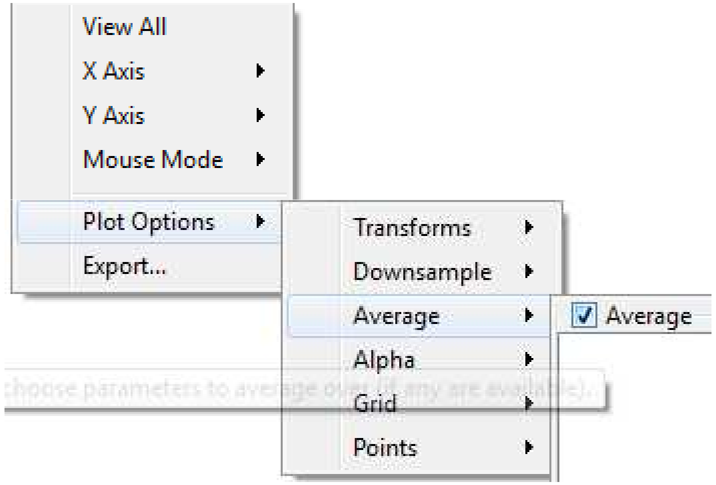
Color: Yellow

2. Set Persistence Mode On



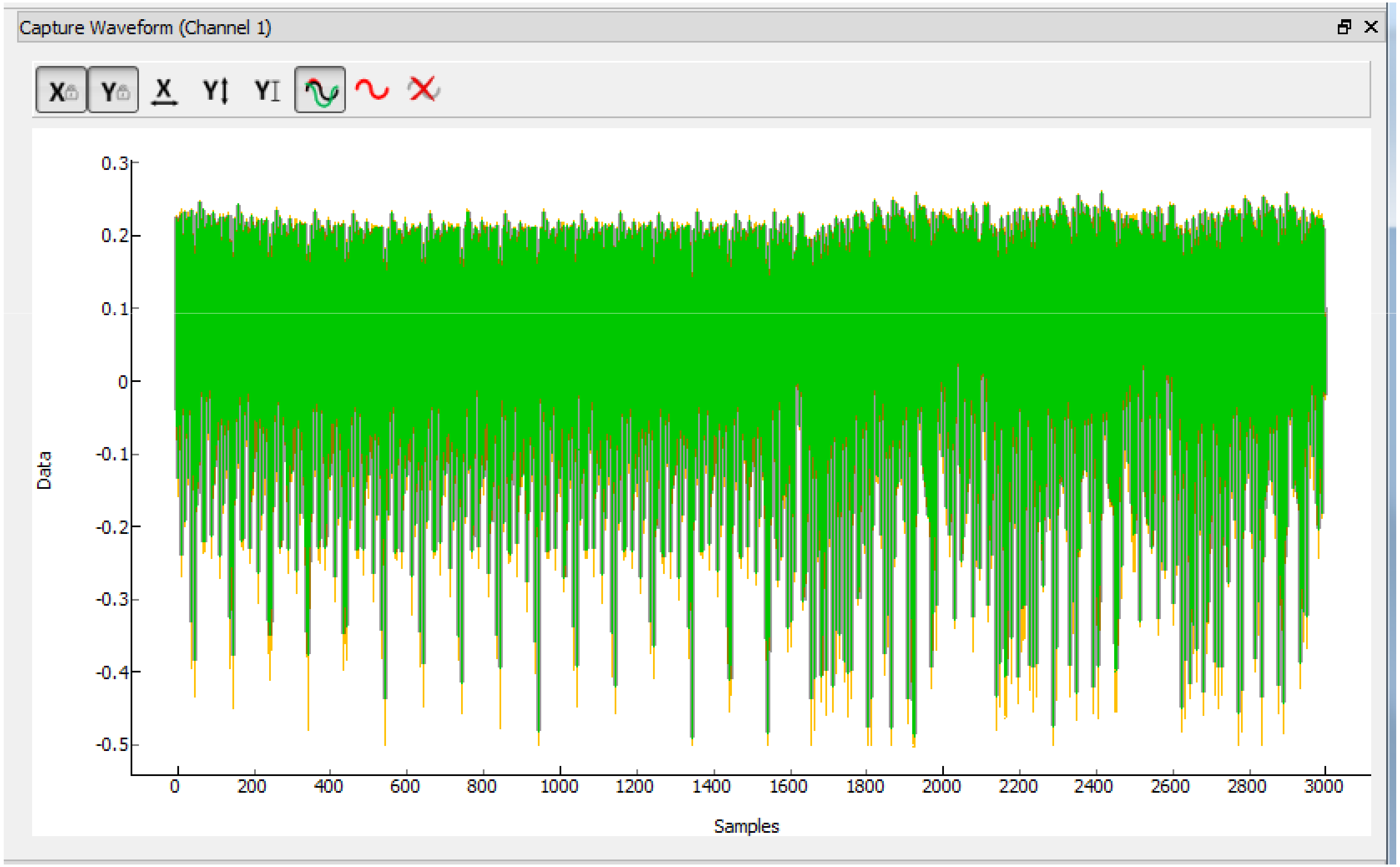
Using Average Mode

3. Right click, enable ‘Average’



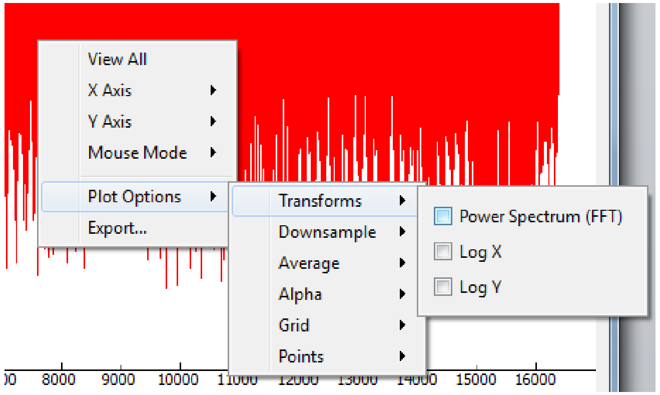
Using Average Mode

4. Plot a bunch of things.

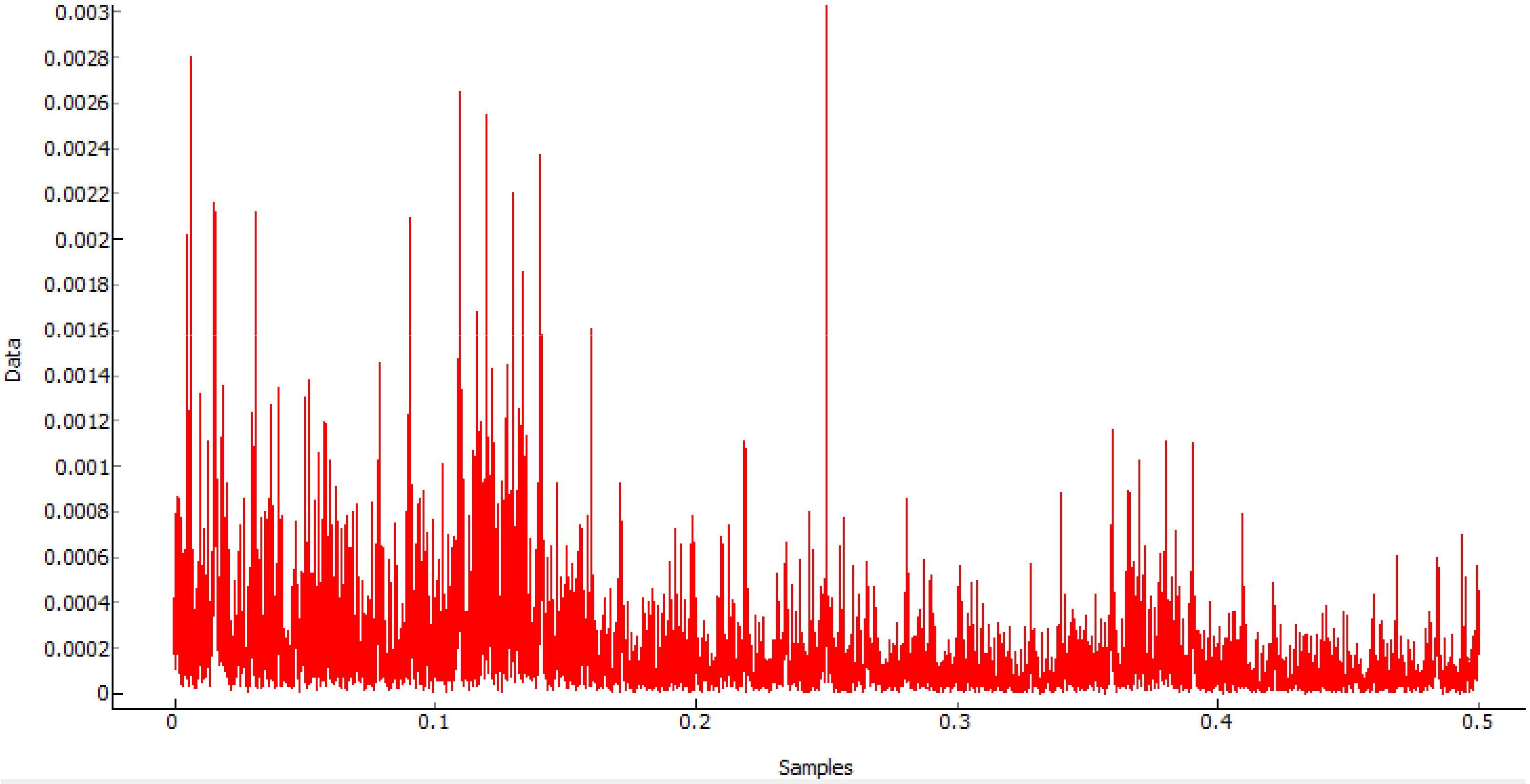


Using Frequency Display Mode

1. Setup display/amplifier as normal, be sure not to have clipping in waveform
2. Set appropriate capture length (ideally some power of two: 4096, 16384, some other multiple)
3. Enable FFT:



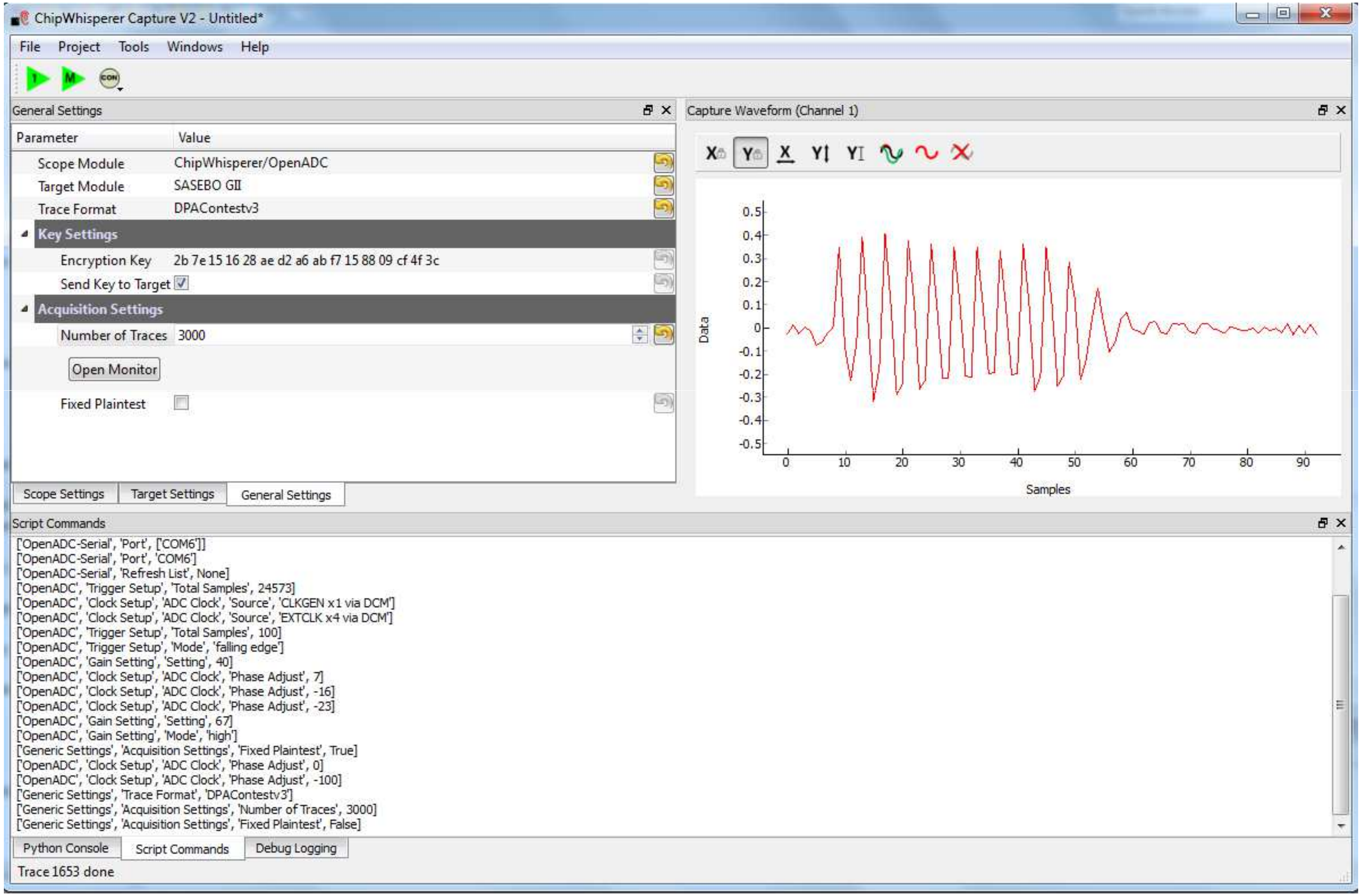
Using Frequency Display Mode



SCRIPTING CW-CAPTURE

Why Script?

- Clicking is boring, error-prone
- Drive CW-Capture from other software
- Easy method of exchanging settings



Parameter	Value
Scope Module	ChipWhisperer/OpenADC
Target Module	SASEBO GII
Trace Format	DPAContestv3
Key Settings	
Encryption Key	2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
Send Key to Target	☑
Acquisition Settings	
Number of Traces	3000
Open Monitor	
Fixed Plaintest	☐

```
['OpenADC-Serial', 'Port', ['COM6']]
['OpenADC-Serial', 'Port', 'COM6']
['OpenADC-Serial', 'Refresh List', None]
['OpenADC', 'Trigger Setup', 'Total Samples', 24573]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'CLKGEN x1 via DCM']
['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'EXTCLK x4 via DCM']
['OpenADC', 'Trigger Setup', 'Total Samples', 100]
['OpenADC', 'Trigger Setup', 'Mode', 'falling edge']
['OpenADC', 'Gain Setting', 'Setting', 40]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Phase Adjust', 7]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Phase Adjust', -16]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Phase Adjust', -23]
['OpenADC', 'Gain Setting', 'Setting', 67]
['OpenADC', 'Gain Setting', 'Mode', 'high']
['Generic Settings', 'Acquisition Settings', 'Fixed Plaintest', True]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Phase Adjust', 0]
['OpenADC', 'Clock Setup', 'ADC Clock', 'Phase Adjust', -100]
['Generic Settings', 'Trace Format', 'DPAContestv3']
['Generic Settings', 'Acquisition Settings', 'Number of Traces', 3000]
['Generic Settings', 'Acquisition Settings', 'Fixed Plaintest', False]
```

Trace 1653 done

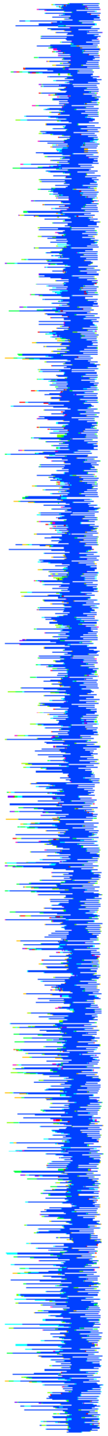
Important Notes/Restrictions

- Only displays things done in the Parameter tabs
- Other functions (e.g. connecting) require use of Python API

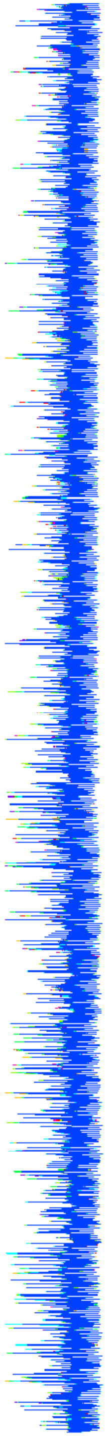
How the Script Works

- Writing a Python program which calls the ChipWhisperer-Capture API

- Special interface to setting parameters (emulates what you do with mouse)

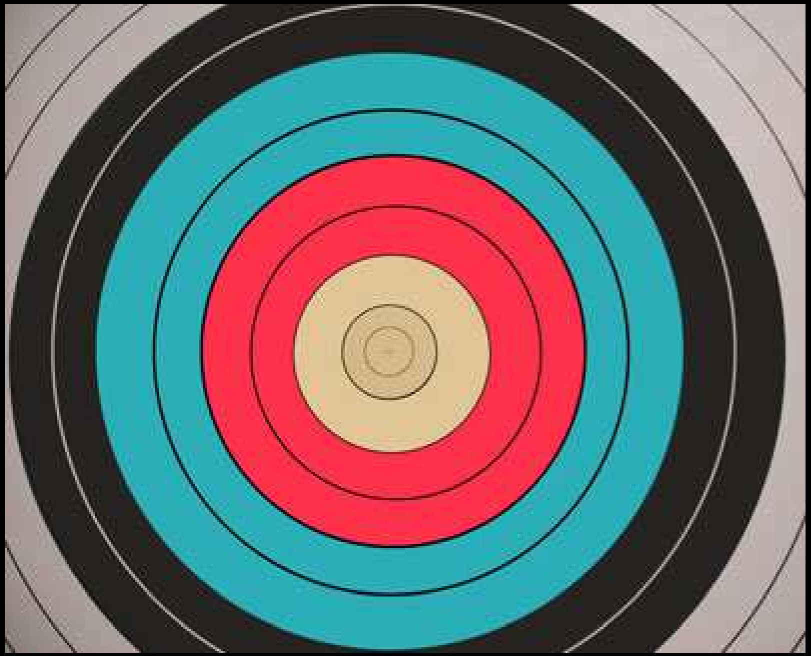


```
#Make the application
app = cwc.makeApplication()

#If you DO NOT want to overwrite/use settings from the GUI version
including
#the recent files list, uncomment the following:
#app.setApplicationName("Capture V2 Scripted")

#Get main module
capture = cwc.ChipWhispererCapture()

#Show window - even if not used
capture.show()

#NB: Must call processEvents since we aren't using proper event loop
pe()
#Call user-specific commands
usercommands = userScript(capture)
usercommands.run()

app.exec_()
sys.exit()
```



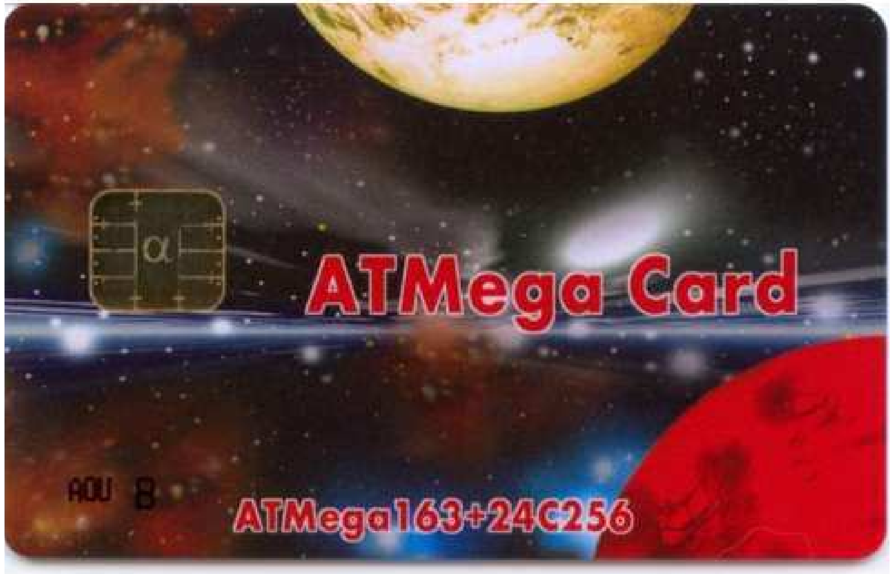
```python
def run(self):
    cap = self.capture

    #User commands here
    print "***** Starting User Script *****"
    cap.setParameter(['Generic Settings', 'Scope Module', 'ChipWhisperer/OpenADC'])
    cap.setParameter(['Generic Settings', 'Target Module', 'Simple Serial'])
    cap.setParameter(['Generic Settings', 'Trace Format', 'ChipWhisperer/Native'])
    cap.setParameter(['Target Connection', 'connection', 'ChipWhisperer'])

    #Load FW (must be configured in GUI first)
    cap.FWLoaderGo()

    #NOTE: You MUST add this call to pe() to process events
    pe()
```

```
cap.doConDis()
pe()

#Example of using a list to set parameters. Slightly easier to copy/paste in this format
lstexample = [['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel A', False],
             ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger
Line)', True],
             ['CW Extra', 'CW Extra Settings', 'Clock Source', 'Target IO-IN'],
             ['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'EXTCLK x4 via DCM'],
             ['OpenADC', 'Trigger Setup', 'Total Samples', 3000],
             ['OpenADC', 'Trigger Setup', 'Offset', 1500],
             ['OpenADC', 'Gain Setting', 'Setting', 45],
             ['OpenADC', 'Trigger Setup', 'Mode', 'rising edge'],
             #Final step: make DCMs relock in case they are lost
             ['OpenADC', 'Clock Setup', 'Relock DCMs', None],
            ]

#Download all hardware setup parameters
for cmd in lstexample: cap.setParameter(cmd)

#Let's only do a few traces
cap.setParameter(['Generic Settings', 'Acquisition Settings', 'Number of Traces', 75])
```



TARGET PRACTICE

ATMega Card
ATMega163+24C256
α


ATMega Card
ATMega163+24C256

=

Original Process Size

Original AVR (~1998) = 0.8um

Mega8 (~2000) = 0.5um

Mega163 (~2000) = 0.5um

Mega128 (~2002) = 0.35um (first 5v 0.35um)

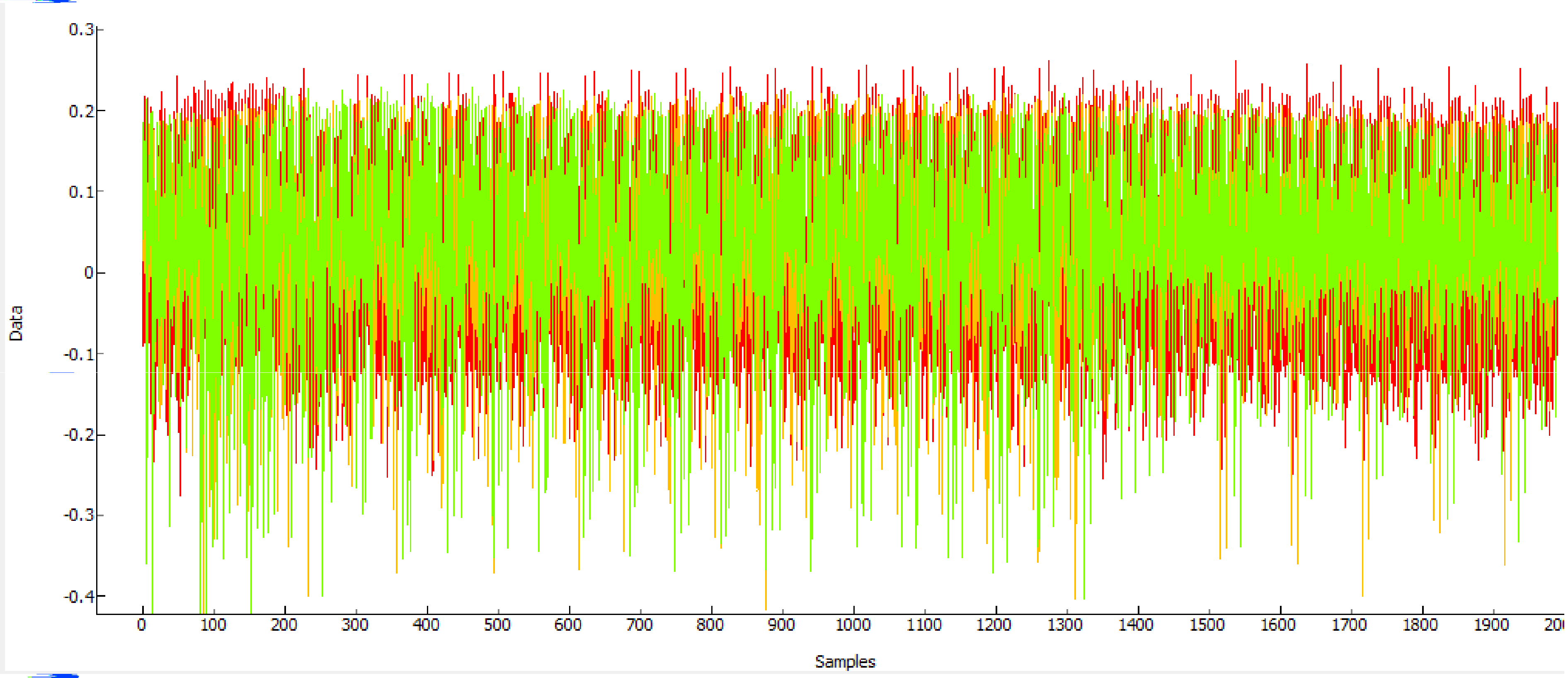
Mega48P (~2007) = 0.35um

Mega48PA (~2011)= 0.18um / 0.12um

(A indicates newer process)

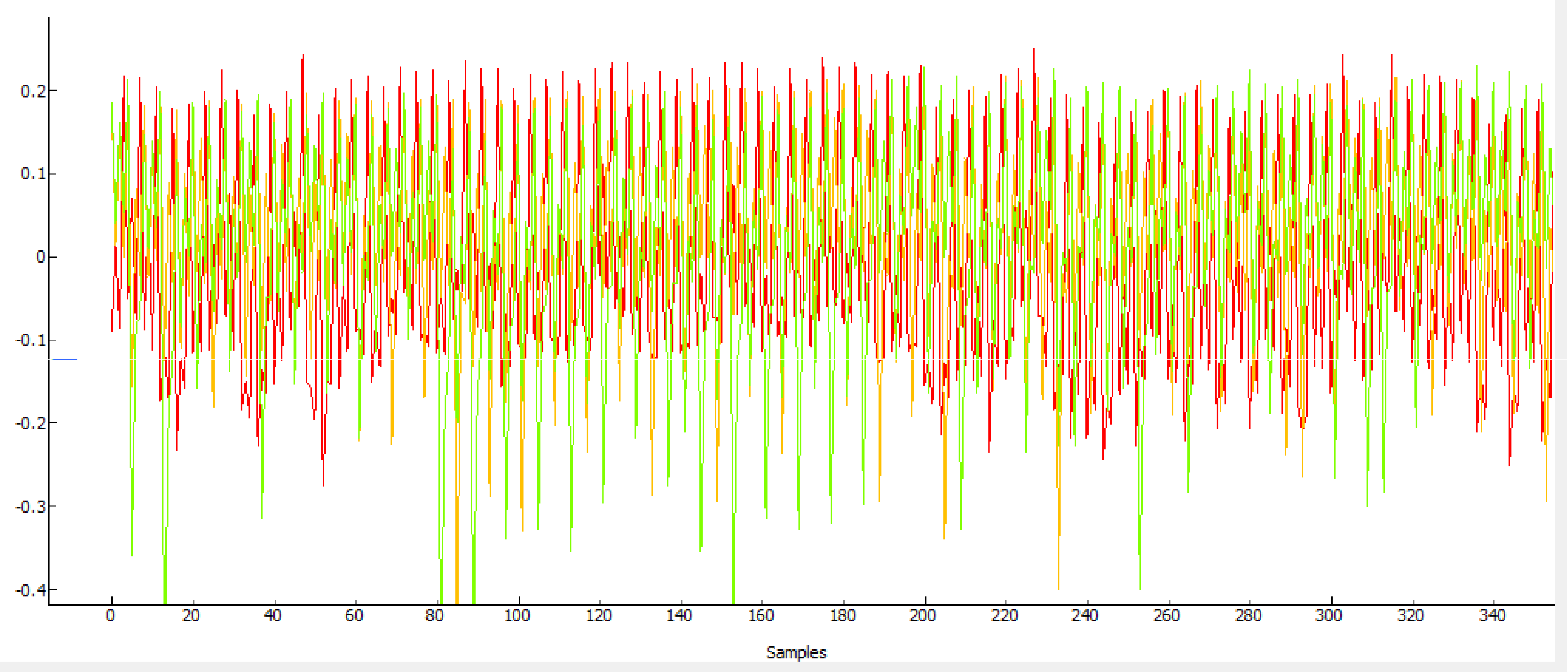
http://zeptobars.ru/en/read/how-to-open-microchip-asic-what-inside

Comparison of Power Signatures



Red = AtMega8, Manufactured 2012, Week 51
Yellow = AtMega48A, Manufactured 2011, Week 31
Green = AtMega328P, Manufactured 2013, Week 10

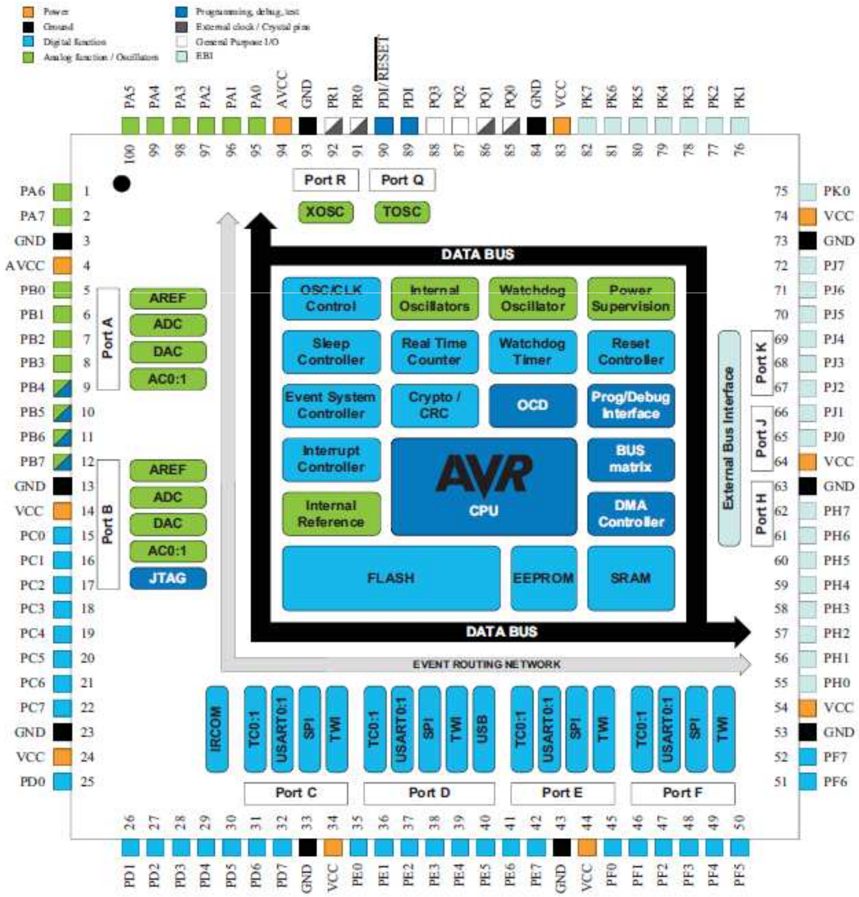
Comparison of Power Signatures



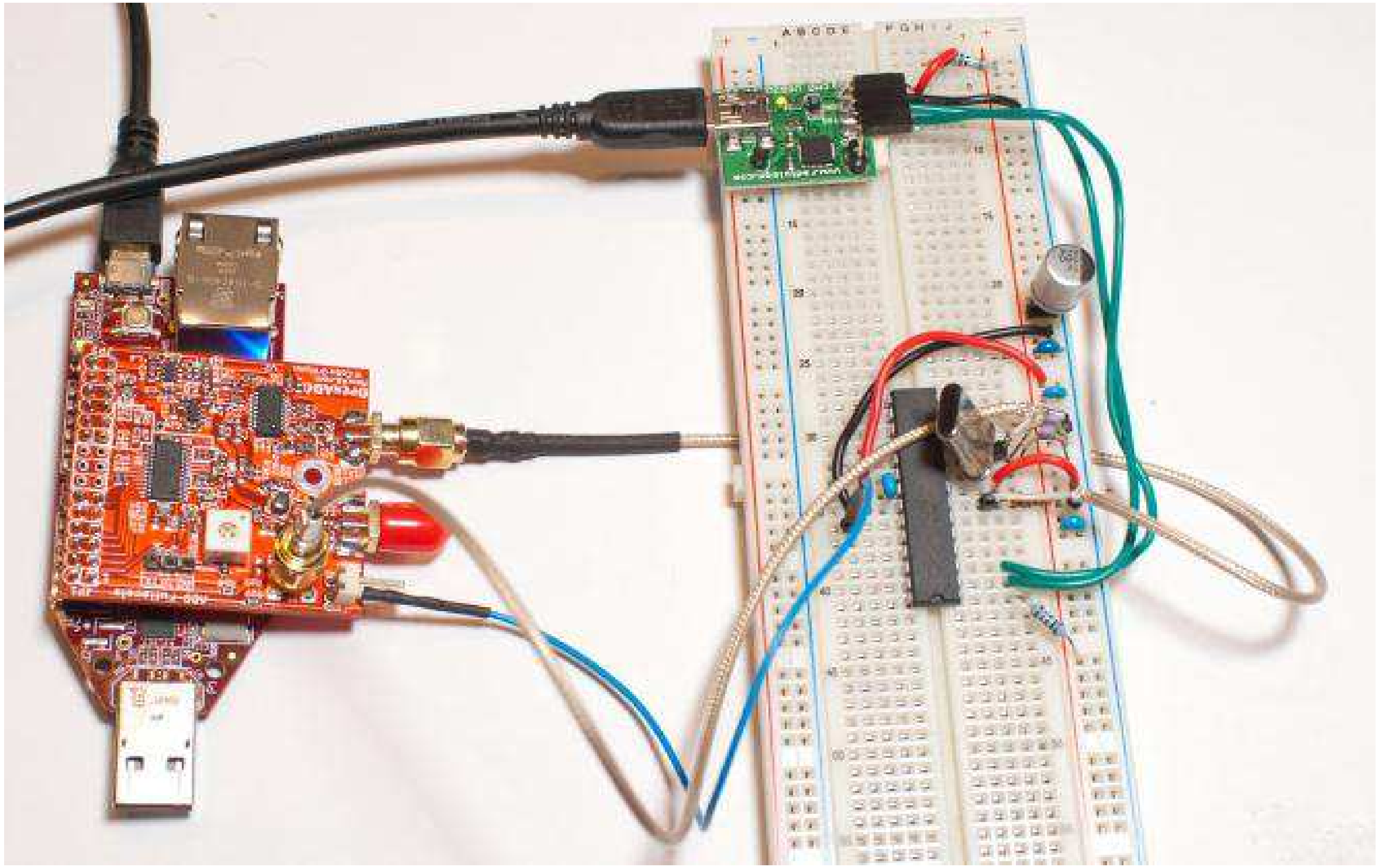
Red = AtMega8, Manufactured 2012, Week 51
Yellow = AtMega48A, Manufactured 2011, Week 31
Green = AtMega328P, Manufactured 2013, Week 10

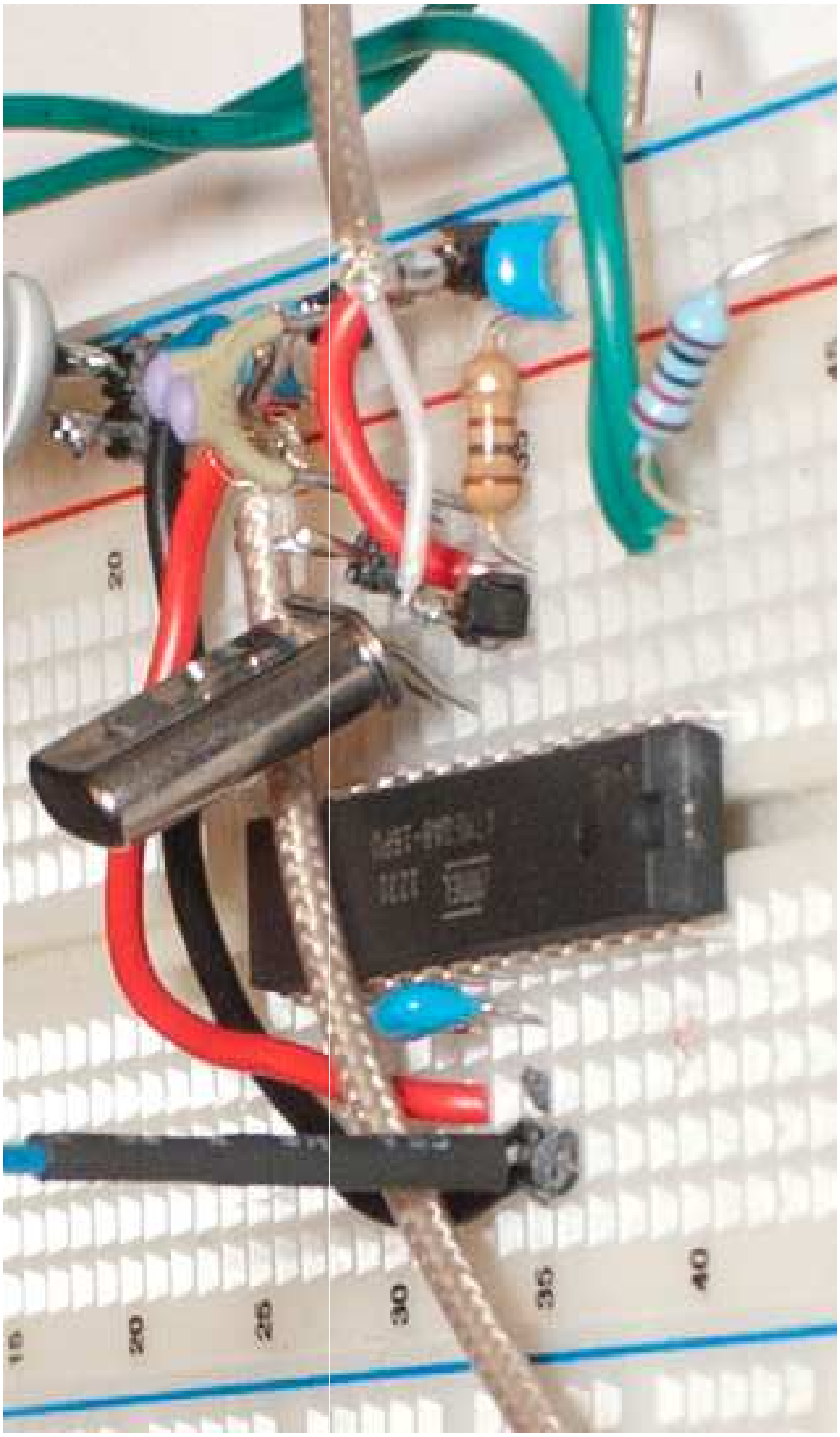
Other Interesting Devices

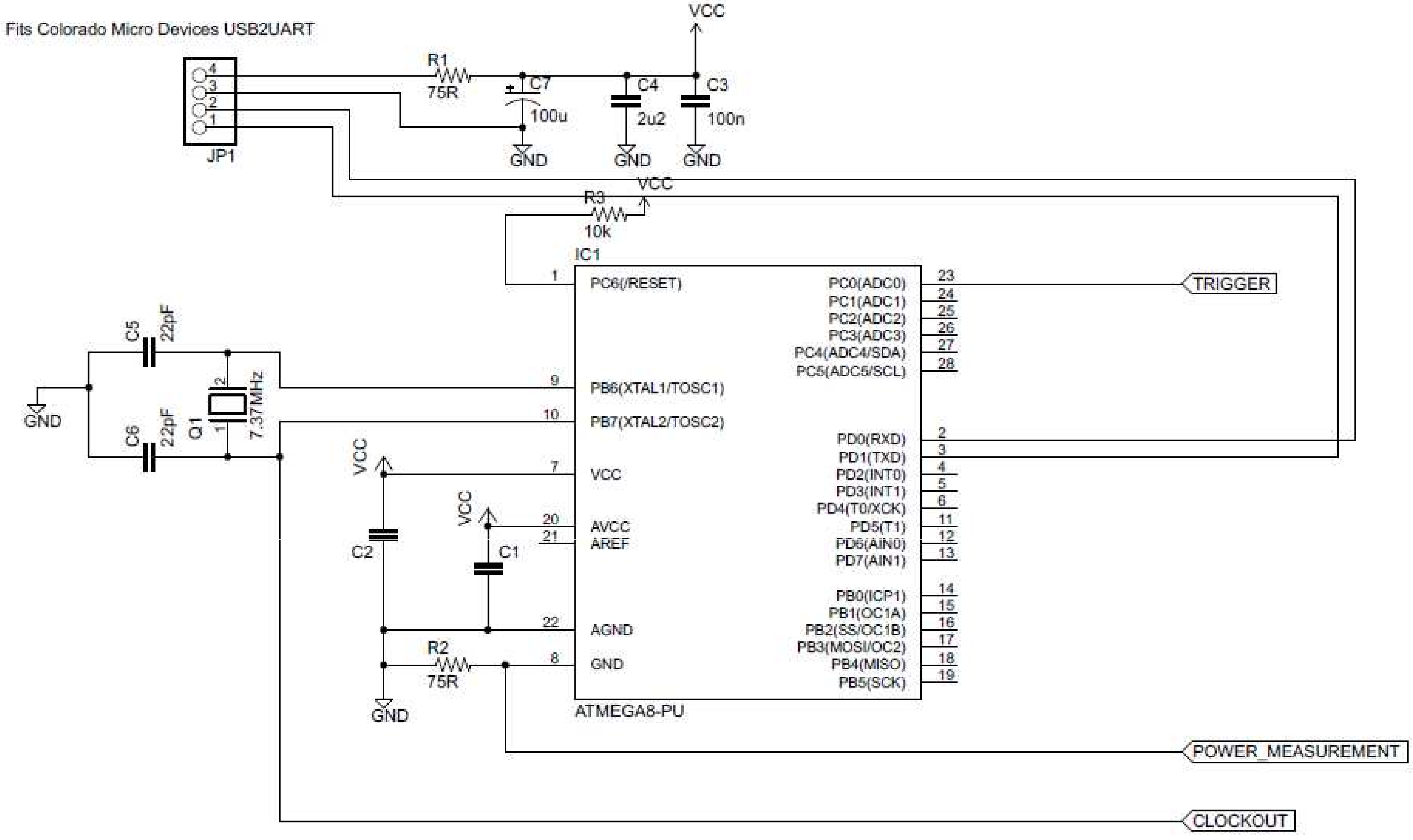
XMega



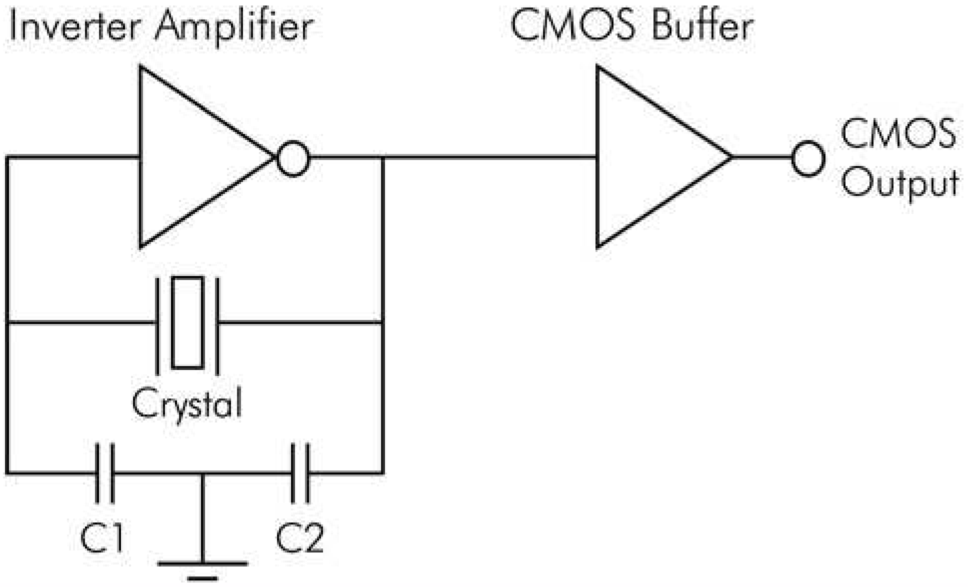
Building a Simple System



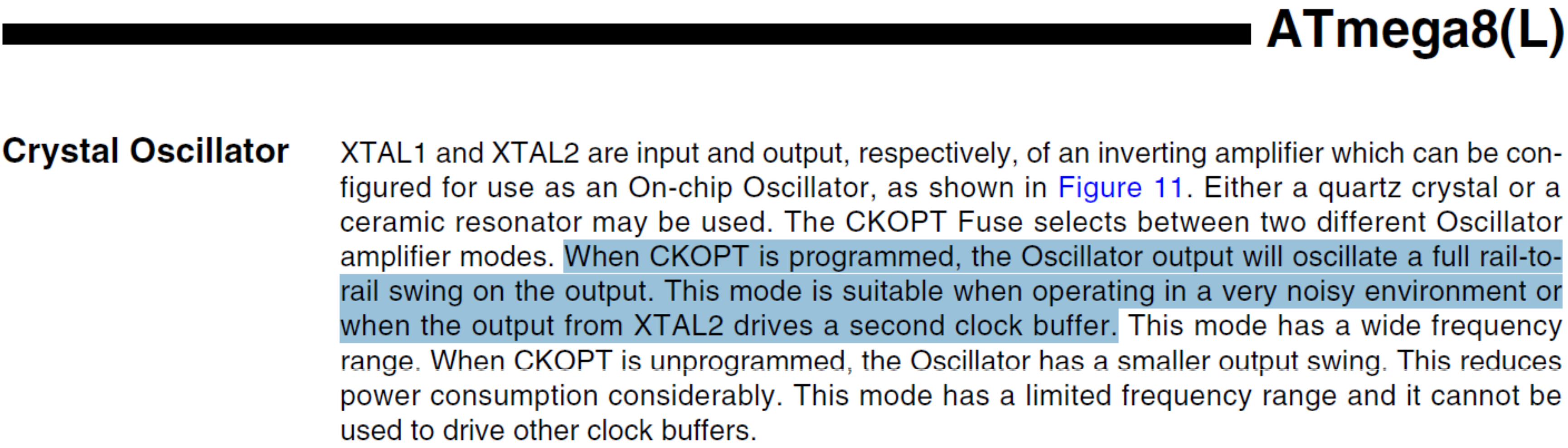



Fits Colorado Micro Devices USB2UART
JP1
4
3
2
1
R1
75R
C7
100u
C4
2u2
C3
100n
VCC
GND
GND
GND
R3
10k
VCC
IC1
1
PC6(/RESET)
9
PB6(XTAL1/TOSC1)
10
PB7(XTAL2/TOSC2)
7
VCC
20
AVCC
21
AREF
22
AGND
8
GND
ATMEGA8-PU
PC0(ADC0) 23
PC1(ADC1) 24
PC2(ADC2) 25
PC3(ADC3) 26
PC4(ADC4/SDA) 27
PC5(ADC5/SCL) 28
PD0(RXD) 2
PD1(TXD) 3
PD2(INT0) 4
PD3(INT1) 5
PD4(T0/XCK) 6
PD5(T1) 11
PD6(AIN0) 12
PD7(AIN1) 13
PB0(ICP1) 14
PB1(OC1A) 15
PB2(SS/OC1B) 16
PB3(MOSI/OC2) 17
PB4(MISO) 18
PB5(SCK) 19
TRIGGER
POWER_MEASUREMENT
CLOCKOUT
C5
22pF
C6
22pF
Q1
7.37MHz
2
1
GND
VCC
C2
VCC
C1
R2
75R
GND

Clock Buffer Note



Special Notes - XTAL



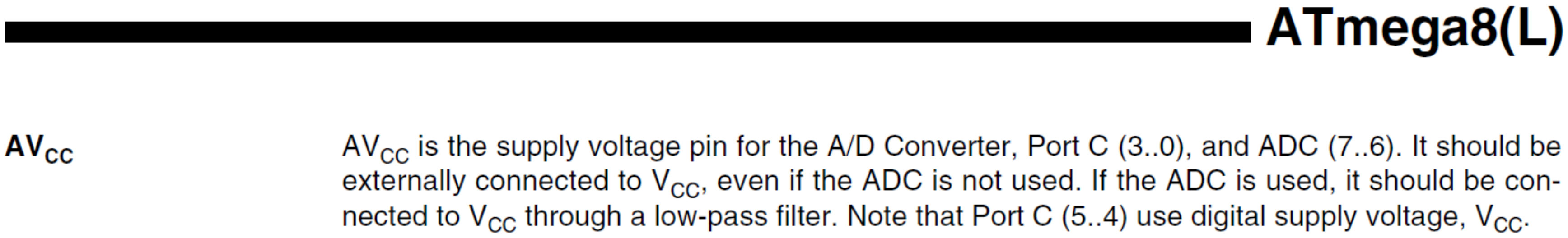
ATmega8(L)

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 11. Either a quartz crystal or a ceramic resonator may be used. The CKOPT Fuse selects between two different Oscillator amplifier modes. When CKOPT is programmed, the Oscillator output will oscillate a full rail-to-rail swing on the output. This mode is suitable when operating in a very noisy environment or when the output from XTAL2 drives a second clock buffer. This mode has a wide frequency range. When CKOPT is unprogrammed, the Oscillator has a smaller output swing. This reduces power consumption considerably. This mode has a limited frequency range and it cannot be used to drive other clock buffers.

- May get away without buffer… but probably need a buffer chip (any fast CMOS buffer IC should work)
- Better to use later chips (AtMega48/168/328) with explicit clock outputs
- For real systems WILL need to buffer clock, very easy to stop crystal oscillator
  - Keep add-on buffer physically close, minimize extra capacitive loading
  - Figure out which side of XTAL is connected to ‘output’

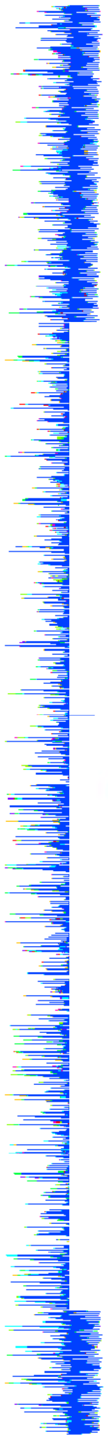
Special Notes – AVCC vs VCC

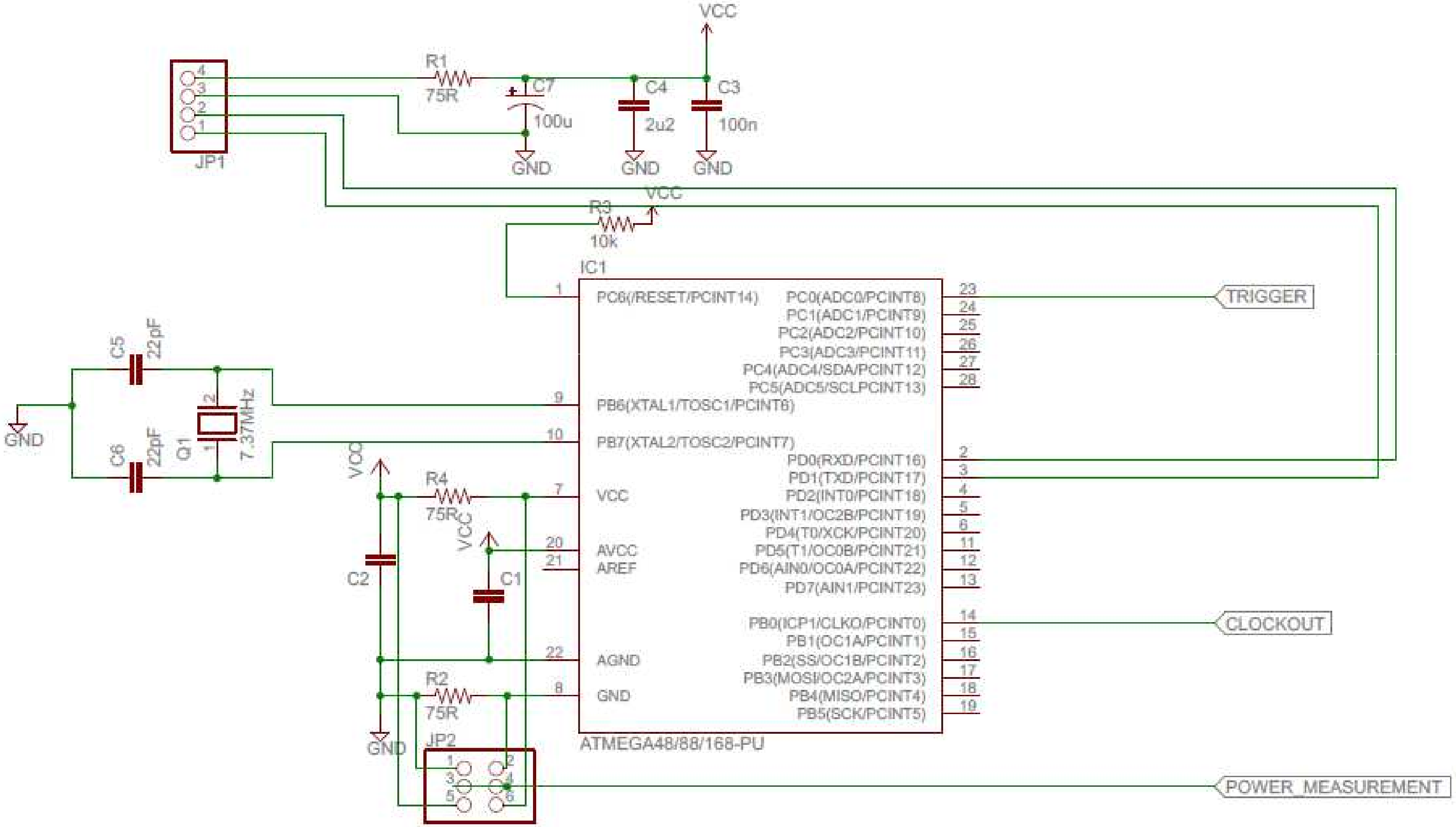


ATmega8(L)

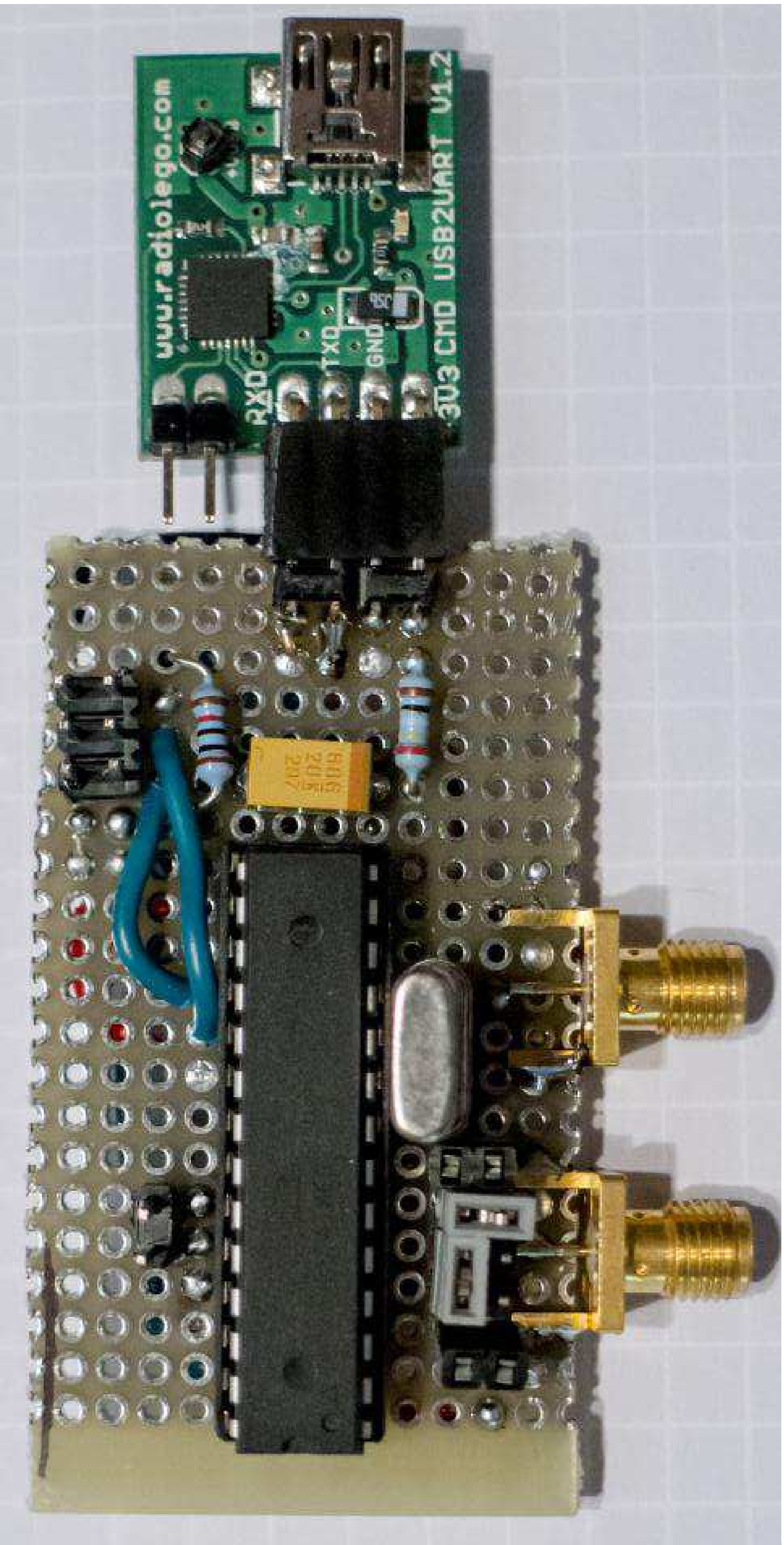
AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, Port C (3..0), and ADC (7..6). It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that Port C (5..4) use digital supply voltage, V_{CC}.

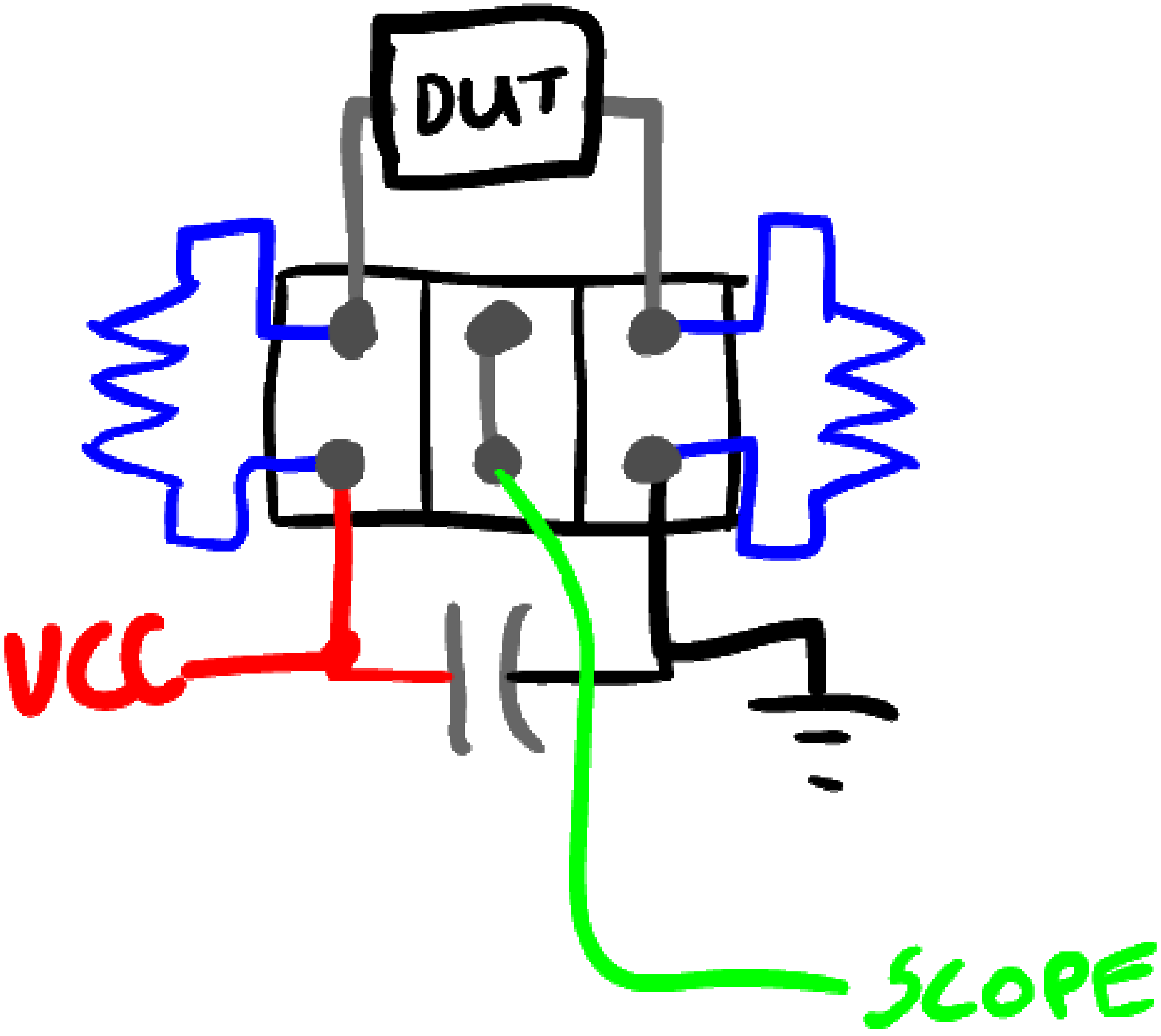


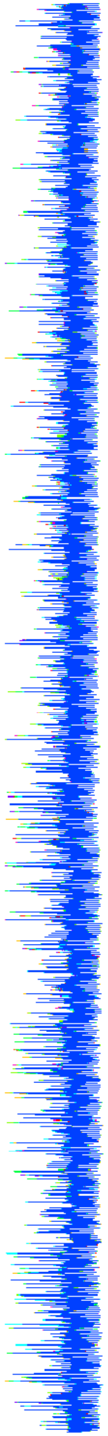


JP1
1
2
3
4
R1
75R
C7
100u
C4
2u2
C3
100n
VCC
GND
GND
GND
VCC
R3
10k
IC1
1
PC6(/RESET/PCINT14)
PC0(ADC0/PCINT8)
PC1(ADC1/PCINT9)
PC2(ADC2/PCINT10)
PC3(ADC3/PCINT11)
PC4(ADC4/SDA/PCINT12)
PC5(ADC5/SCLPCINT13)
23
24
25
26
27
28
TRIGGER
C5
22pF
C6
22pF
GND
Q1
7.37MHz
1
2
9
PB6(XTAL1/TOSC1/PCINT6)
10
PB7(XTAL2/TOSC2/PCINT7)
PD0(RXD/PCINT16)
PD1(TXD/PCINT17)
PD2(INT0/PCINT18)
PD3(INT1/OC2B/PCINT19)
PD4(T0/XCK/PCINT20)
PD5(T1/OC0B/PCINT21)
PD6(AIN0/OC0A/PCINT22)
PD7(AIN1/PCINT23)
2
3
4
5
6
11
12
13
VCC
R4
75R
7
VCC
VCC
20
AVCC
21
AREF
C2
C1
22
AGND
PB0(ICP1/CLKO/PCINT0)
PB1(OC1A/PCINT1)
PB2(SS/OC1B/PCINT2)
PB3(MOSI/OC2A/PCINT3)
PB4(MISO/PCINT4)
PB5(SCK/PCINT5)
14
15
16
17
18
19
CLOCKOUT
R2
75R
8
GND
GND
JP2
1
2
3
4
5
6
ATMEGA48/88/168-PU
POWER_MEASUREMENT

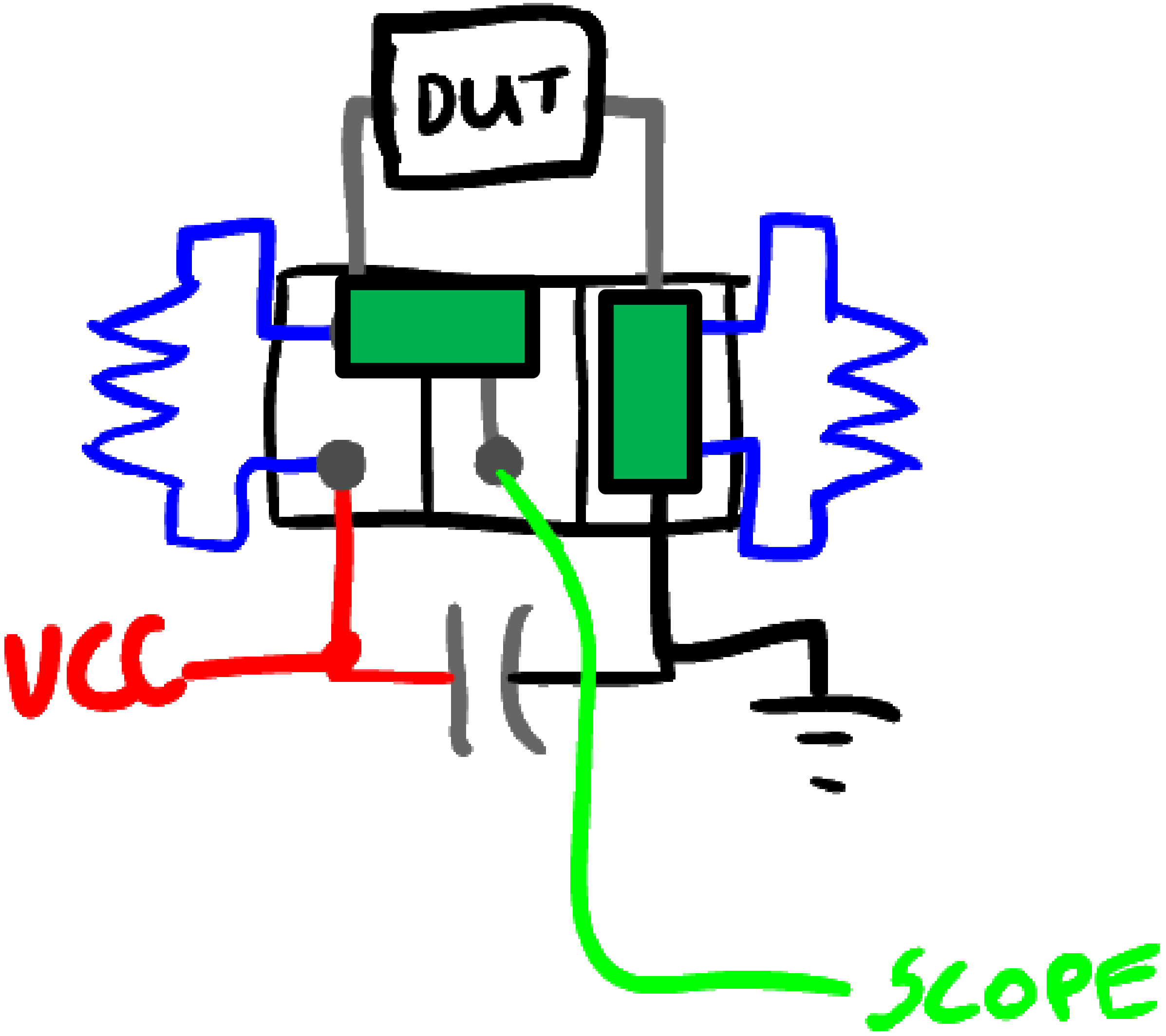

www.radiolego.com
3V3 CMD USB2UART V1.2
RXD
TXD
GND

6-pin DIP Header

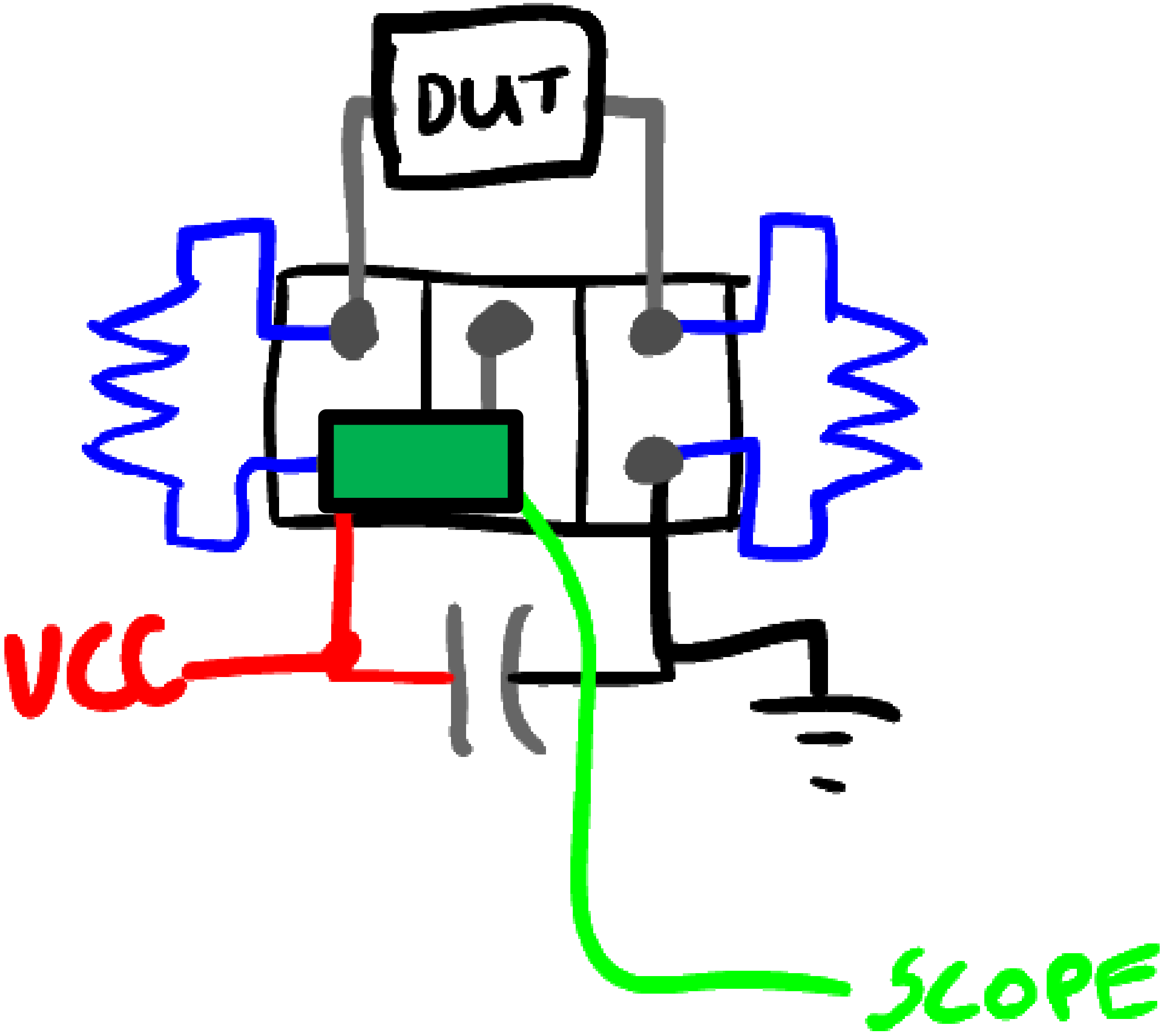


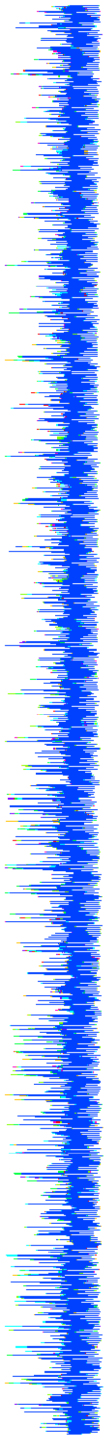


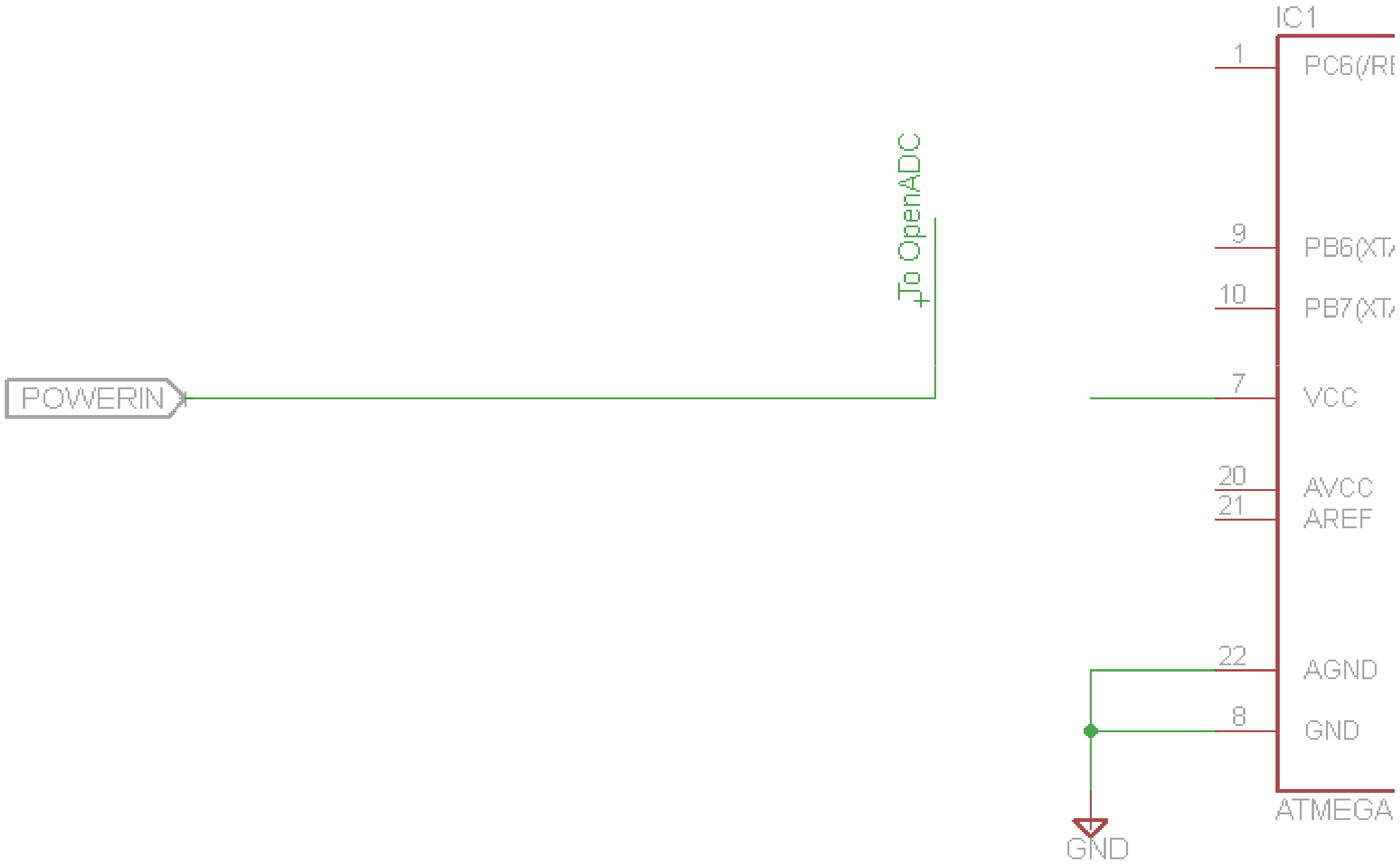
6-pin DIP Header



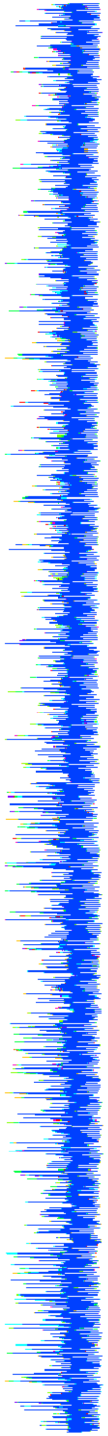
6-pin DIP Header

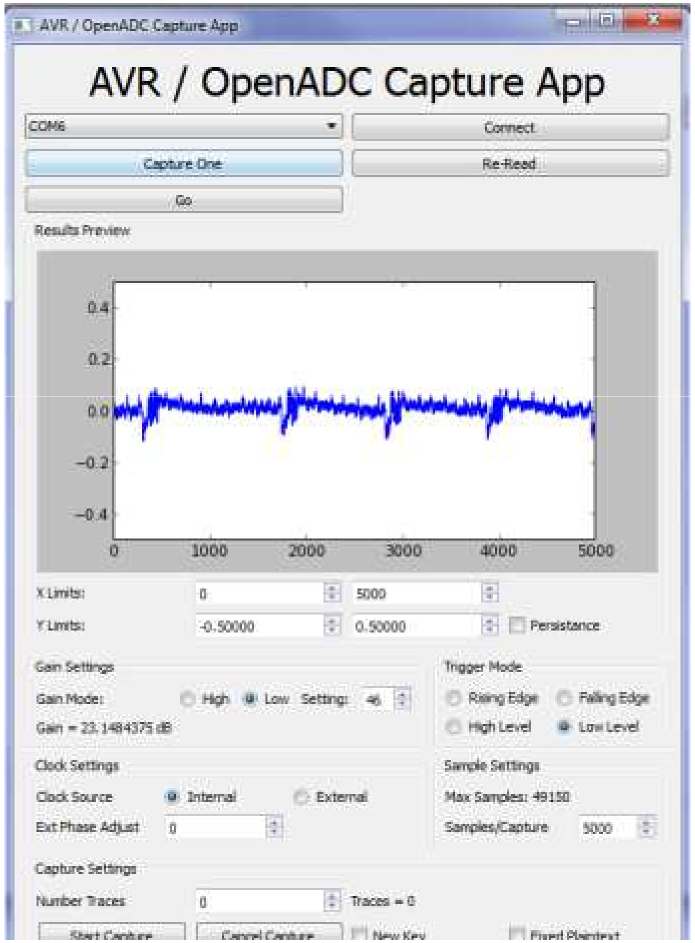






IC1
1
PC6(/RI
9
PB6(XT/
10
PB7(XT/
7
VCC
20
AVCC
21
AREF
22
AGND
8
GND
ATMEGA
GND
To OpenADC
POWERIN





AVR / OpenADC Capture App

AVR / OpenADC Capture App

COM6

Results Preview

X Limits: 0 5000

Y Limits: -0.50000 0.50000 □ Persistance

Gain Settings

Gain Mode: ○ High ● Low Setting: 46

Gain = 23.1484375 dB

Trigger Mode

○ Rising Edge ○ Falling Edge

○ High Level ● Low Level

Clock Settings

Clock Source ● Internal ○ External

Ext Phase Adjust 0

Sample Settings

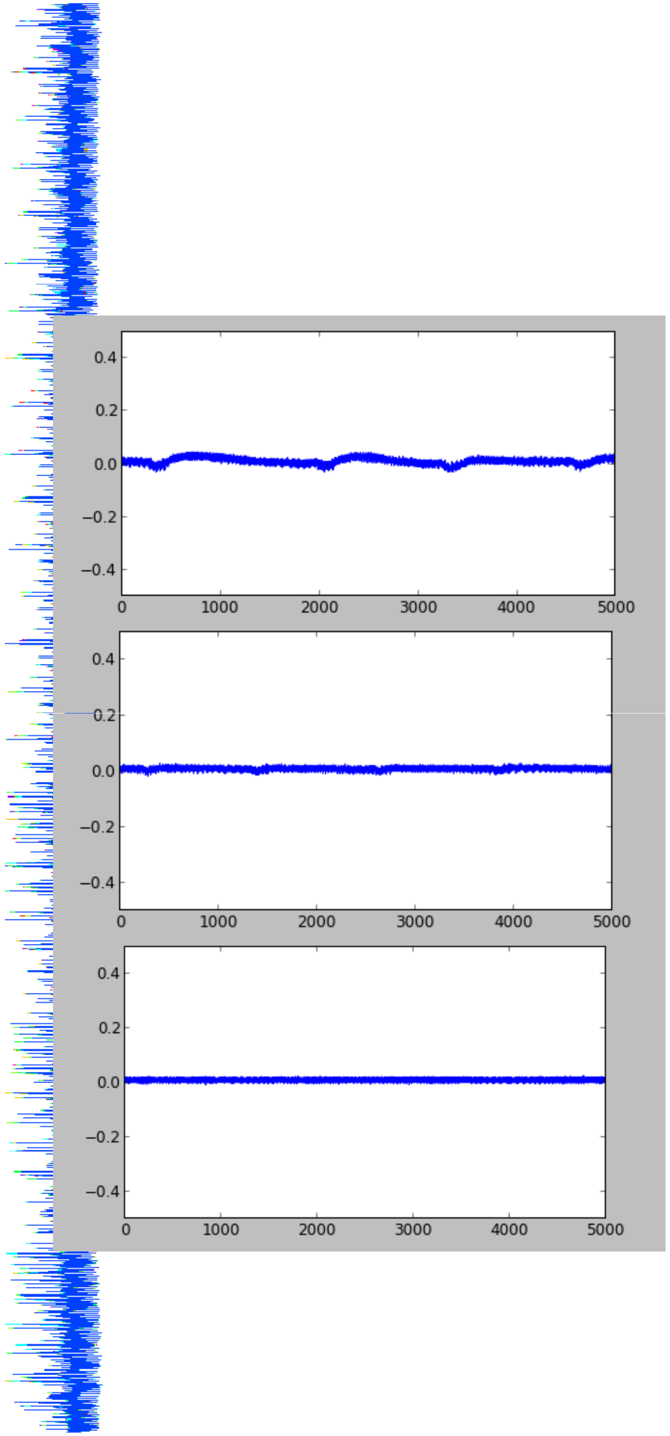
Max Samples: 49150

Samples/Capture 5000

Capture Settings

Number Traces 0 Traces = 0

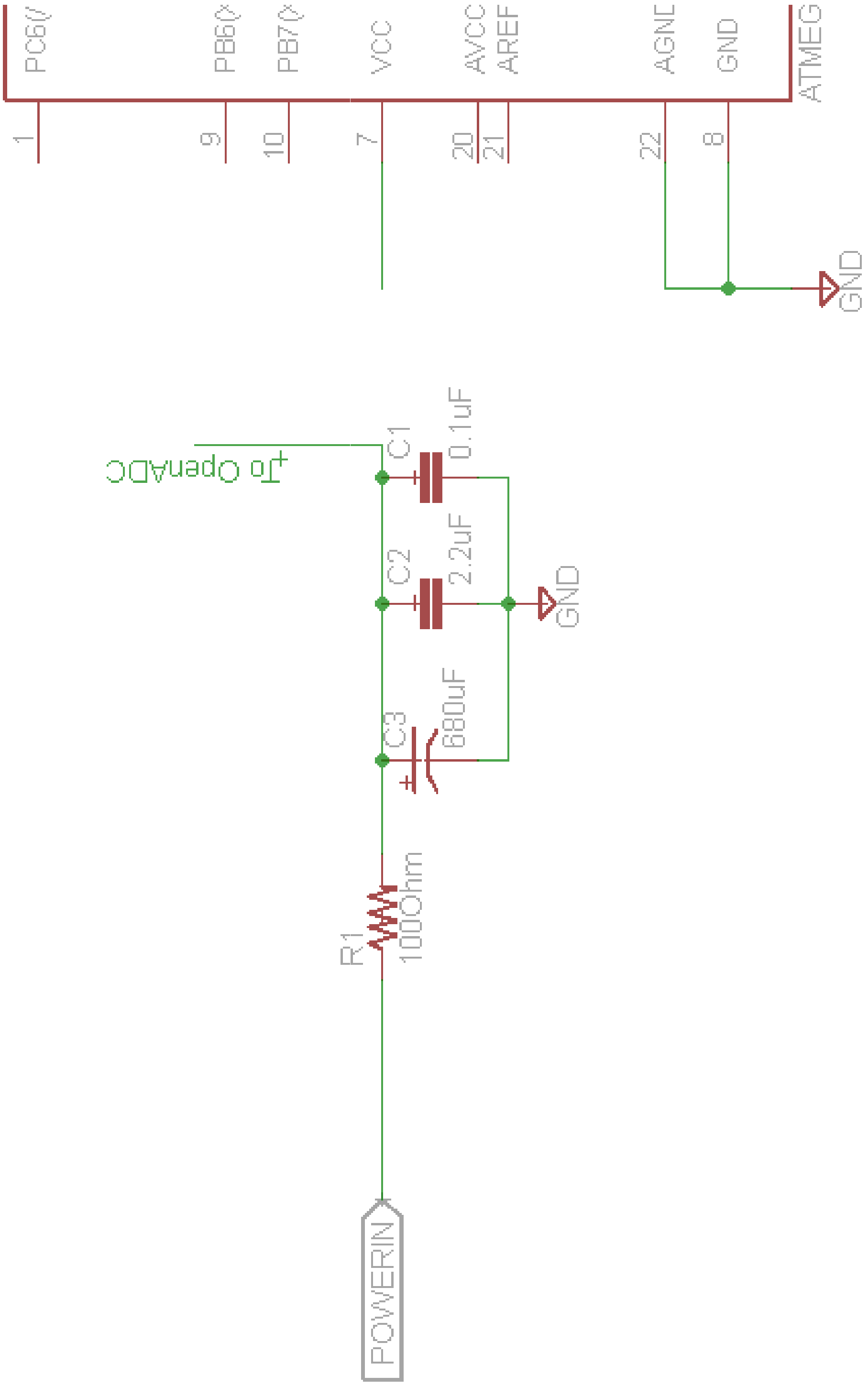
□ New Key □ Fixed Plaintext

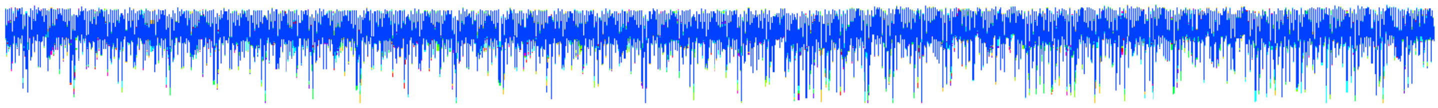


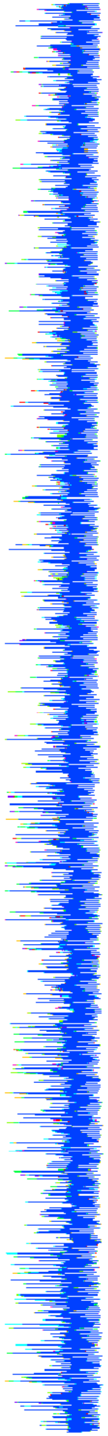
2.2uF Ceramic Capacitor

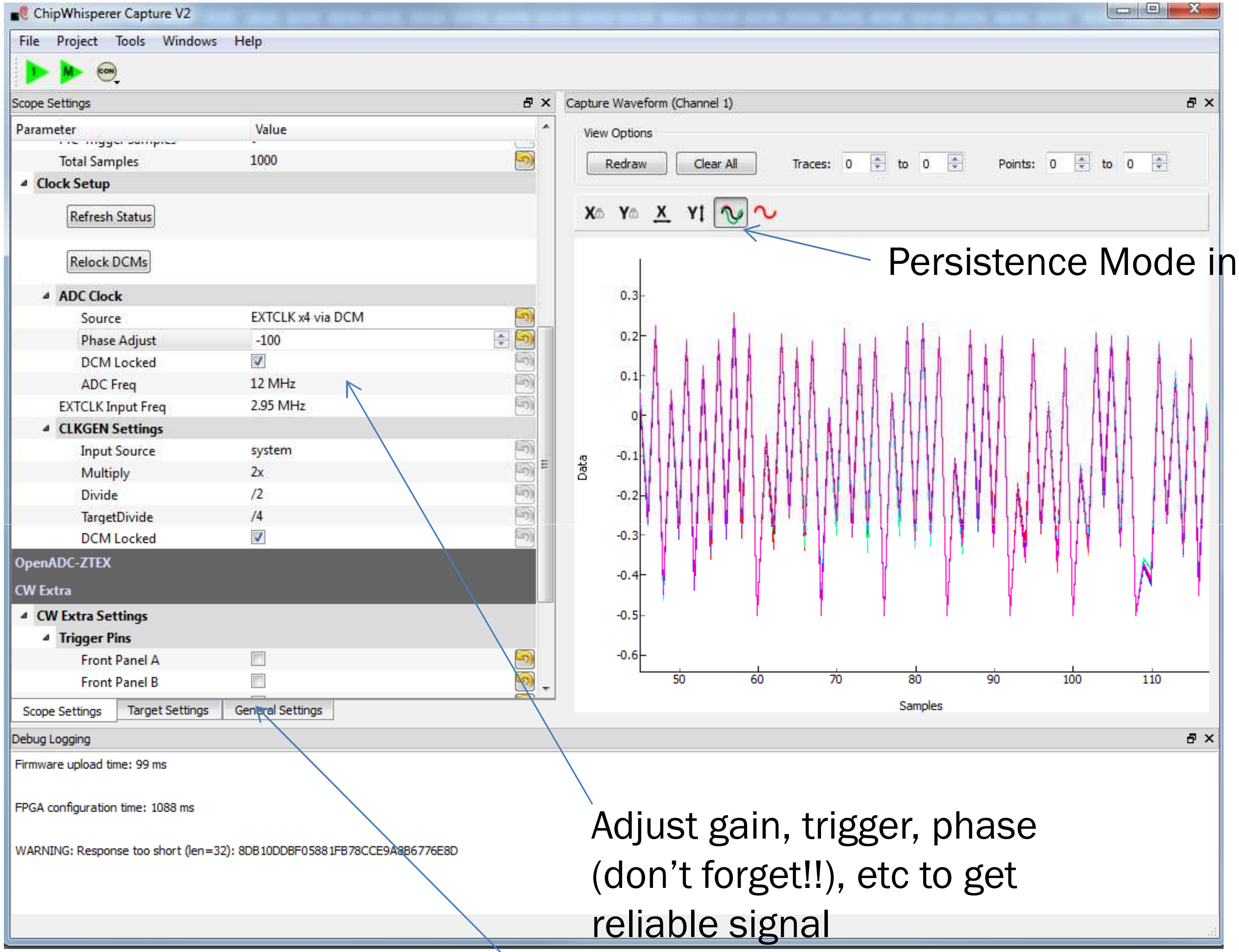
+680uF Electrolyctic

+100 ohm series resistor


PC6(/
PB6(>
PB7(>
VCC
AVCC
AREF
AGNI
GND
ATMEG
1
9
10
7
20
21
22
8
GND
To OpenADC
C1
0.1uF
C2
2.2uF
GND
C3
680uF
R1
100Ohm
POWERIN







Traces: 0 to 0   Points: 0 to 0

Parameter	Value
Total Samples	1000
Clock Setup	
ADC Clock	
Source	EXTCLK x4 via DCM
Phase Adjust	-100
DCM Locked	☑
ADC Freq	12 MHz
EXTCLK Input Freq	2.95 MHz
CLKGEN Settings	
Input Source	system
Multiply	2x
Divide	/2
TargetDivide	/4
DCM Locked	☑
CW Extra Settings	
Trigger Pins	
Front Panel A	□
Front Panel B	□

```
Firmware upload time: 99 ms

FPGA configuration time: 1088 ms

WARNING: Response too short (len=32): 8DB10DDBF05881FB78CCE9A8B6776E8D
```

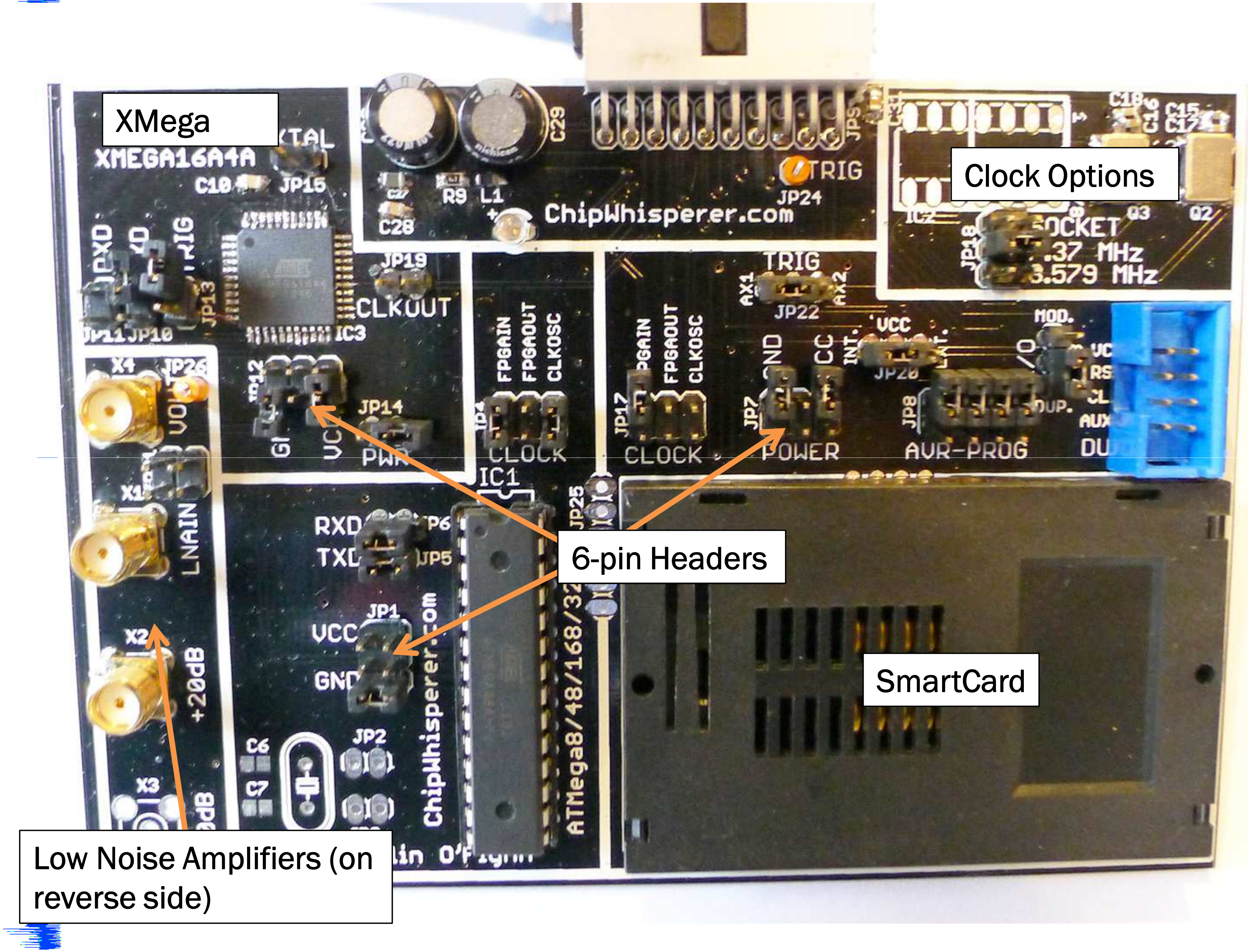
Persistence Mode in Scope

Adjust gain, trigger, phase (don’t forget!!), etc to get reliable signal

Set fixed plaintext on “General Settings” tab

Problems with Synchronous Sampling

- Cannot intuitively ‘see’ noise like with normal scope
  - Consider using normal scope for initial setup if available
  - Validation of noise-free environment is CRITICAL with synchronous sampling



XMega

Clock Options

6-pin Headers

SmartCard

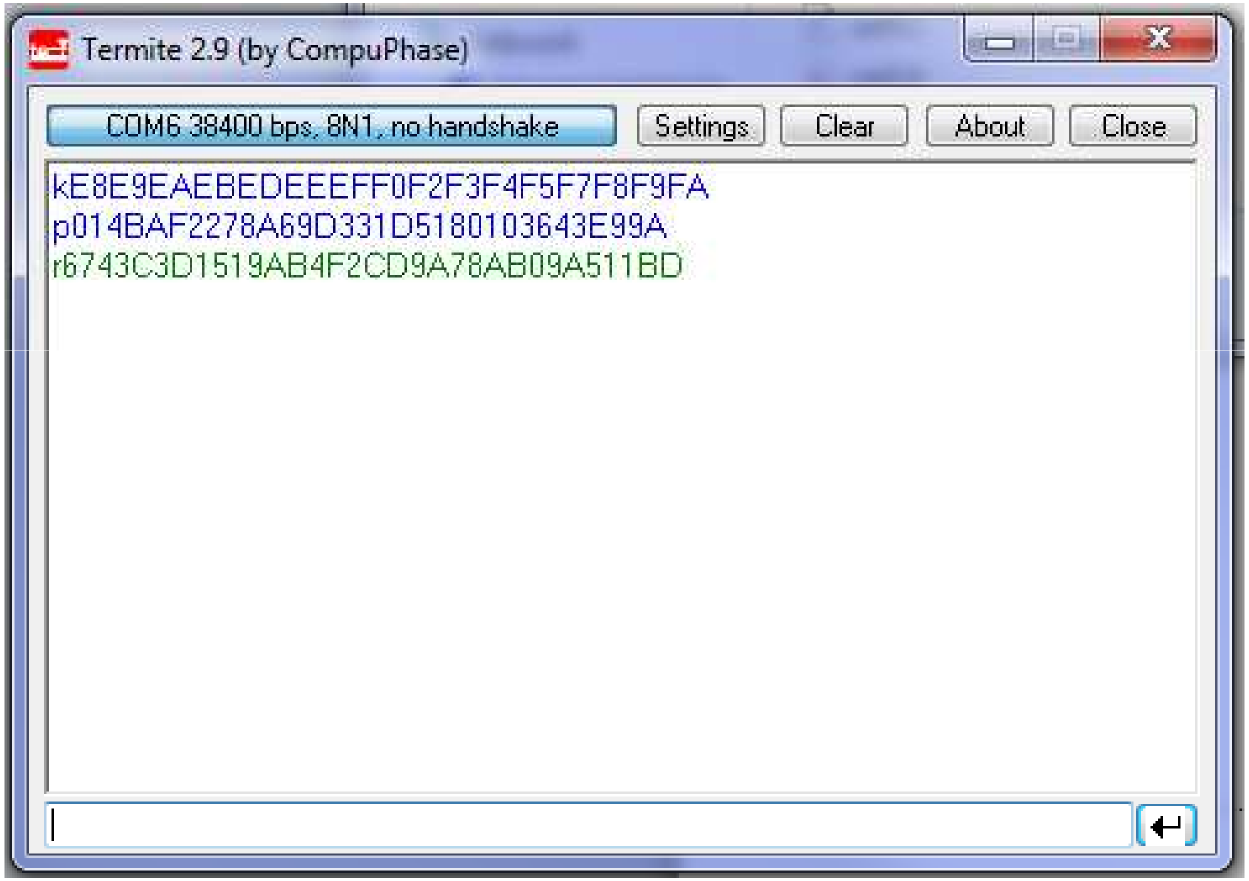
Low Noise Amplifiers (on reverse side)

Multi-Target Victim Features

- Supports AVR, Xmega, SmartCard targets
- Supports AVR/XMega programmer built into ChipWhisperer Rev2 (including MegaCard/FunCard SmartCards)
- Supports external SmartCard reader in pass-through or modify mode
- LNA on-board for using H-Field probe

SIMPLE SERIAL – EXAMPLE CAPTURE

SimpleSerial Protocol


Termite 2.9 (by CompuPhase)
COM6 38400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
kE8E9EAEBEDEEEFF0F2F3F4F5F7F8F9FA
p014BAF2278A69D331D5180103643E99A
r6743C3D1519AB4F2CD9A78AB09A511BD
```

SimpleSerial Protocol

Set key:
k00112233445566778899AABBCCDDEEFF\n

Encrypt with software AES:
p00112233445566778899AABBCCDDEEFF\n

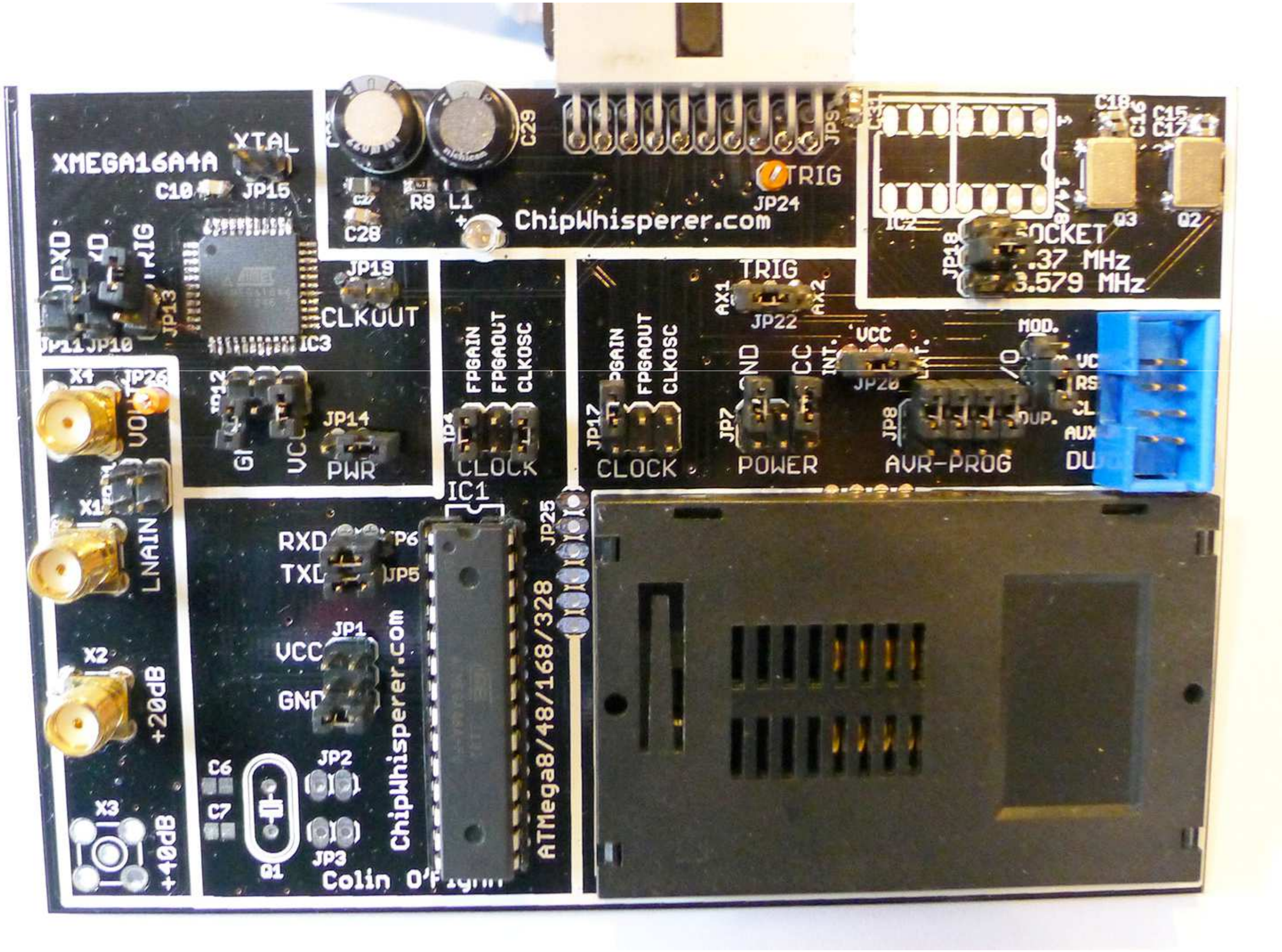
Encrypt with hardware AES (Xmega target only):
h00112233445566778899AABBCCDDEEFF\n

Encryption Response:
r00112233445566778899AABBCCDDEEFF\n

Serial Port = 38400, 8N1

Building the Hardware

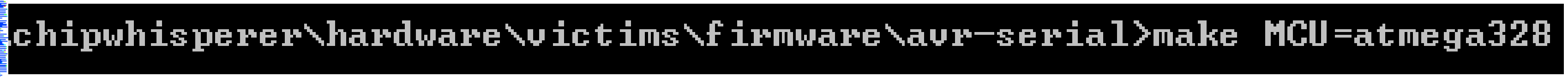
Using Multi-Target Board



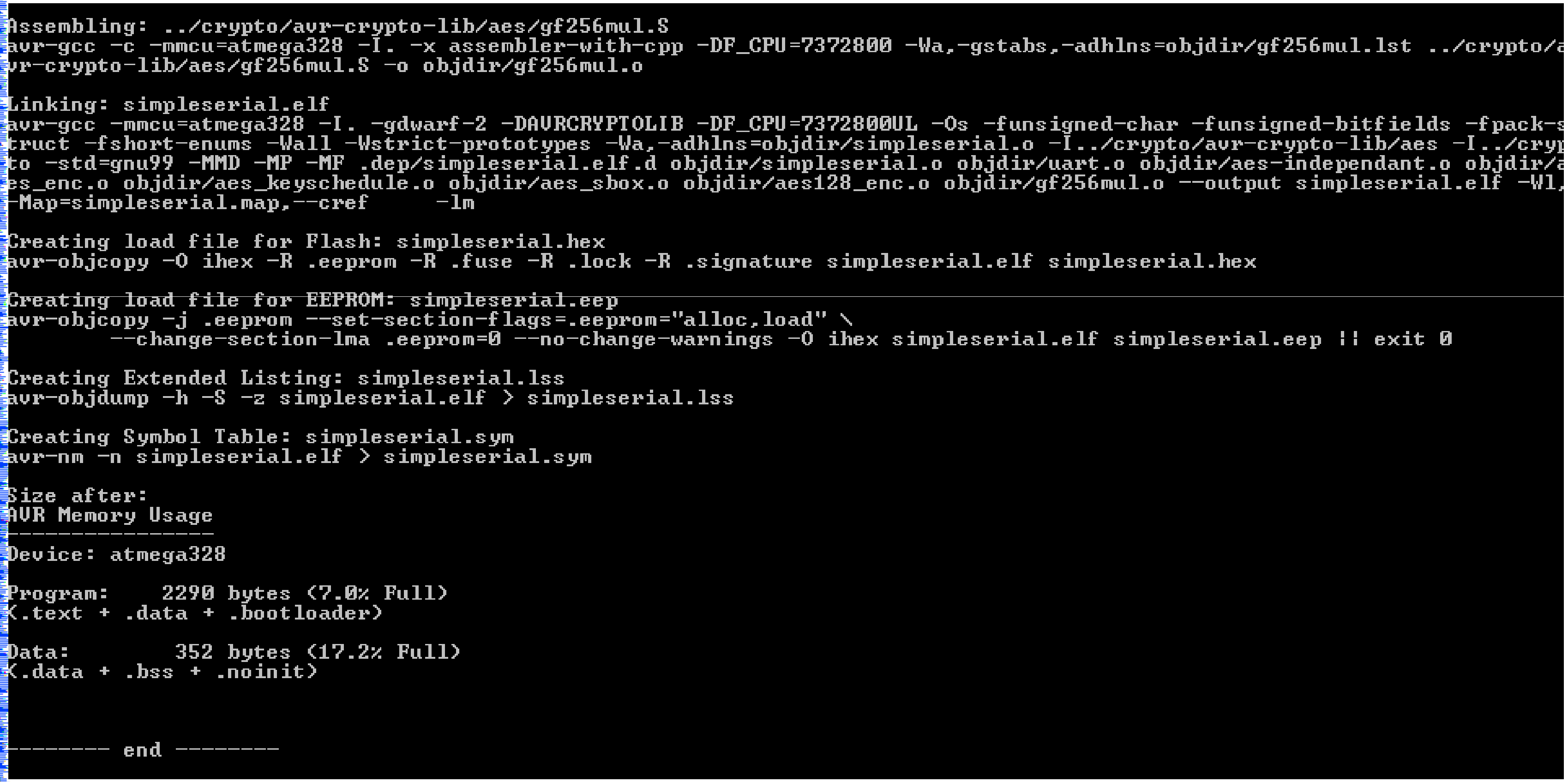
Building the Firmware

1. Get WinAVR on Windows, AVR Toolchain on Linux
2. Checkout GIT Repository
3. Copy avr-crypto-lib into appropriate place
4. Run ‘`make MCU=atmega328p`’ at `chipwhisperer\hardware\victims\firmware\avr-serial`

Building the Firmware



```
chipwhisperer\hardware\victims\firmware\avr-serial>make MCU=atmega328
```



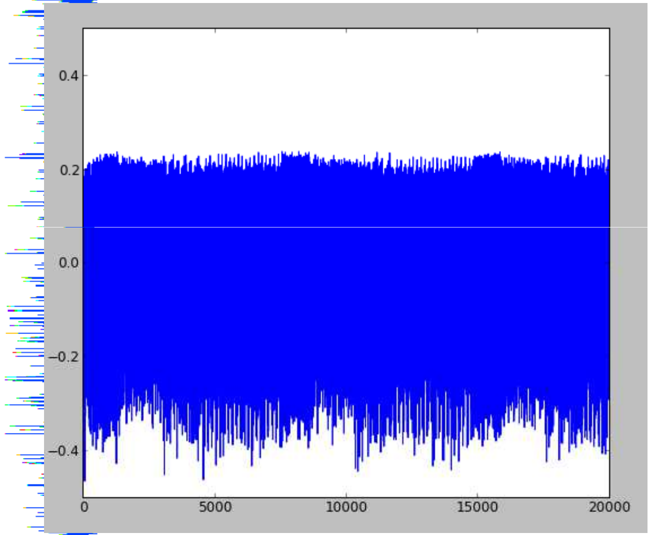
```
Assembling: ../crypto/avr-crypto-lib/aes/gf256mul.S
avr-gcc -c -mmcu=atmega328 -I. -x assembler-with-cpp -DF_CPU=7372800 -Wa,-gstabs,-adhlns=objdir/gf256mul.lst ../crypto/a
vr-crypto-lib/aes/gf256mul.S -o objdir/gf256mul.o

Linking: simpleserial.elf
avr-gcc -mmcu=atmega328 -I. -gdwarf-2 -DAVRCRYPTOLIB -DF_CPU=7372800UL -Os -funsigned-char -funsigned-bitfields -fpack-s
truct -fshort-enums -Wall -Wstrict-prototypes -Wa,-adhlns=objdir/simpleserial.o -I../crypto/avr-crypto-lib/aes -I../cryp
to -std=gnu99 -MMD -MP -MF .dep/simpleserial.elf.d objdir/simpleserial.o objdir/uart.o objdir/aes-independant.o objdir/a
es_enc.o objdir/aes_keyschedule.o objdir/aes_sbox.o objdir/aes128_enc.o objdir/gf256mul.o --output simpleserial.elf -Wl,
-Map=simpleserial.map,--cref     -lm

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: atmega328

Program:    2290 bytes (7.0% Full)
(.text + .data + .bootloader)

Data:        352 bytes (17.2% Full)
(.data + .bss + .noinit)



-------- end --------
```

Selection of Crypto in Use

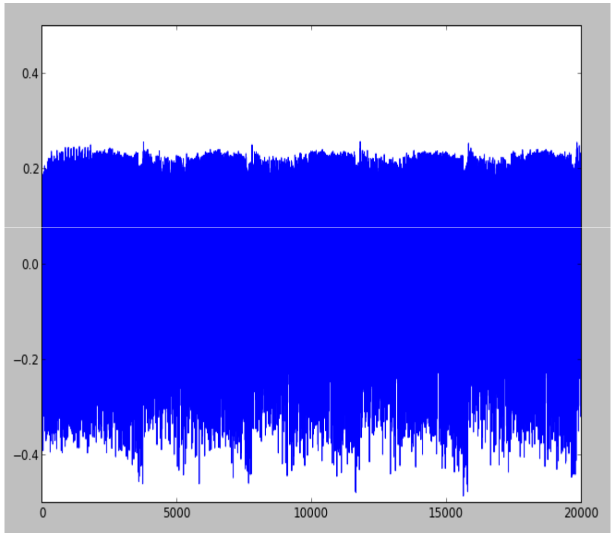
1. Edit ‘Makefile’ to select Crypto module
    1. NB: Crypto libs NOT included in distribution yet

Selection of Crypto in Use

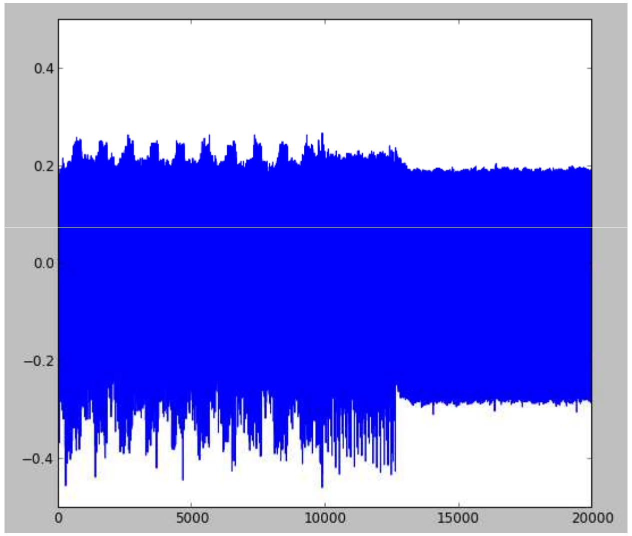
avr-crypto-lib in C



Straightforward C



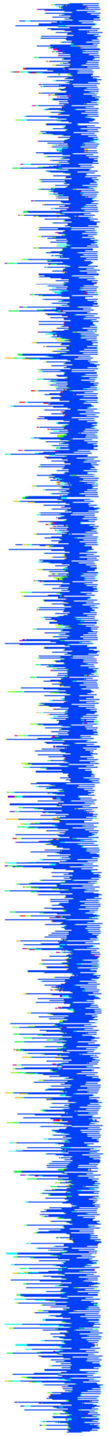
avr-crypto-lib in ASM



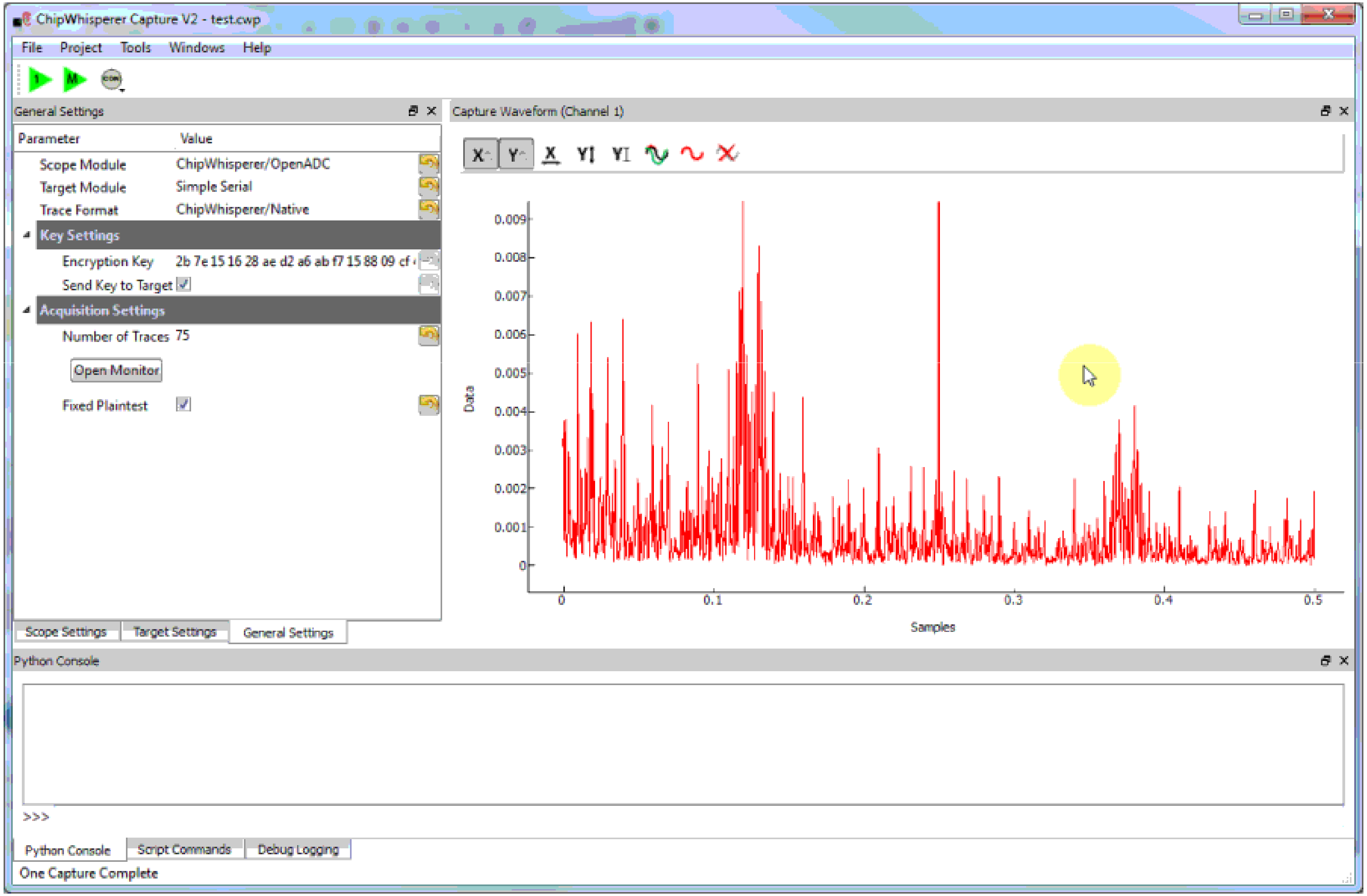
Programming the Target

Validating

Capturing Waveform - Normal



Capturing Waveform - Frequency



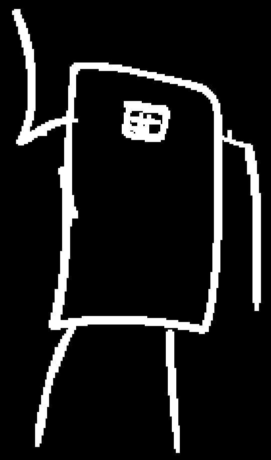
Parameter	Value
Scope Module	ChipWhisperer/OpenADC
Target Module	Simple Serial
Trace Format	ChipWhisperer/Native
Key Settings	
Encryption Key	2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf
Send Key to Target	☑
Acquisition Settings	
Number of Traces	75
Fixed Plaintext	☑

```
>>>
```

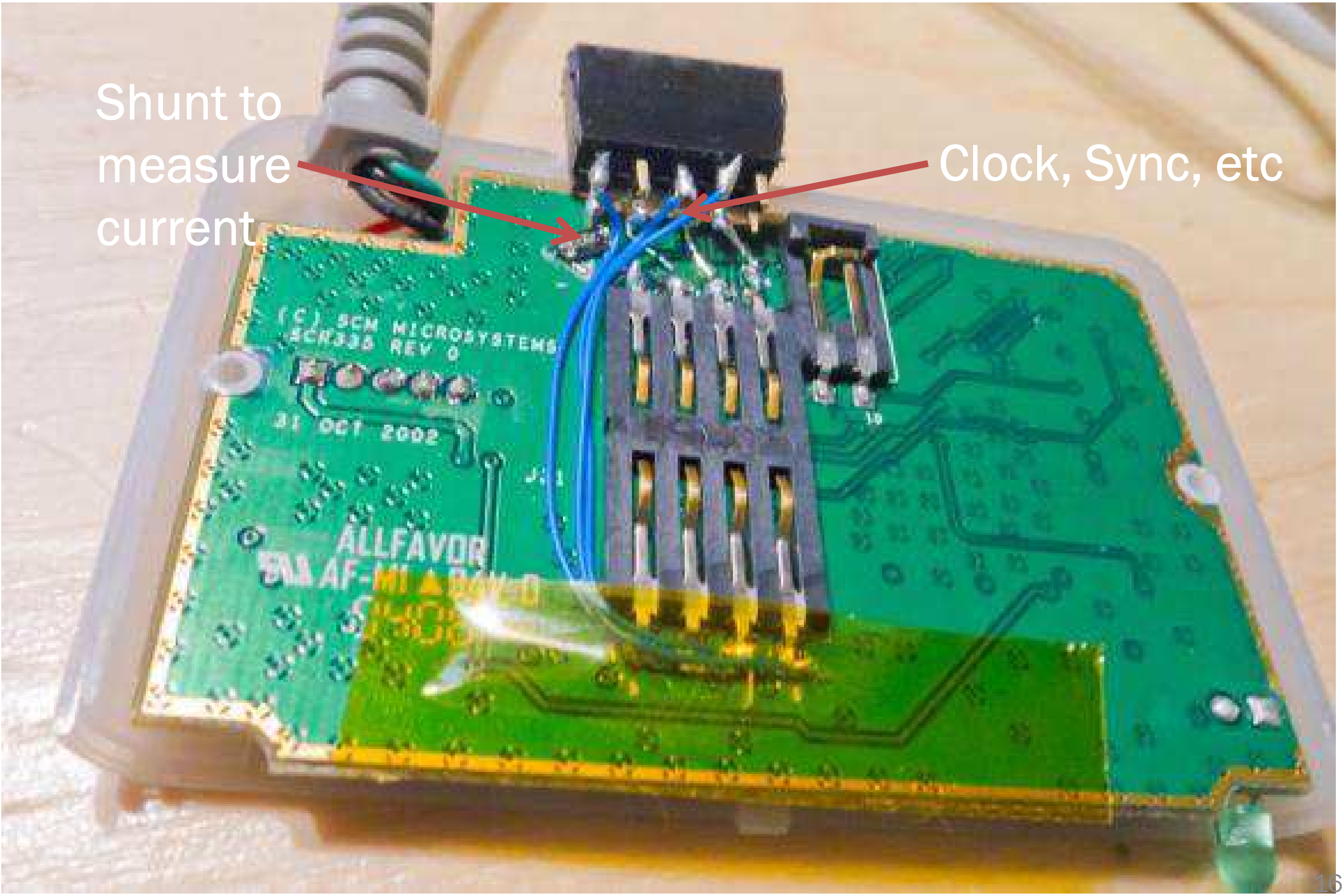
One Capture Complete

Capturing Waveform – Diff Probe

SMART CARDS



Smart Card Reader

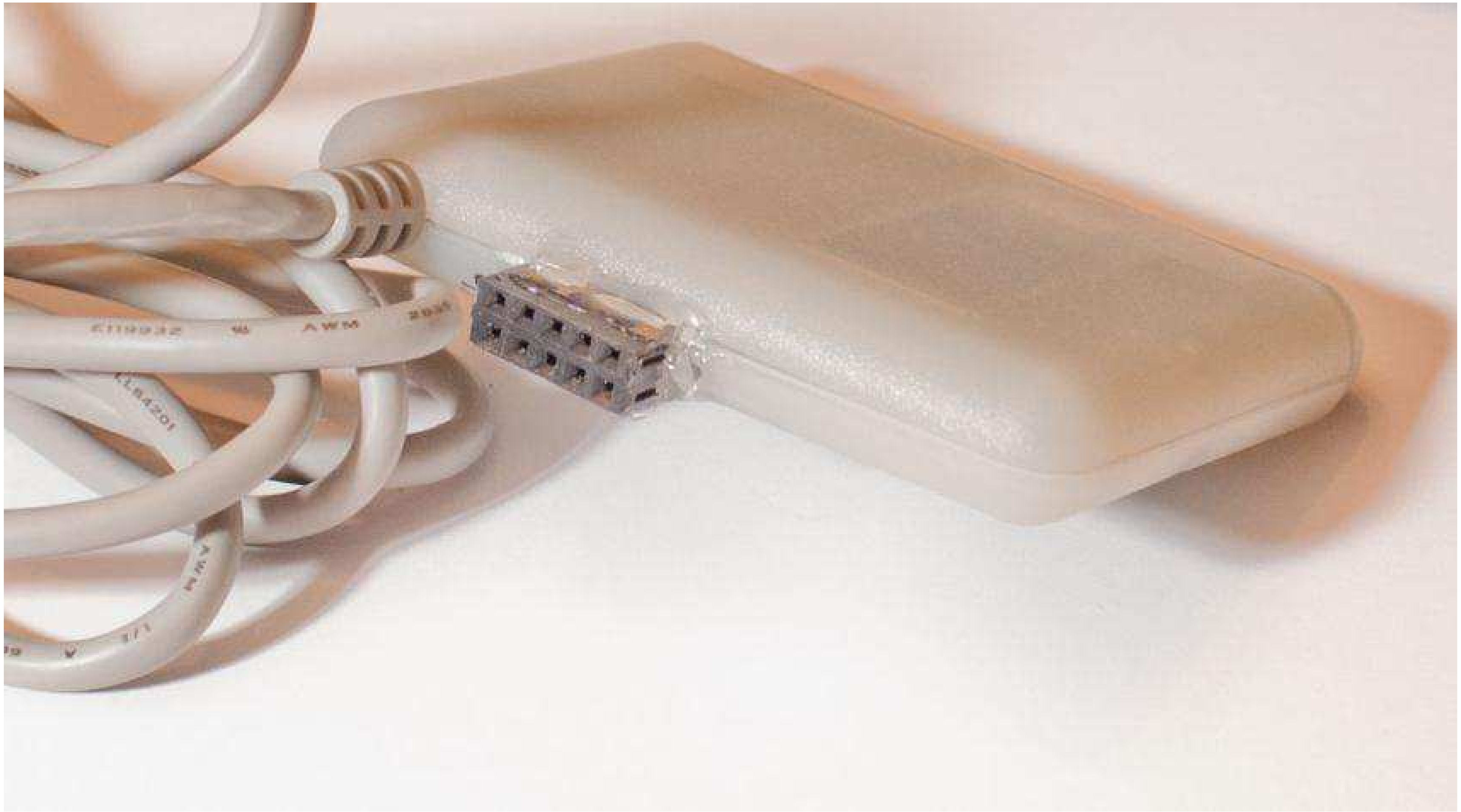


Shunt to measure current

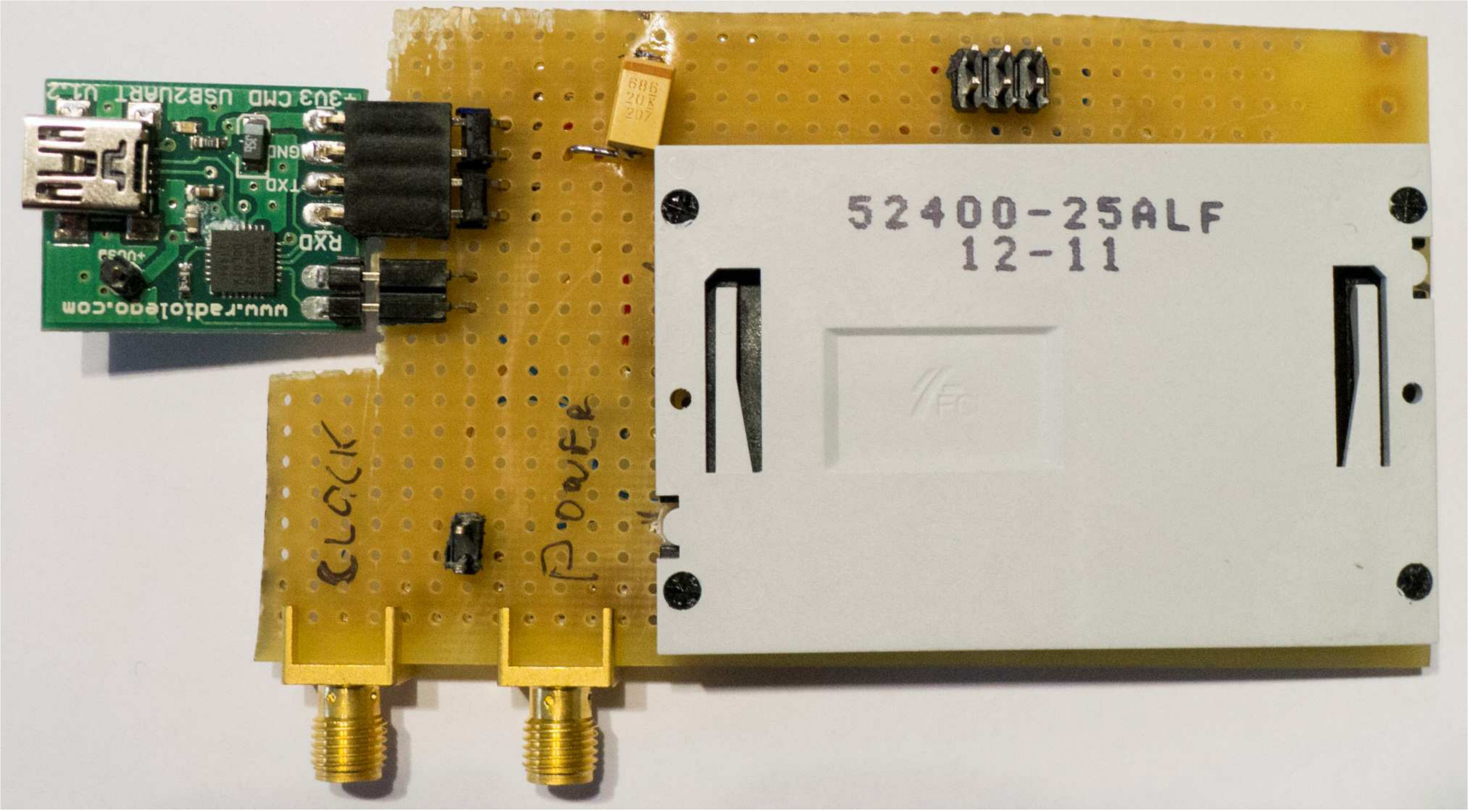
Clock, Sync, etc

SMART CARD READER

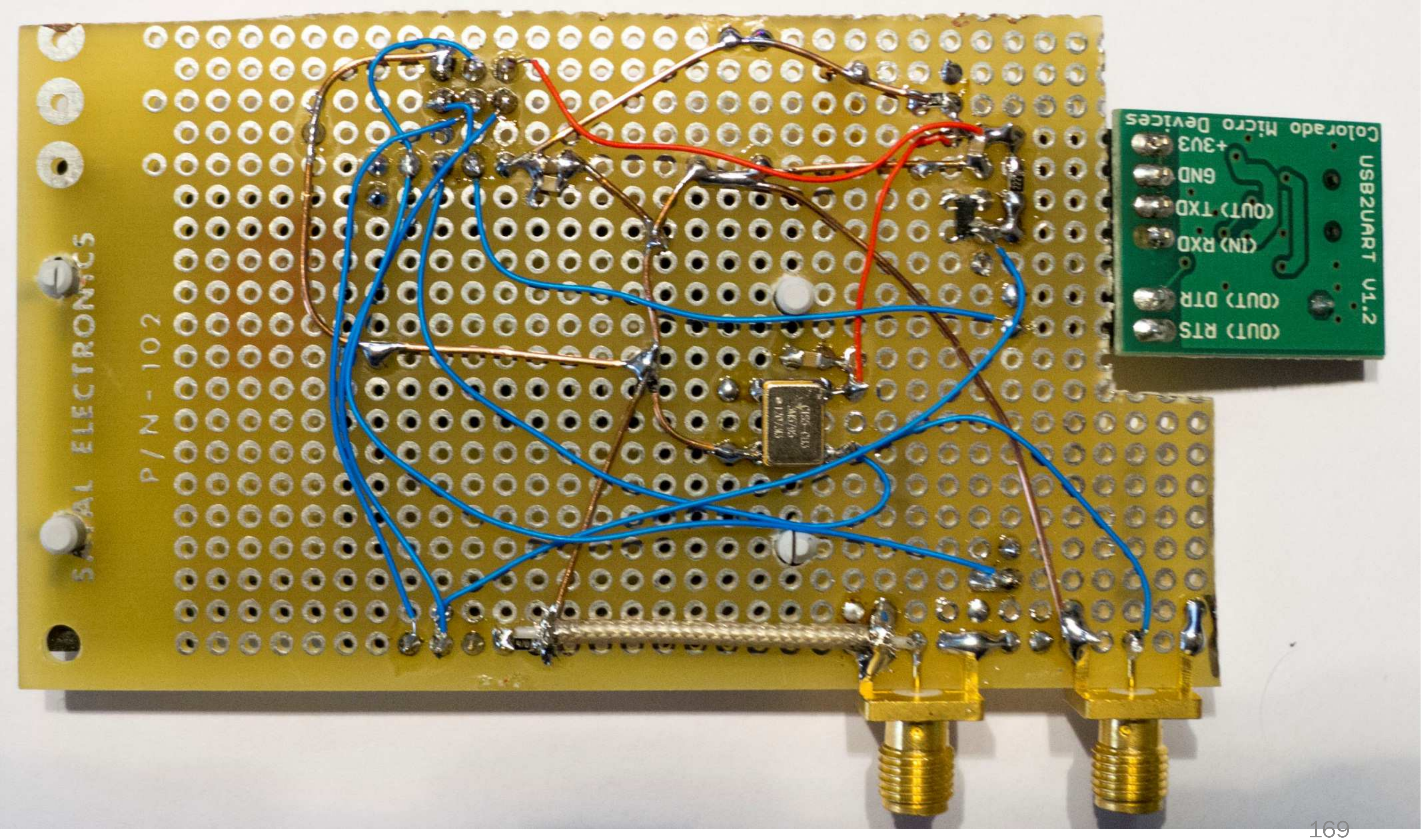
SMART CARD READER



CHEAPER READERS



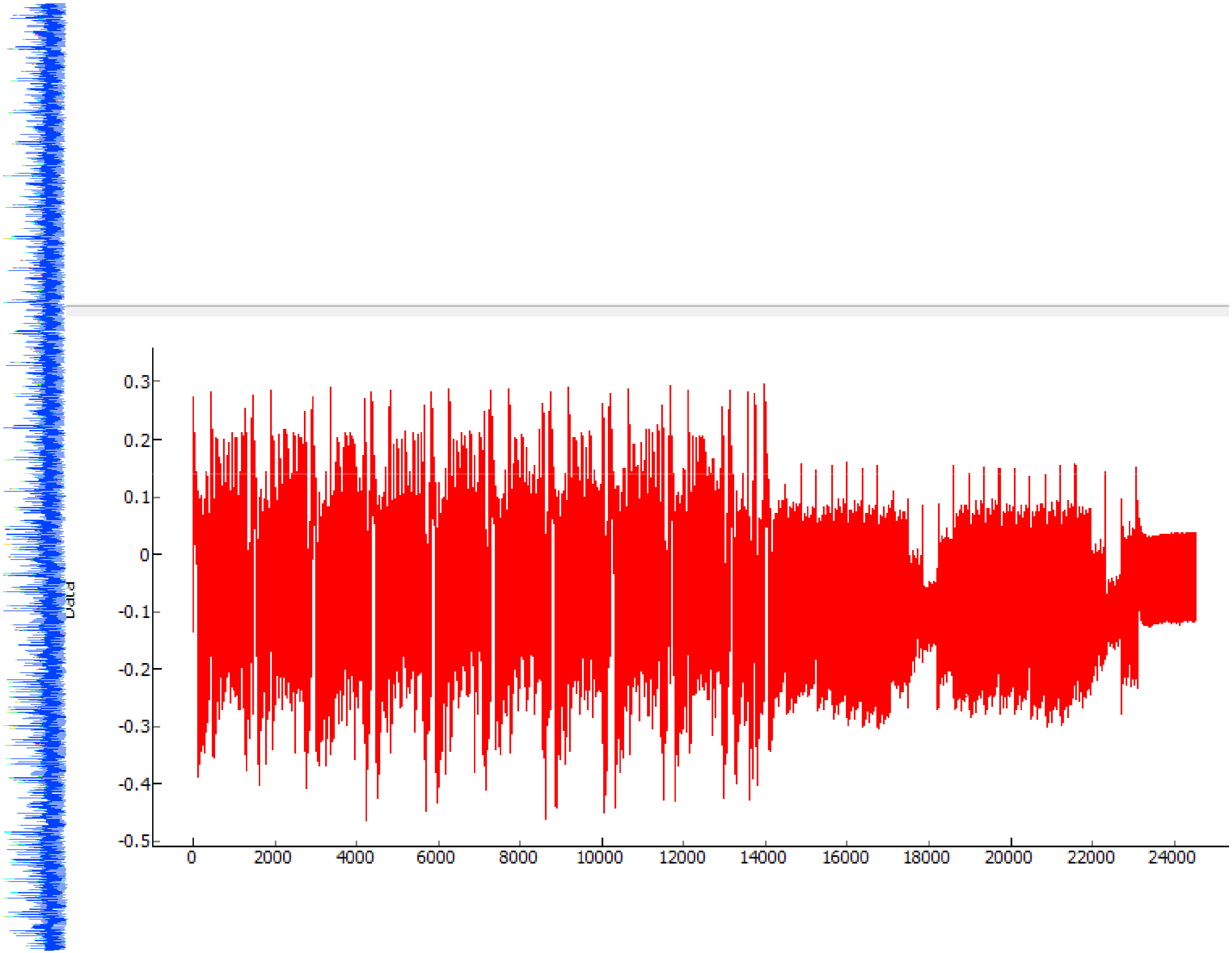
CHEAPER READERS



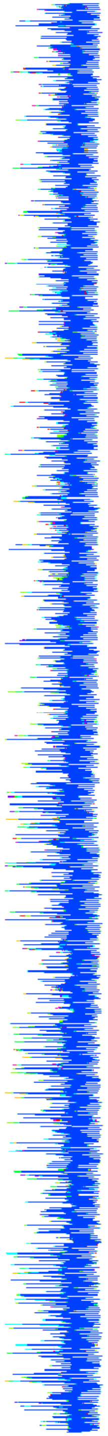
SMARTCARD – EXAMPLE CAPTURE

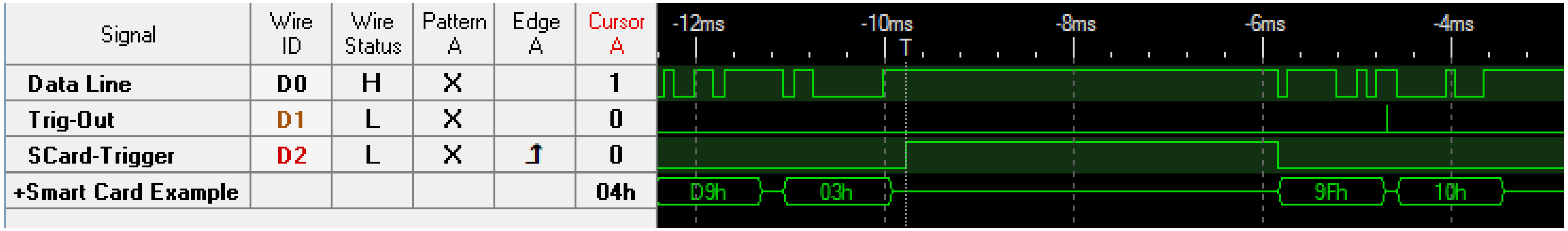
SmartCard Example Capture

- Note: DCM input frequency should be >= 5MHz
  - SmartCard clock = 3.58 MHz
  - DCM Lock may fail (especially on CW-Rev2 due to clock routing making matters worse, works on SASEBO-W)
  - Instead use 7.37 MHz clock, change baud to 19763
  - …Or use “EXTCLK Direct”, although some bug with external trigger (works OK with advanced trigger)

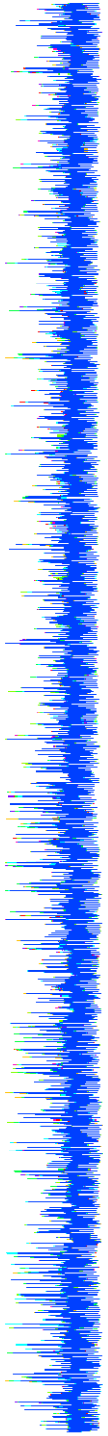

Data
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
22000
24000

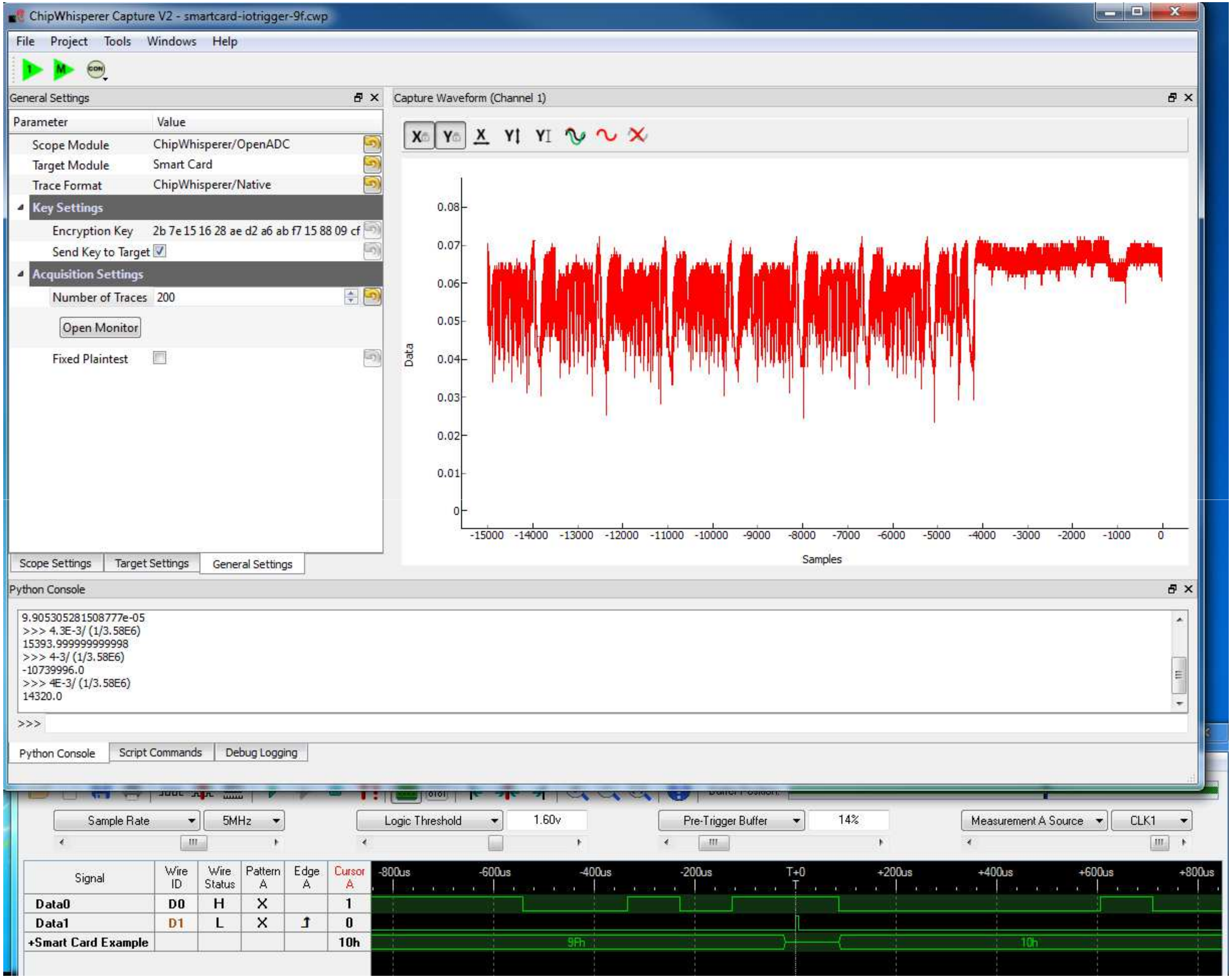
SMARTCARD – ADVANCED IO TRIGGER





Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data Line	D0	H	X		1
Trig-Out	D1	L	X		0
SCard-Trigger	D2	L	X	↑	0
+Smart Card Example					04h





ChipWhisperer Capture V2 - smartcard-iotrigger-9f.cwp

Parameter	Value
Scope Module	ChipWhisperer/OpenADC
Target Module	Smart Card
Trace Format	ChipWhisperer/Native
Key Settings	
Encryption Key	2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf
Send Key to Target	☑
Acquisition Settings	
Number of Traces	200
Fixed Plaintest	□

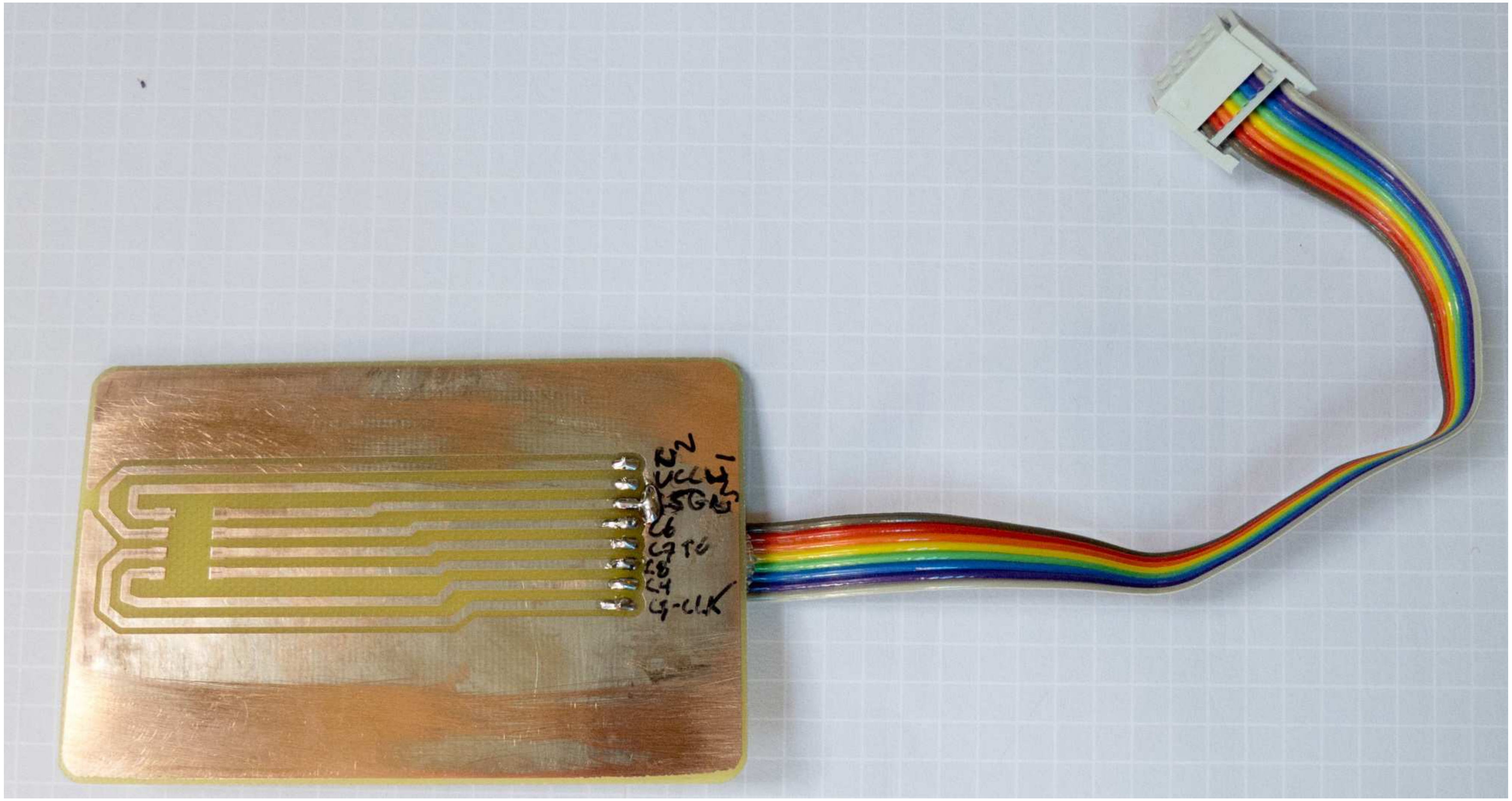
```
9.905305281508777e-05
>>> 4.3E-3/ (1/3.58E6)
15393.999999999998
>>> 4-3/ (1/3.58E6)
-10739996.0
>>> 4E-3/ (1/3.58E6)
14320.0
>>>
```

Sample Rate: 5MHz
Logic Threshold: 1.60v
Pre-Trigger Buffer: 14%
Measurement A Source: CLK1

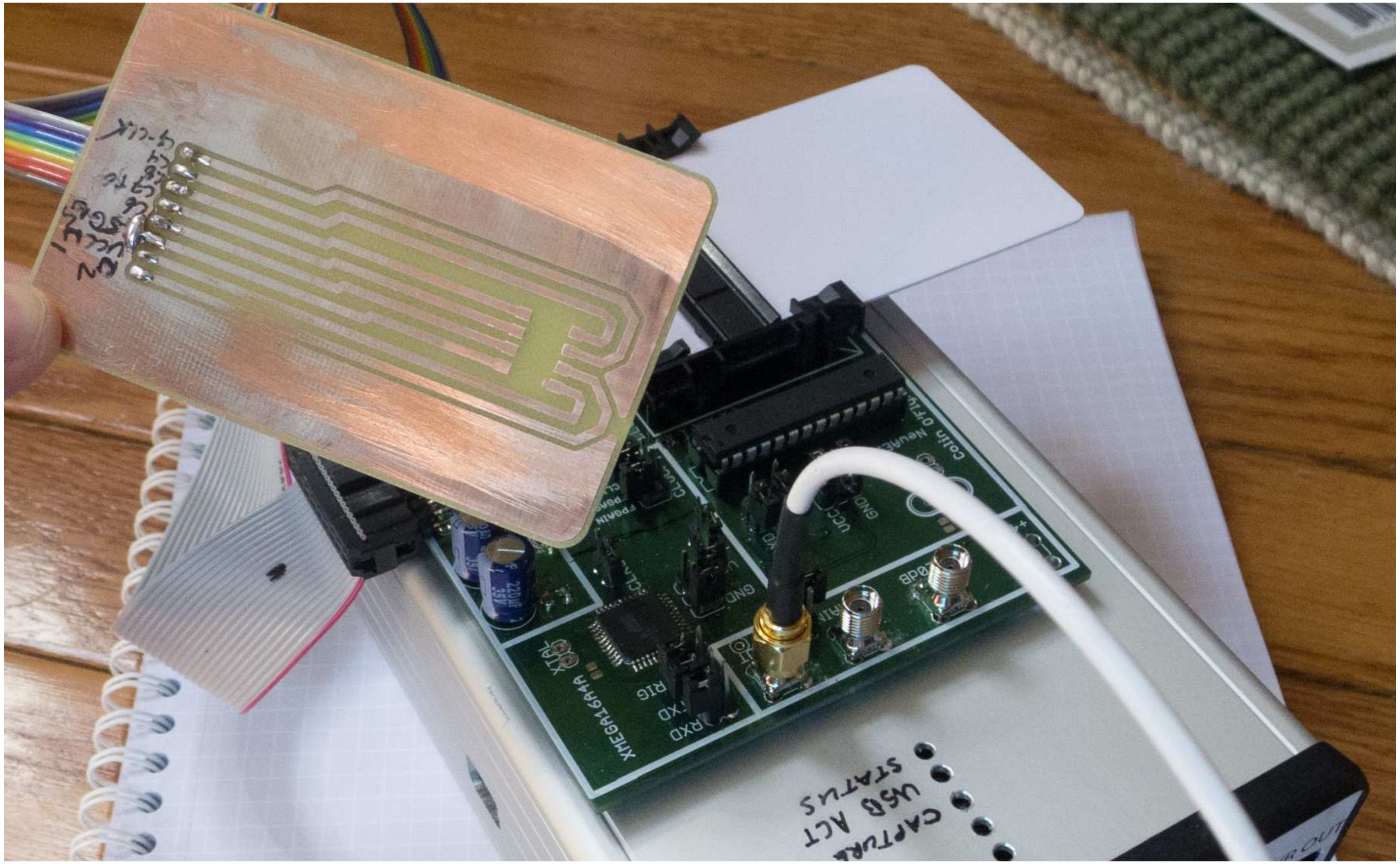
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data0	D0	H	X		1
Data1	D1	L	X	↥	0
+Smart Card Example					10h

SMARTCARD – FEED THRU

CWR2: IN-THE-MIDDLE

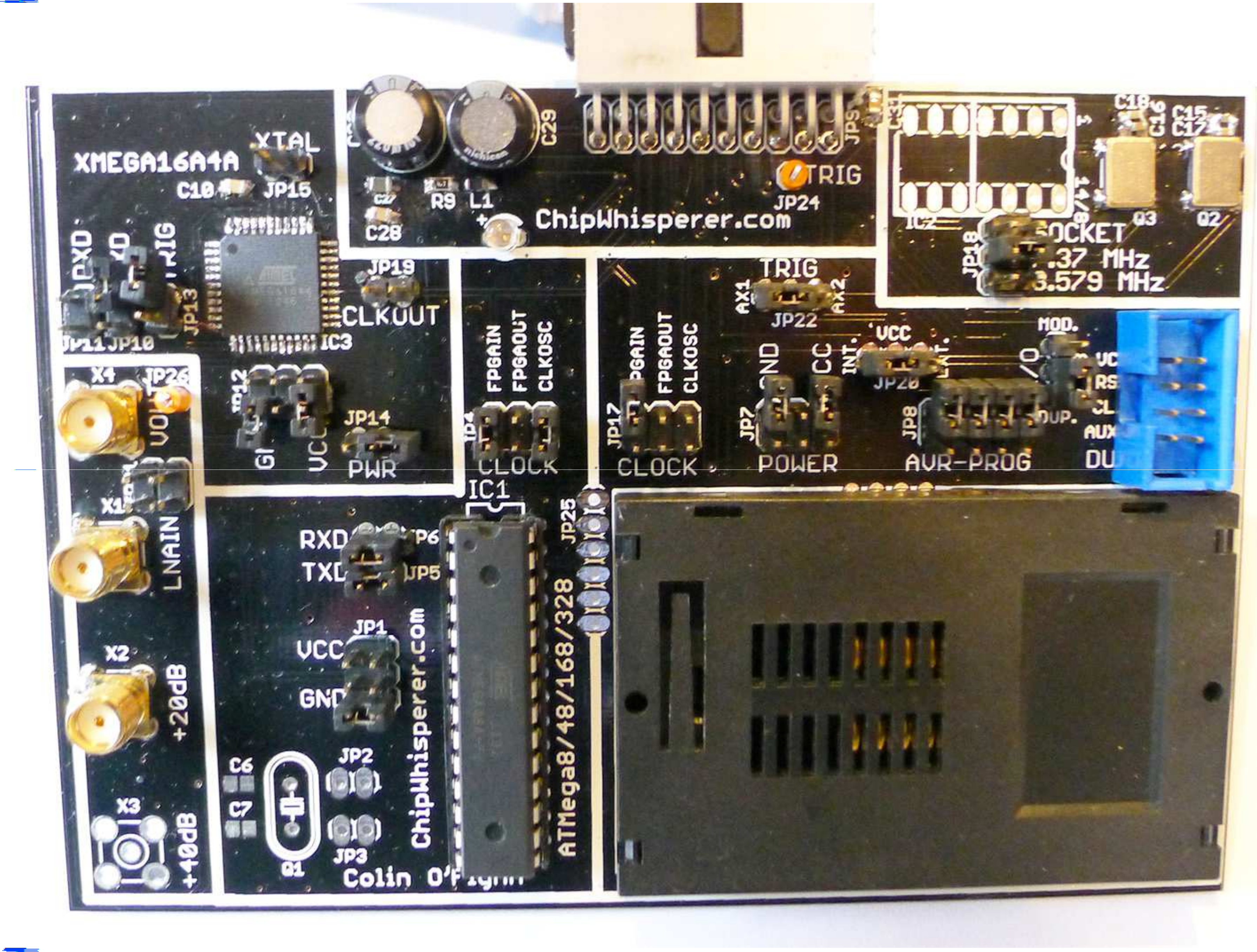


CWR2: IN-THE-MIDDLE



Feedthru Features/Notes

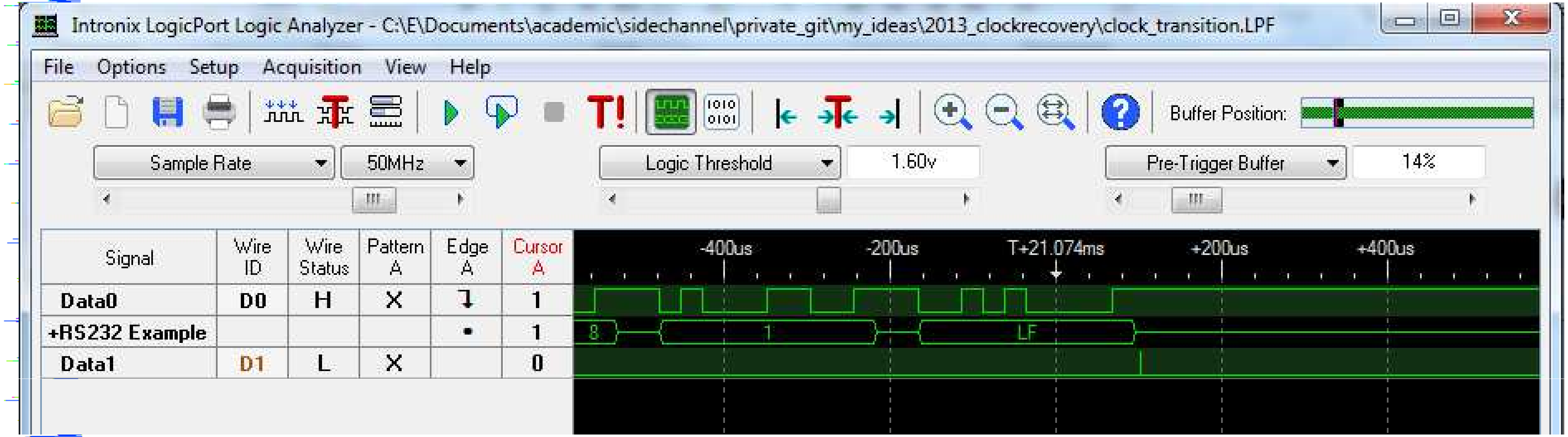
- Be VERY CAREFUL not to bridge 5V from Scard onto 3.3V rail (IO translation will work at 5V)
- Option to insert FPGA into data-line for data modification attacks
- Power Signature, Clocks, brought into CW-Rev2


XMEGA16A4A
XTAL
C10
JP15
TXD
RXD
TRIG
JP11
JP10
JP13
IC3
JP19
CLKOUT
C27
R9
L1
C28
C29
ChipWhisperer.com
TRIG
JP24
JP9
IC2
C18
C16
C15
C17
Q3
Q2
JP18
SOCKET
.37 MHz
3.579 MHz
X4
JP26
VOUT
X1
LNAIN
X2
+20dB
X3
+40dB
JP12
GND
VCC
JP14
PWR
FPGAIN
FPGAOUT
CLKOSC
JP4
CLOCK
FPGAIN
FPGAOUT
CLKOSC
JP17
CLOCK
TRIG
AX1
AX2
JP22
GND
VCC
JP7
POWER
VCC
INT.
EXT.
JP20
JP8
AVR-PROG
MOD.
I/O
DUP.
VCC
RST
CLK
AUX
DUT
IC1
JP25
ATMega8/48/168/328
RXD
TXD
JP6
JP5
JP1
VCC
GND
ChipWhisperer.com
C6
C7
Q1
JP2
JP3
Colin O'Flynn

CW-CAPTURE & CW-ANALYZER TUTORIAL

ADVANCED IO TRIGGER

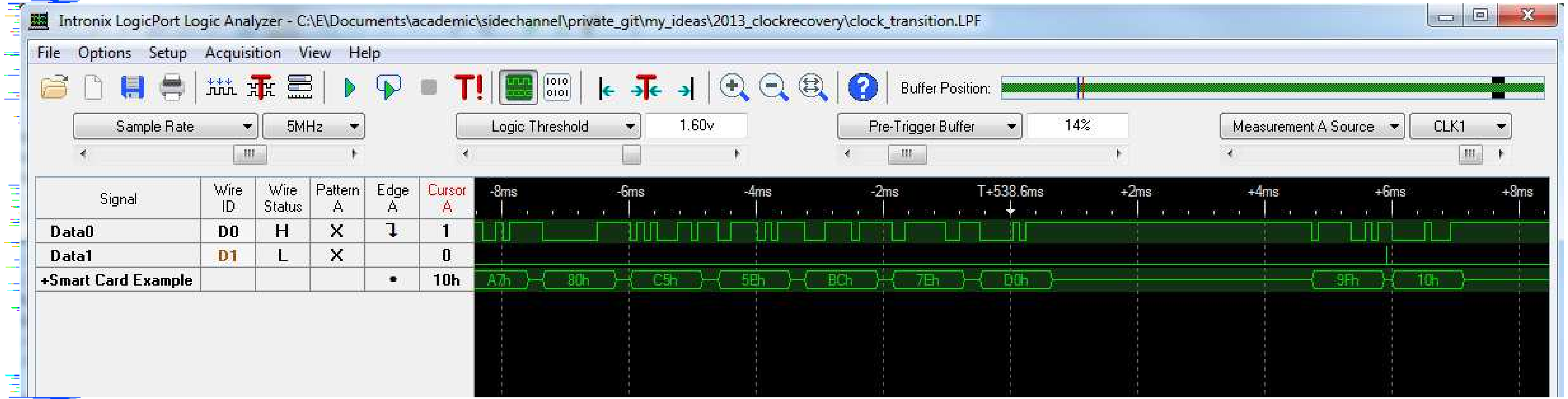
Advanced IO Trigger



Intronix LogicPort Logic Analyzer - C:\E\Documents\academic\sidechannel\private_git\my_ideas\2013_clockrecovery\clock_transition.LPF

Sample Rate: 50MHz
Logic Threshold: 1.60v
Pre-Trigger Buffer: 14%

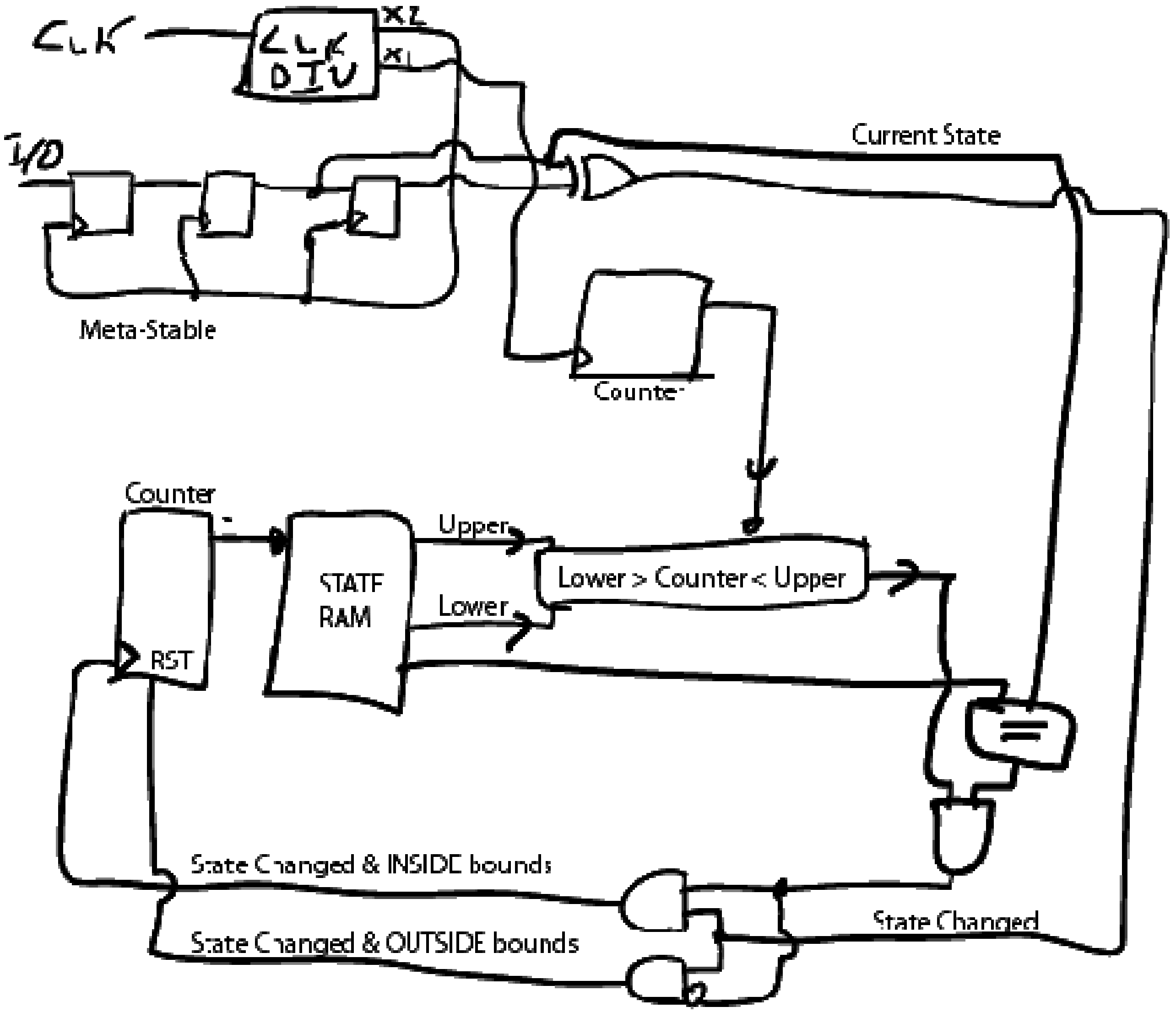
Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data0	D0	H	X	↴	1
+RS232 Example				•	1
Data1	D1	L	X		0



Intronix LogicPort Logic Analyzer - C:\E\Documents\academic\sidechannel\private_git\my_ideas\2013_clockrecovery\clock_transition.LPF

Sample Rate: 5MHz
Logic Threshold: 1.60v
Pre-Trigger Buffer: 14%
Measurement A Source: CLK1

Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data0	D0	H	X	↴	1
Data1	D1	L	X		0
+Smart Card Example				•	10h



Notes:

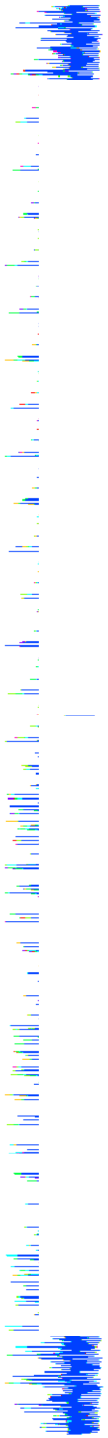
1. Use single (system) clock. CLKDIV actually generates CLKEN pulses, not clocks
2. State RAM = 18 bits. Bit [17] = line state, bits [16:8] = upper limit, bits [7:0] = low limit
3. Special limits:

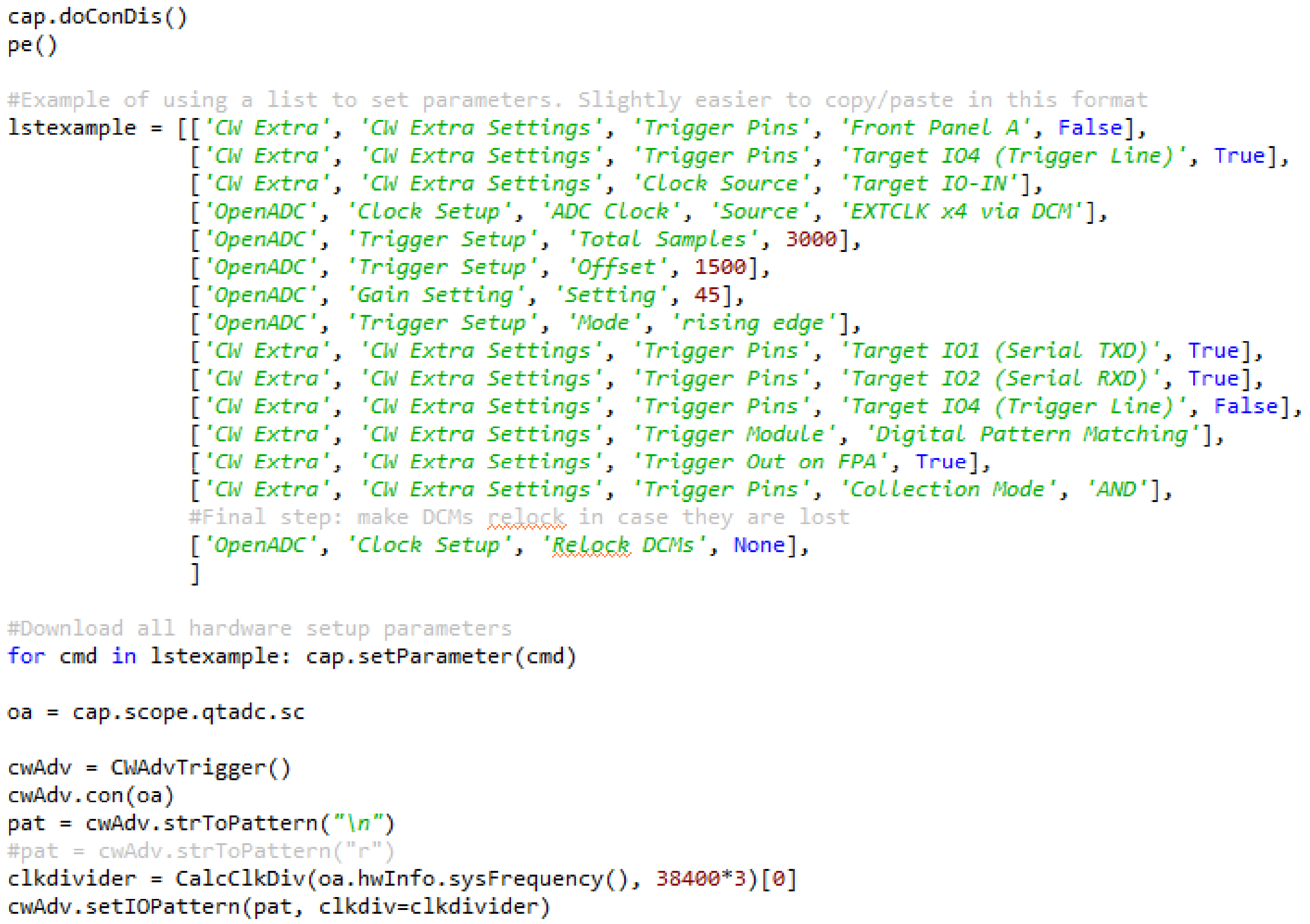
state=1, upper=512, lower=255 (e.g. all 18 bits 1) indicate 'done', trigger now.

upper = 511 indicates should transition to next state immediatly once lower limit exceeded

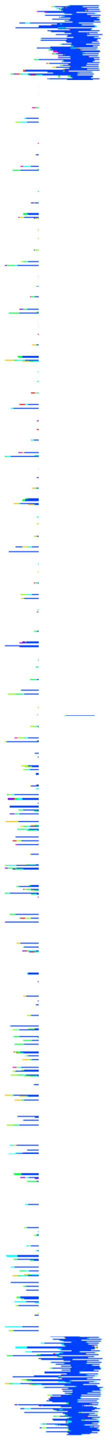
upper = 510 indicates no upper limit, but wait for both lower limit exceeded & IO state change before transitioning

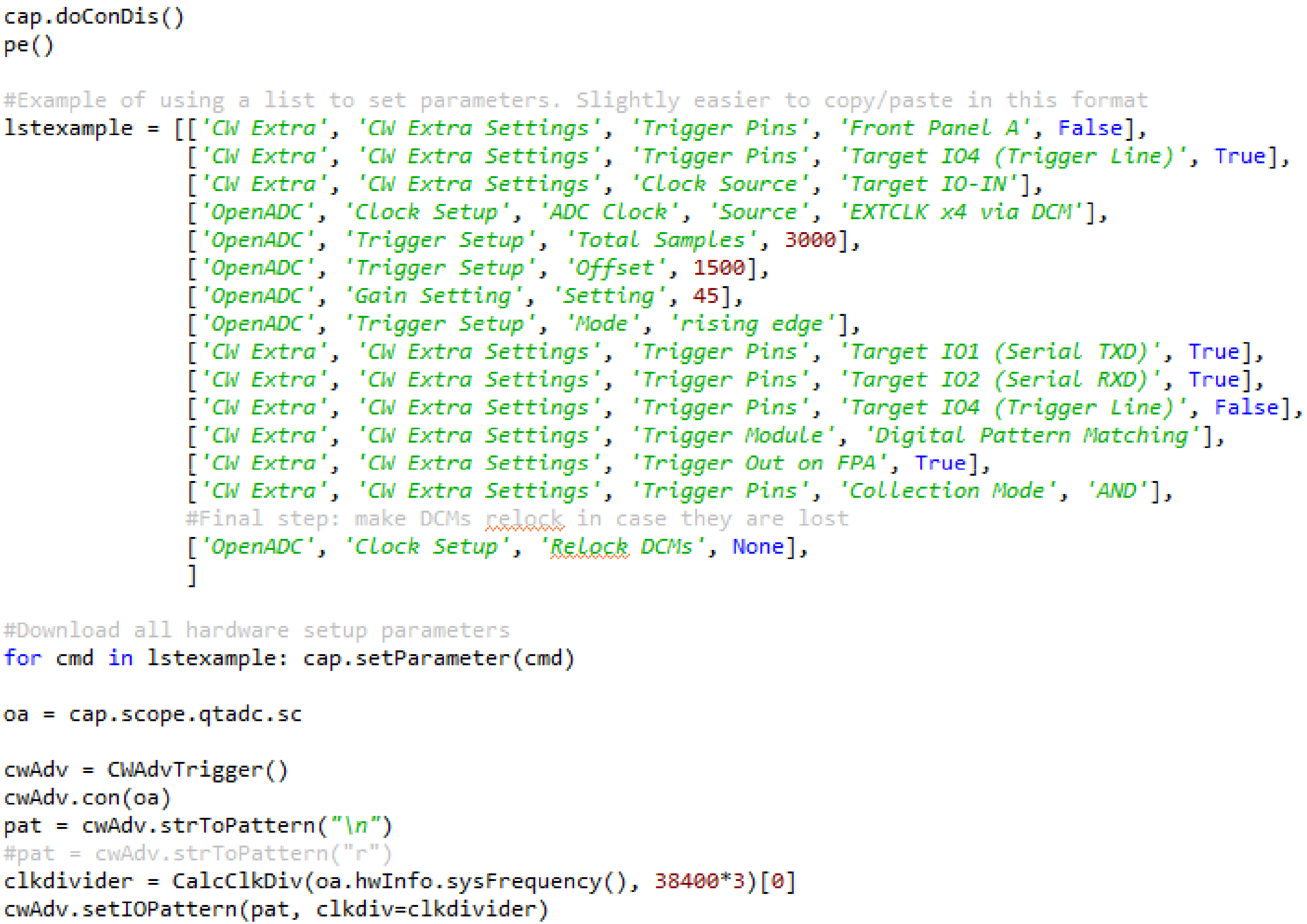
510 for example used for waiting during idle periods, where line MUST be idle for some minimum clock cycles, and wait for transition





```
cap.doConDis()
pe()

#Example of using a list to set parameters. Slightly easier to copy/paste in this format
lstexample = [['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel A', False],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', True],
              ['CW Extra', 'CW Extra Settings', 'Clock Source', 'Target IO-IN'],
              ['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'EXTCLK x4 via DCM'],
              ['OpenADC', 'Trigger Setup', 'Total Samples', 3000],
              ['OpenADC', 'Trigger Setup', 'Offset', 1500],
              ['OpenADC', 'Gain Setting', 'Setting', 45],
              ['OpenADC', 'Trigger Setup', 'Mode', 'rising edge'],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO1 (Serial TXD)', True],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO2 (Serial RXD)', True],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', False],
              ['CW Extra', 'CW Extra Settings', 'Trigger Module', 'Digital Pattern Matching'],
              ['CW Extra', 'CW Extra Settings', 'Trigger Out on FPA', True],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Collection Mode', 'AND'],
              #Final step: make DCMs relock in case they are lost
              ['OpenADC', 'Clock Setup', 'Relock DCMs', None],
              ]

#Download all hardware setup parameters
for cmd in lstexample: cap.setParameter(cmd)

oa = cap.scope.qtadc.sc

cwAdv = CWAdvTrigger()
cwAdv.con(oa)
pat = cwAdv.strToPattern("\n")
#pat = cwAdv.strToPattern("r")
clkdivider = CalcClkDiv(oa.hwInfo.sysFrequency(), 38400*3)[0]
cwAdv.setIOPattern(pat, clkdiv=clkdivider)
```





```
cap.doConDis()
pe()

#Example of using a list to set parameters. Slightly easier to copy/paste in this format
lstexample = [['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Front Panel A', False],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', True],
              ['CW Extra', 'CW Extra Settings', 'Clock Source', 'Target IO-IN'],
              ['OpenADC', 'Clock Setup', 'ADC Clock', 'Source', 'EXTCLK x4 via DCM'],
              ['OpenADC', 'Trigger Setup', 'Total Samples', 3000],
              ['OpenADC', 'Trigger Setup', 'Offset', 1500],
              ['OpenADC', 'Gain Setting', 'Setting', 45],
              ['OpenADC', 'Trigger Setup', 'Mode', 'rising edge'],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO1 (Serial TXD)', True],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO2 (Serial RXD)', True],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Target IO4 (Trigger Line)', False],
              ['CW Extra', 'CW Extra Settings', 'Trigger Module', 'Digital Pattern Matching'],
              ['CW Extra', 'CW Extra Settings', 'Trigger Out on FPA', True],
              ['CW Extra', 'CW Extra Settings', 'Trigger Pins', 'Collection Mode', 'AND'],
              #Final step: make DCMs relock in case they are lost
              ['OpenADC', 'Clock Setup', 'Relock DCMs', None],
              ]

#Download all hardware setup parameters
for cmd in lstexample: cap.setParameter(cmd)

oa = cap.scope.qtadc.sc

cwAdv = CWAdvTrigger()
cwAdv.con(oa)
pat = cwAdv.strToPattern("\n")
#pat = cwAdv.strToPattern("r")
clkdivider = CalcClkDiv(oa.hwInfo.sysFrequency(), 38400*3)[0]
cwAdv.setIOPattern(pat, clkdiv=clkdivider)
```

SELECTING HARDWARE

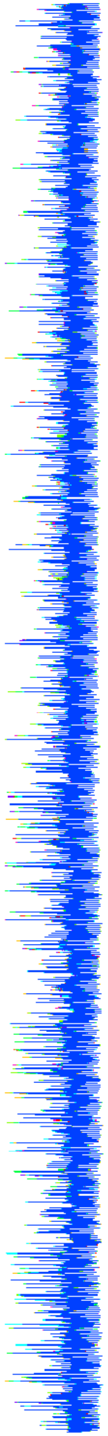
Which Hardware to Use?

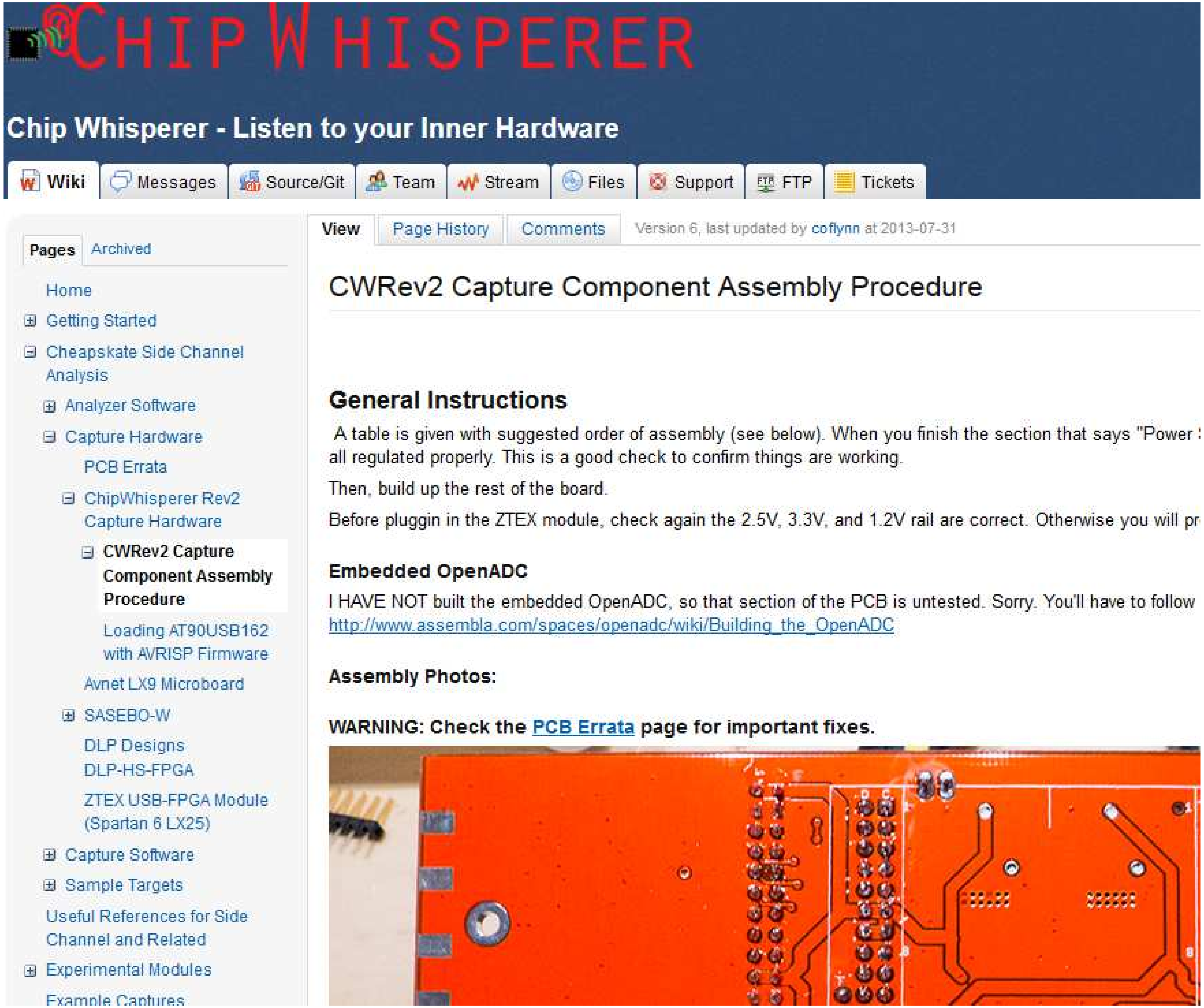
- Avnet LX9 is Cheapeast
- CW-Rev2 Hardware currently used by me
- Other Options available too

GETTING THE CW-REV2 HARDWARE

CW-Rev2 Assembling

- Most Cheap:
  - Get PCBs. Buy some parts, get free samples for some. Buy ZTEX.de module ($200). Assemble.
- Less Cheap, More Legitimate:
  - Get PCBs. Buy parts (~$170) + Buy ZTEX.de module ($200). Assemble.
- Easier Assembly:
  - Get PCBs. Buy Parts (~$130) + Buy OpenADC ($140) + Buy ZTEX.de Module ($200). Assemble.
- Easiest:
  - Buy complete kit (~$1100, not actually available at all)





Version 6, last updated by coflynn at 2013-07-31

CWRev2 Capture Component Assembly Procedure

General Instructions

A table is given with suggested order of assembly (see below). When you finish the section that says "Power : all regulated properly. This is a good check to confirm things are working.

Then, build up the rest of the board.

Before pluggin in the ZTEX module, check again the 2.5V, 3.3V, and 1.2V rail are correct. Otherwise you will pr

Embedded OpenADC

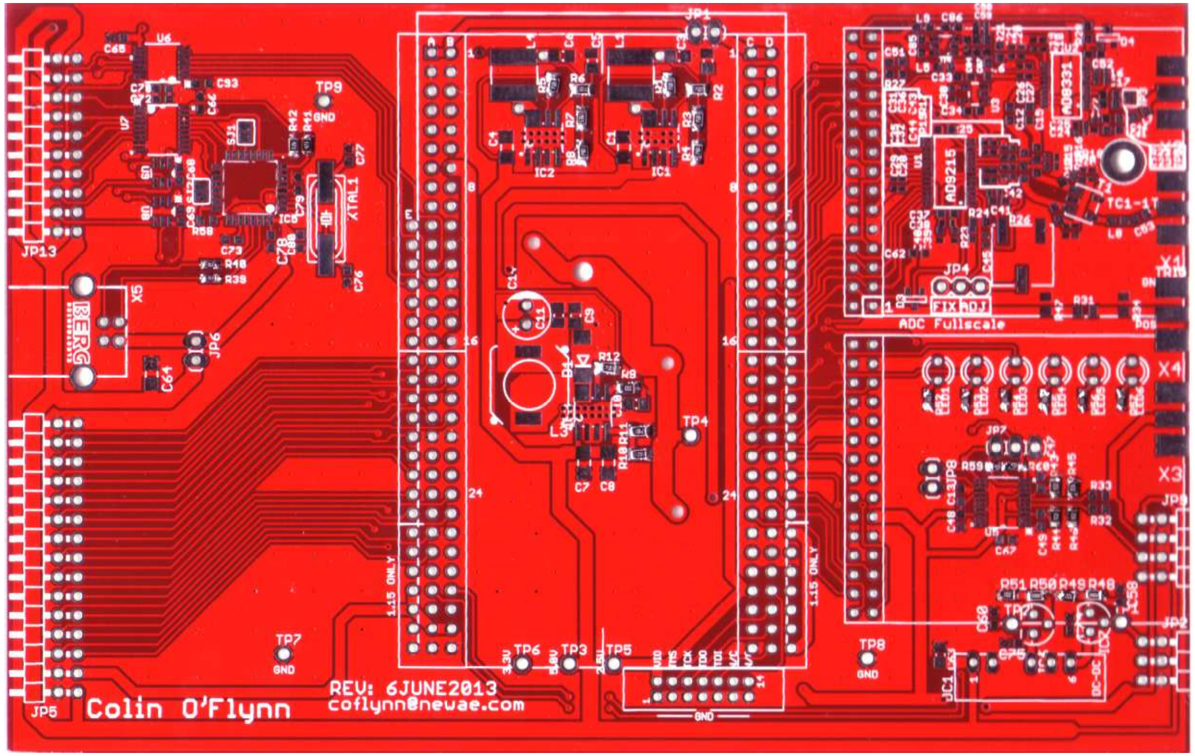
I HAVE NOT built the embedded OpenADC, so that section of the PCB is untested. Sorry. You'll have to follow http://www.assembla.com/spaces/openadc/wiki/Building_the_OpenADC

Assembly Photos:

WARNING: Check the PCB Errata page for important fixes.

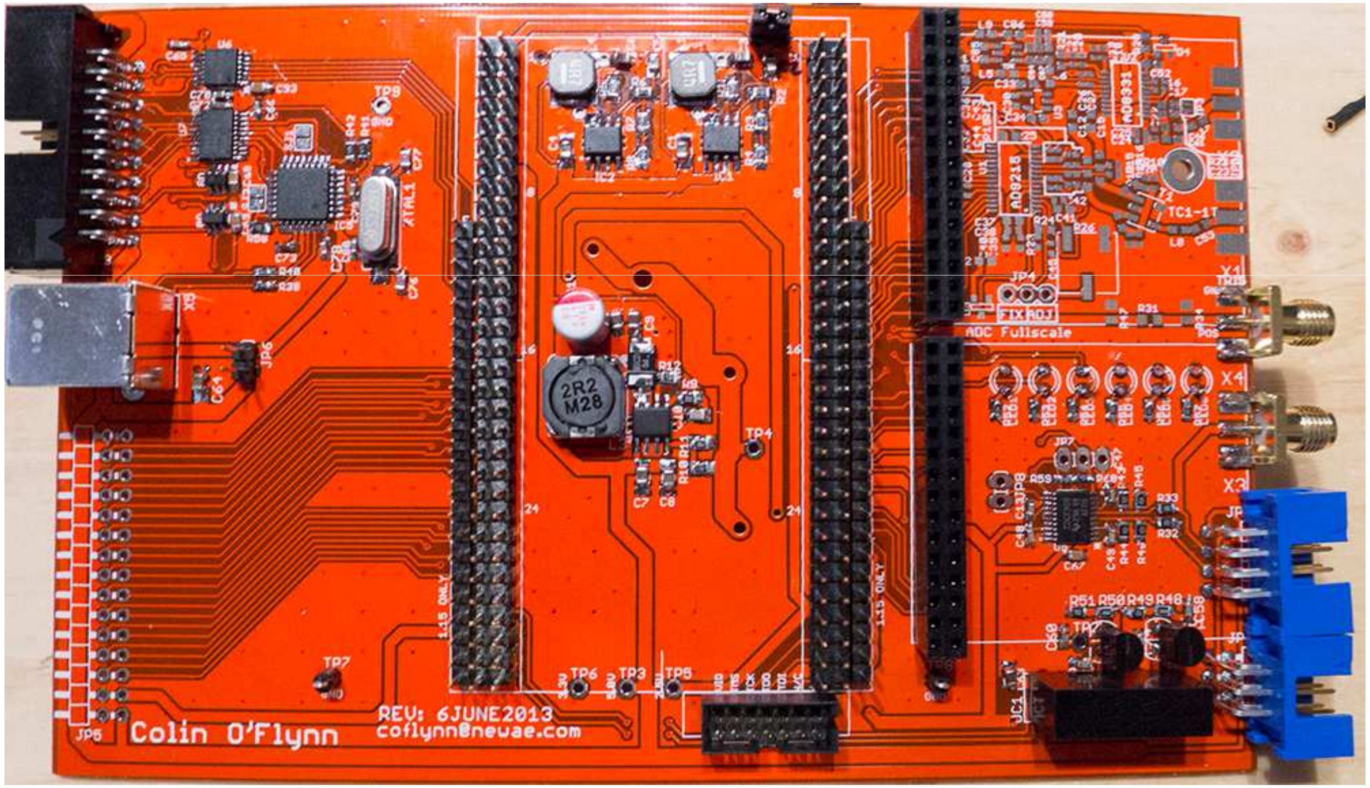
How to Build?

- Step 1: Carefully mount resistors



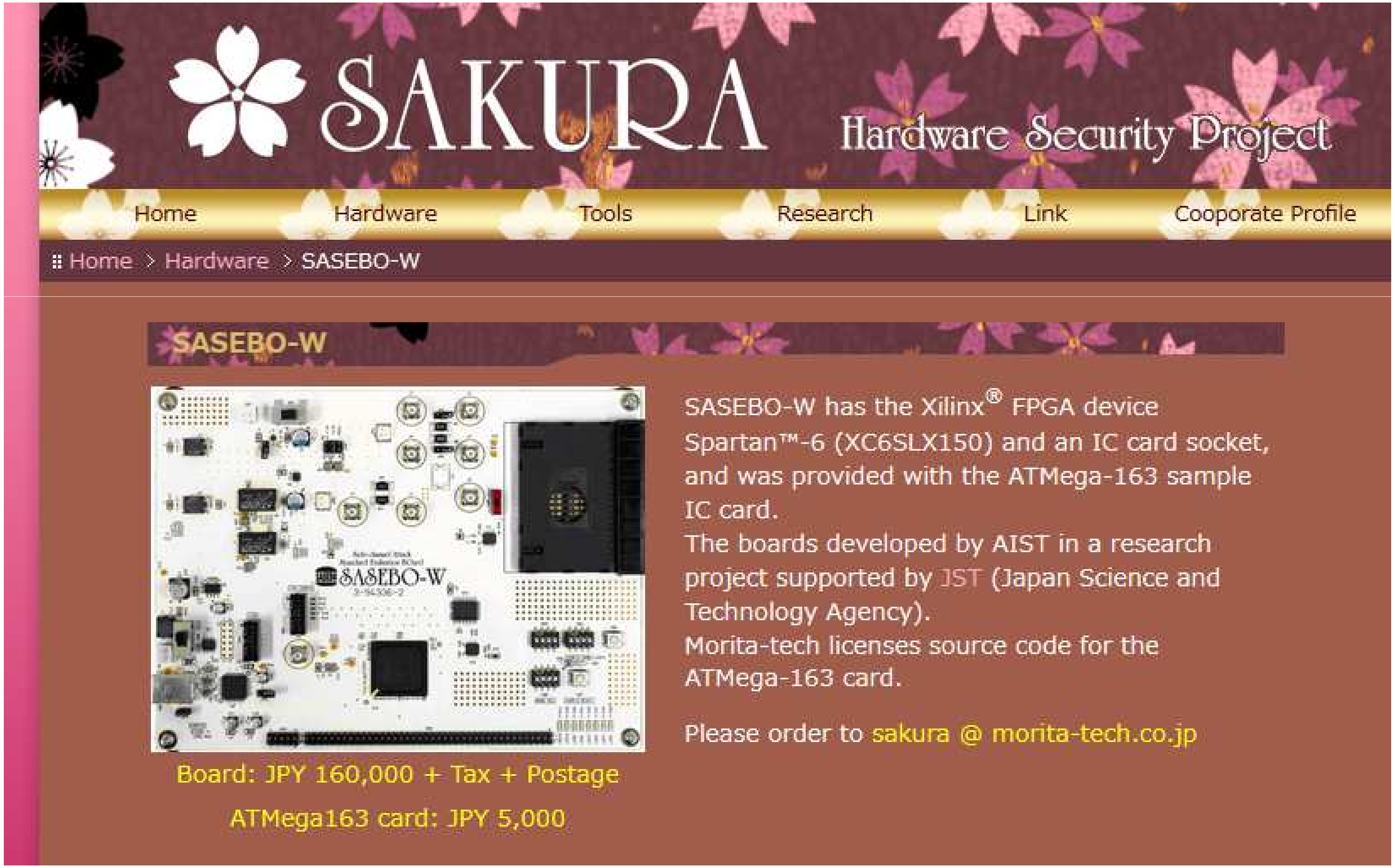
How to Build?

- Step 2: Mount everything else



SASEBO-W BOARD

OpenADC



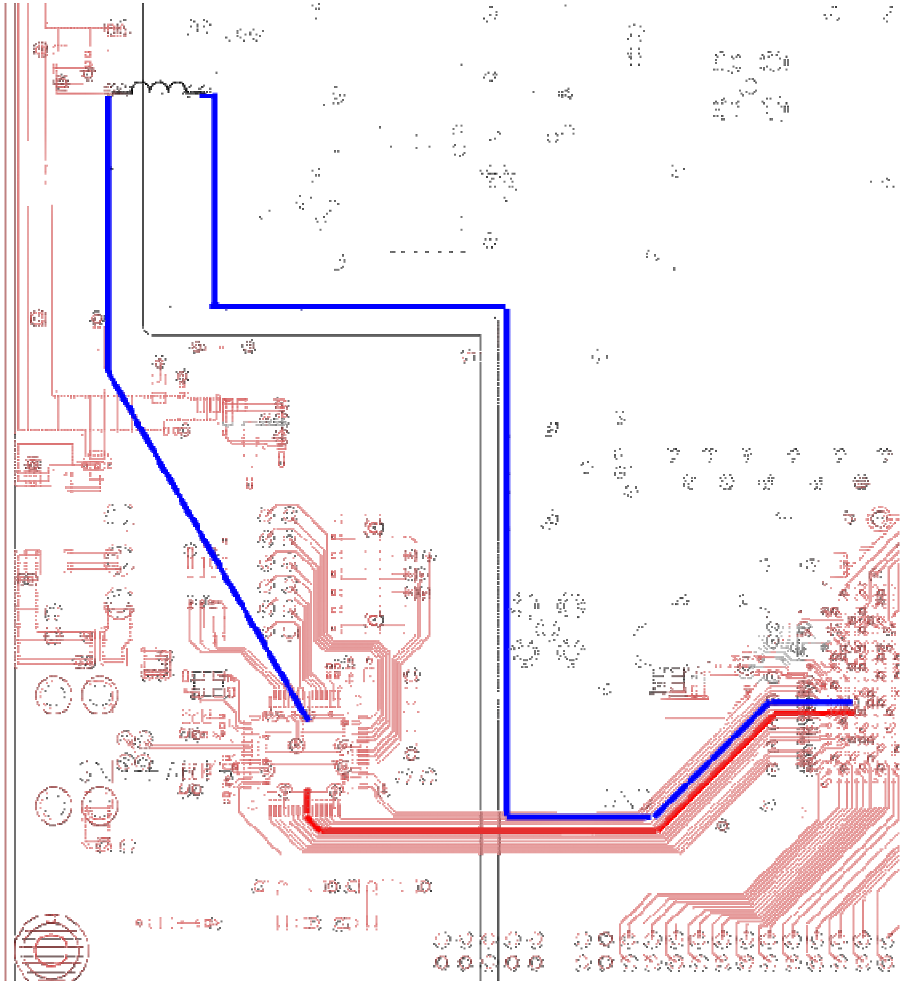
SASEBO-W has the Xilinx® FPGA device Spartan™-6 (XC6SLX150) and an IC card socket, and was provided with the ATMega-163 sample IC card.
The boards developed by AIST in a research project supported by JST (Japan Science and Technology Agency).
Morita-tech licenses source code for the ATMega-163 card.

Please order to sakura @ morita-tech.co.jp

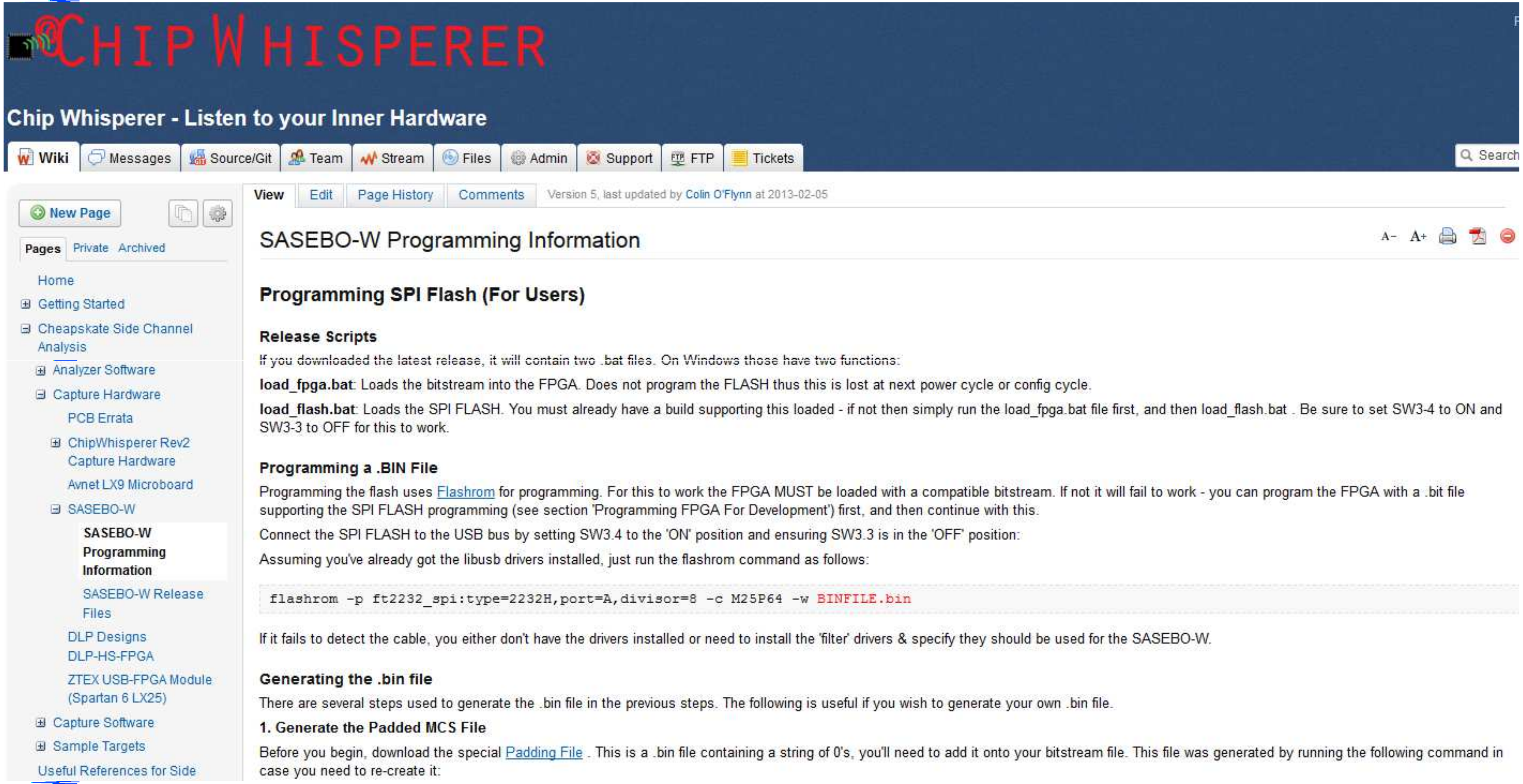
Board: JPY 160,000 + Tax + Postage

ATMega163 card: JPY 5,000

Hardware Issues



Loading the FPGA: Method #1



SASEBO-W Programming Information

Programming SPI Flash (For Users)

Release Scripts

If you downloaded the latest release, it will contain two .bat files. On Windows those have two functions:

load_fpga.bat: Loads the bitstream into the FPGA. Does not program the FLASH thus this is lost at next power cycle or config cycle.

load_flash.bat: Loads the SPI FLASH. You must already have a build supporting this loaded - if not then simply run the load_fpga.bat file first, and then load_flash.bat . Be sure to set SW3-4 to ON and SW3-3 to OFF for this to work.

Programming a .BIN File

Programming the flash uses Flashrom for programming. For this to work the FPGA MUST be loaded with a compatible bitstream. If not it will fail to work - you can program the FPGA with a .bit file supporting the SPI FLASH programming (see section 'Programming FPGA For Development') first, and then continue with this.

Connect the SPI FLASH to the USB bus by setting SW3.4 to the 'ON' position and ensuring SW3.3 is in the 'OFF' position:

Assuming you've already got the libusb drivers installed, just run the flashrom command as follows:

```
flashrom -p ft2232_spi:type=2232H,port=A,divisor=8 -c M25P64 -w BINFILE.bin
```

If it fails to detect the cable, you either don't have the drivers installed or need to install the 'filter' drivers & specify they should be used for the SASEBO-W.

Generating the .bin file

There are several steps used to generate the .bin file in the previous steps. The following is useful if you wish to generate your own .bin file.

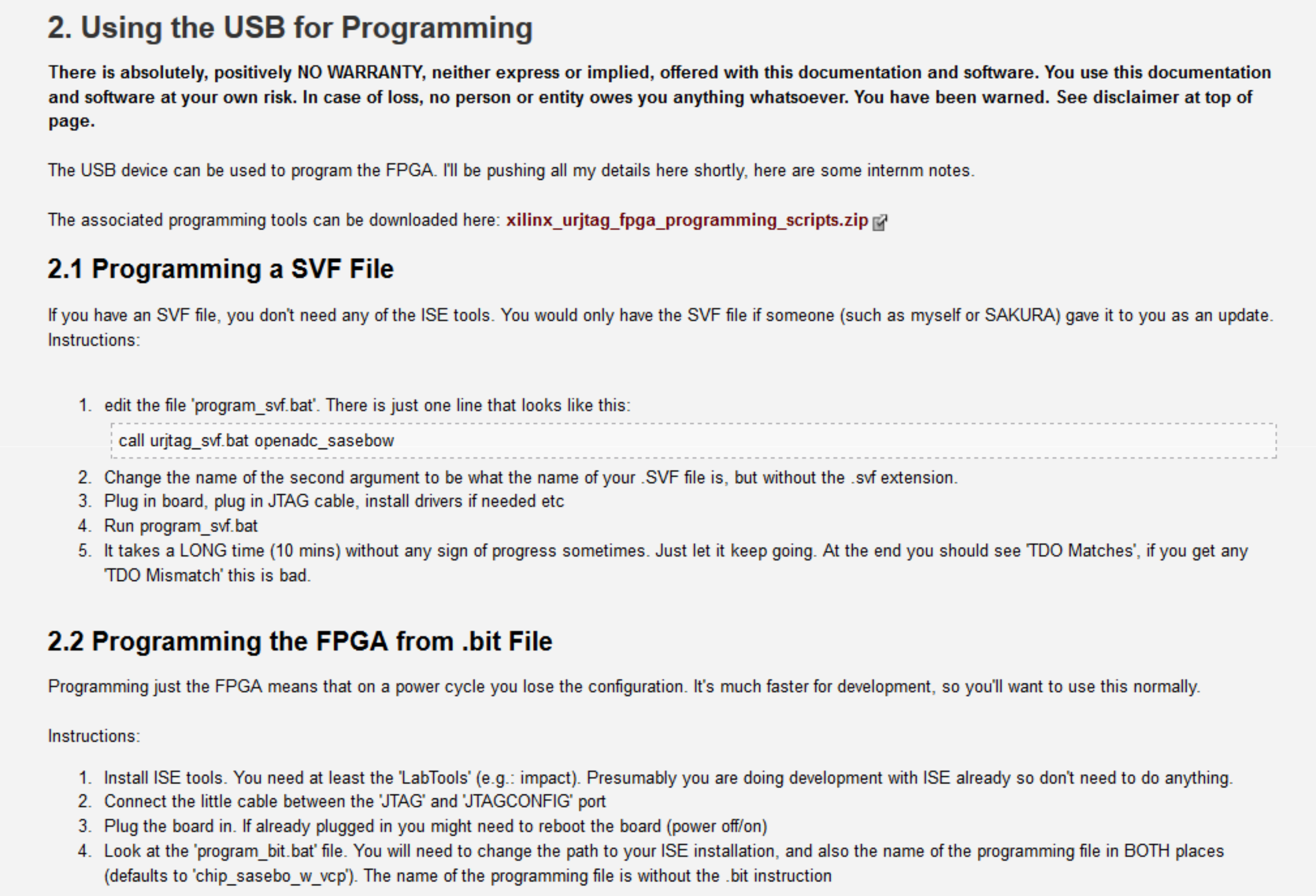
1. Generate the Padded MCS File

Before you begin, download the special Padding File . This is a .bin file containing a string of 0's, you'll need to add it onto your bitstream file. This file was generated by running the following command in case you need to re-create it:

Loading the FPGA: Method #1

- Load bitstream WITH SPI Flash programming interface into FPGA (default does not), fairly slow
- Program SPI Flash through faster interface

Loading the FPGA: Method #2



2. Using the USB for Programming

There is absolutely, positively NO WARRANTY, neither express or implied, offered with this documentation and software. You use this documentation and software at your own risk. In case of loss, no person or entity owes you anything whatsoever. You have been warned. See disclaimer at top of page.

The USB device can be used to program the FPGA. I'll be pushing all my details here shortly, here are some internm notes.

The associated programming tools can be downloaded here: **xilinx_urjtag_fpga_programming_scripts.zip**

2.1 Programming a SVF File

If you have an SVF file, you don't need any of the ISE tools. You would only have the SVF file if someone (such as myself or SAKURA) gave it to you as an update. Instructions:

1. edit the file 'program_svf.bat'. There is just one line that looks like this:
   ```
   call urjtag_svf.bat openadc_sasebow
   ```
2. Change the name of the second argument to be what the name of your .SVF file is, but without the .svf extension.
3. Plug in board, plug in JTAG cable, install drivers if needed etc
4. Run program_svf.bat
5. It takes a LONG time (10 mins) without any sign of progress sometimes. Just let it keep going. At the end you should see 'TDO Matches', if you get any 'TDO Mismatch' this is bad.

2.2 Programming the FPGA from .bit File

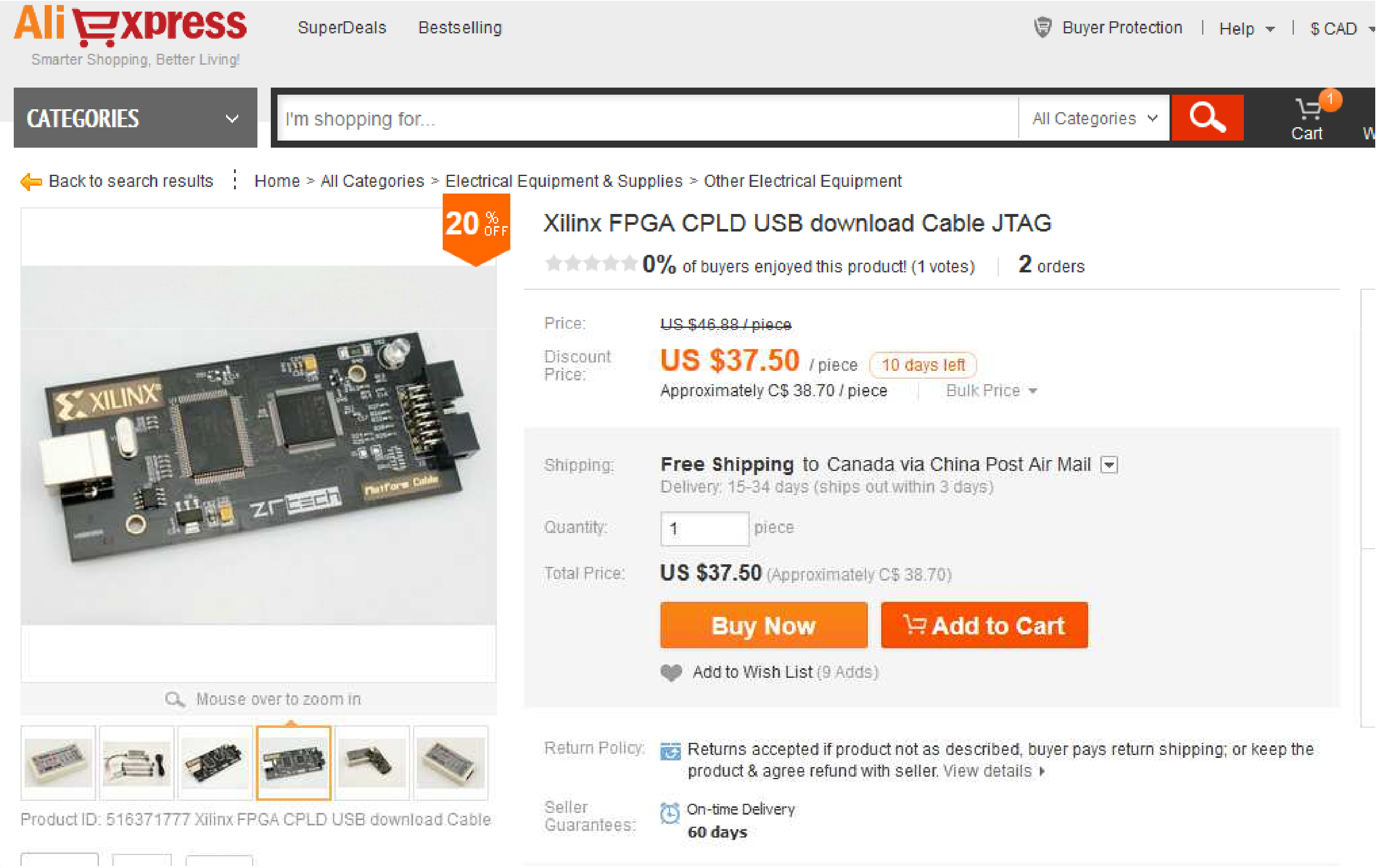
Programming just the FPGA means that on a power cycle you lose the configuration. It's much faster for development, so you'll want to use this normally.

Instructions:

1. Install ISE tools. You need at least the 'LabTools' (e.g.: impact). Presumably you are doing development with ISE already so don't need to do anything.
2. Connect the little cable between the 'JTAG' and 'JTAGCONFIG' port
3. Plug the board in. If already plugged in you might need to reboot the board (power off/on)
4. Look at the 'program_bit.bat' file. You will need to change the path to your ISE installation, and also the name of the programming file in BOTH places (defaults to 'chip_sasebo_w_vcp'). The name of the programming file is without the .bit instruction

http://newae.com/tiki-index.php?page=SASEBOW

Loading the FPGA: Method #3



AliExpress
Smarter Shopping, Better Living!

SuperDeals Bestselling

Buyer Protection | Help | $ CAD

CATEGORIES

I'm shopping for...

All Categories

Cart

Back to search results | Home > All Categories > Electrical Equipment & Supplies > Other Electrical Equipment

20 % OFF

Xilinx FPGA CPLD USB download Cable JTAG

0% of buyers enjoyed this product! (1 votes) | 2 orders

Price: ~~US $46.88 / piece~~

Discount Price: US $37.50 / piece 10 days left

Approximately C$ 38.70 / piece | Bulk Price

Shipping: **Free Shipping** to Canada via China Post Air Mail
Delivery: 15-34 days (ships out within 3 days)

Quantity: 1 piece

Total Price: **US $37.50** (Approximately C$ 38.70)

Buy Now | Add to Cart

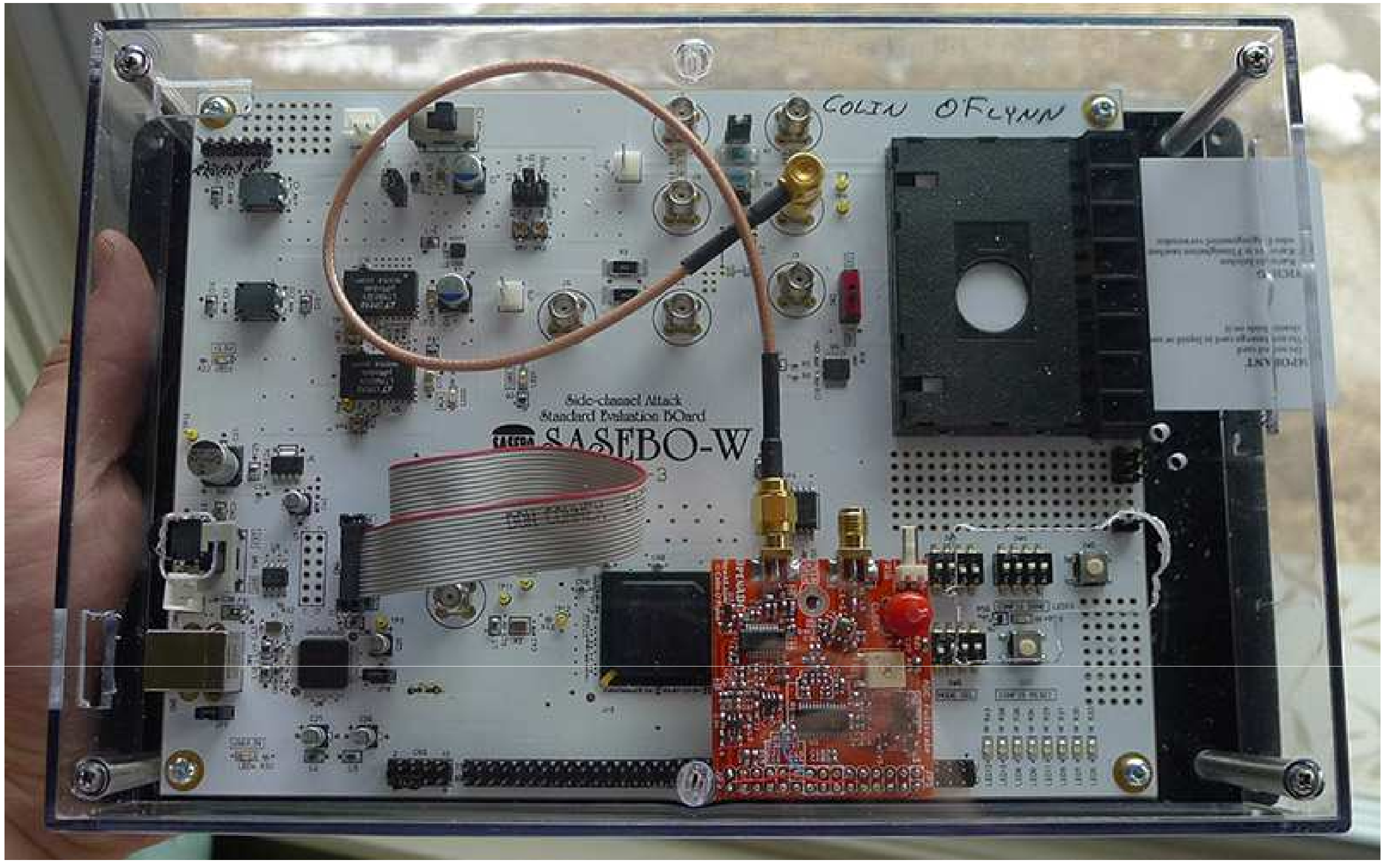
Add to Wish List (9 Adds)

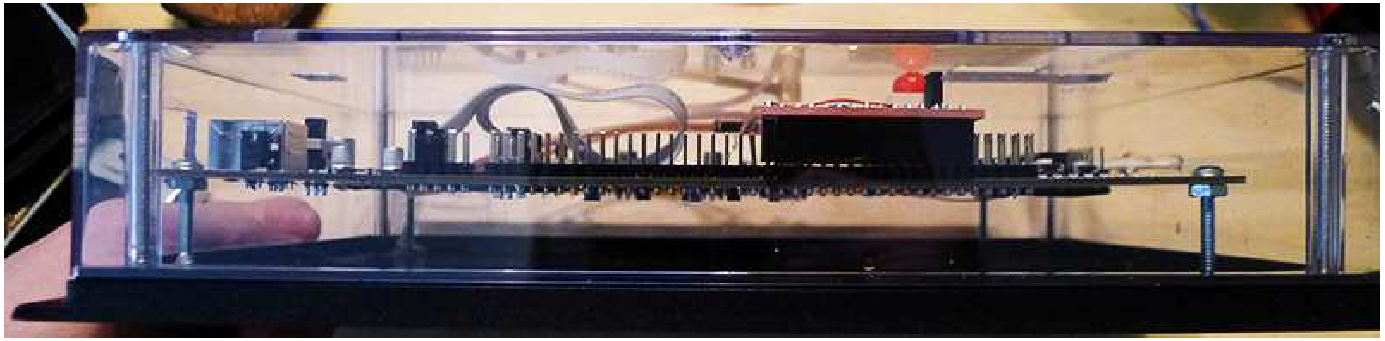
Mouse over to zoom in

Product ID: 516371777 Xilinx FPGA CPLD USB download Cable

Return Policy: Returns accepted if product not as described, buyer pays return shipping; or keep the product & agree refund with seller. View details

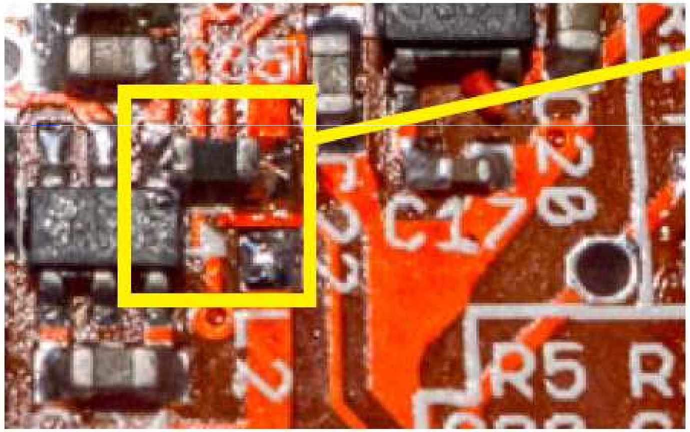
Seller Guarantees: On-time Delivery **60 days**





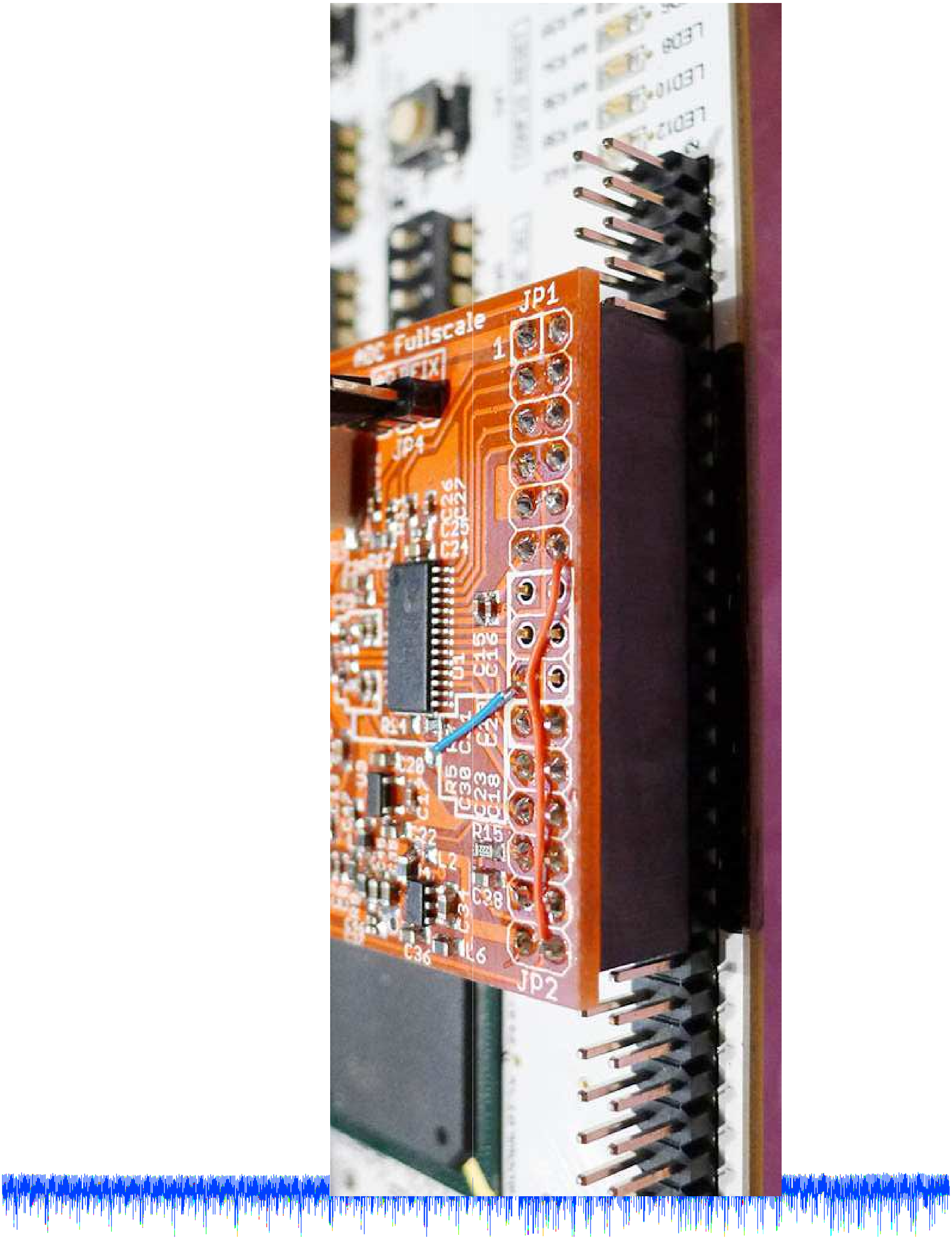
http://www.digikey.com/product-detail/en/WM093RC,BK/SRW093-RCB-ND/2286068

Modify OpenADC for 2.5V Operation



By default the 3.0V rail for the ADC is regulated from the AVCC pin, requiring at minimum 3.1V on the AVCC pin.

L2 can be moved as here to take power from the output of the 5V boost converter. This allows you to power the AVCC pin from as low as 2.5V. The entire OpenADC can be run from a 2.5V system with this change.


ADC Fullscale
JP1
1
JP4
JP2
U1
R14
R15
C15
C16
C24
C25
C26
C27
C28
C38
C36
C22
L2
L6

AVNET LX9 MICROBOARD

LX9 Microboard

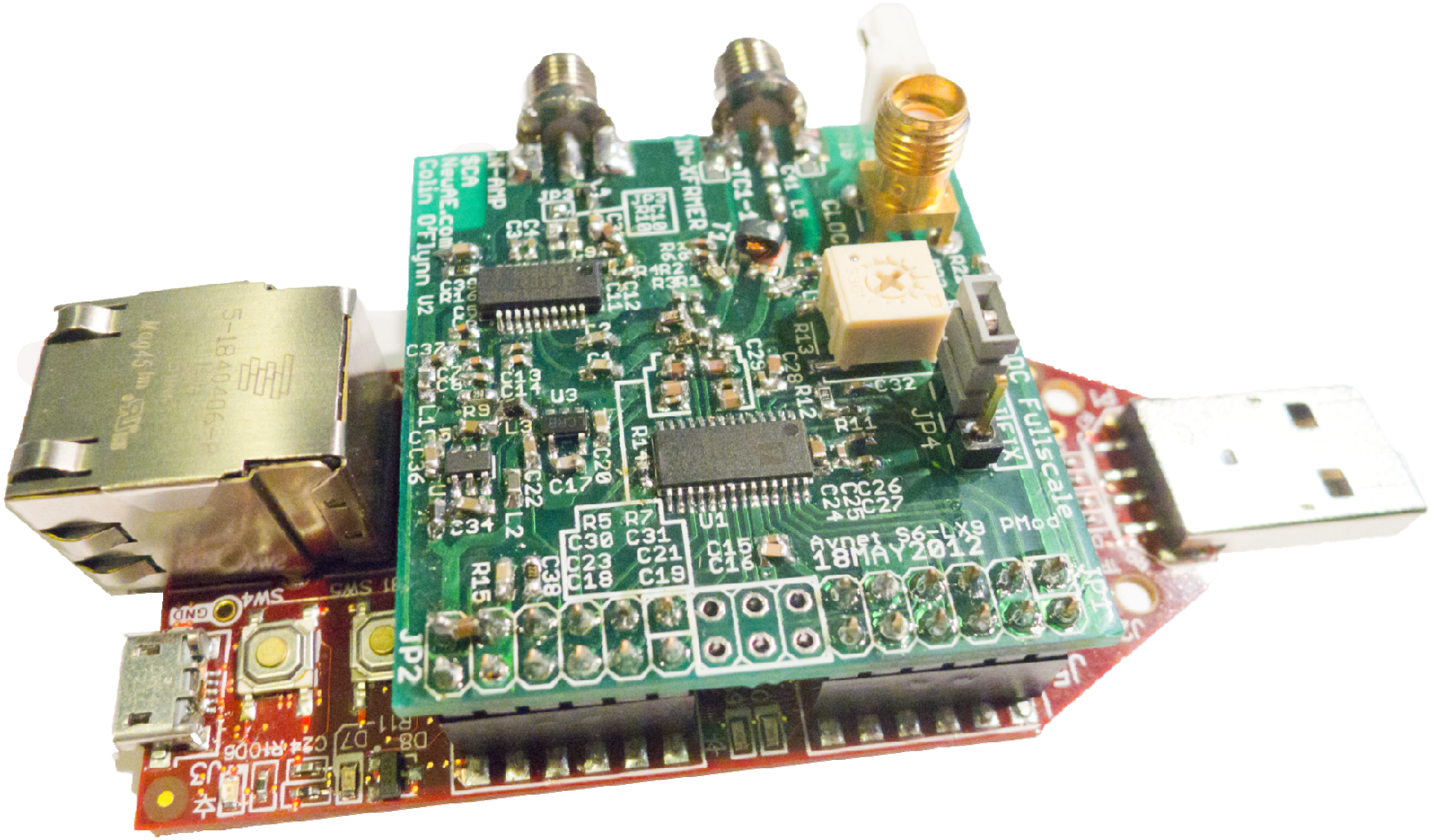
Advantages:

- Cheap
- Small

Disadvantages:

- Very slow capture
- Smaller FPGA, fits less features
- Limited IO (two IO lines… usually one clock, one trigger)

LX9 Microboard



ADDING MODULES TO FPGA

Basic Instructions

- Provide standard register-based Interface

```
module reg_triggerio(
         input          reset_i,
         input          clk,
         input [5:0]    reg_address,  // Address of register
         input [15:0]   reg_bytecnt,  // Current byte count
         input [7:0]    reg_datai,    // Data to write
         output [7:0]   reg_datao,    // Data to read
         input [15:0]   reg_size,     // Total size being read/write
         input          reg_read,     // Read flag
         input          reg_write,     // Write flag
         input          reg_addrvalid,// Address valid flag
         output         reg_stream,

         input [5:0]    reg_hypaddress,
         output [15:0]  reg_hyplen,

         input          io_line,
         output reg     trig_out

   );
```

Basic Instructions

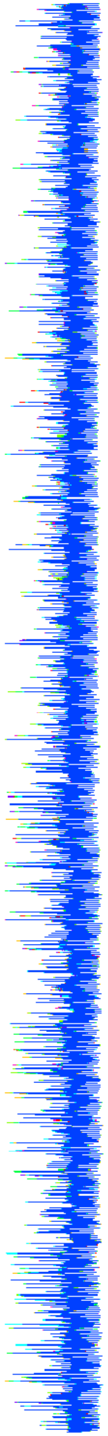
- Define Address(es) used by your module
- Define size of registers at those addresses
  - Each register can be from 1-32768 bytes long
  - Address space only 6 bits, so don't waste addresses!

Basic Instructions

- Use scripting/python console to help debug issues

```
 sc = cap.scope.qtadc.sc
#Read 4 bytes from address 0x34
stat = sc.sendMessage(CODE_READ, 0x34,
Validate=False, maxResp=4)
```

ADDING SOFTWARE MODULES



Adding Targets

Adding Scopes

Adding Attacks

Adding Preprocessing

FUTURE DIRECTIONS

Short Term

- Documentation
- Adding new FPGA Modules
  - Correlation/Matching trigger
- Fixing/Adding Software Features
  - Working Project Files

Medium Term

- Glitching Support
- PLL for Clock Recovery

Longer Term

- Redesigned capture HW for higher bandwidth
- API to use capture HW from other Software

Questions?

Colin O’Flynn

coflynn@newae.com

My Site: NewAE.com

This Project: ChipWhisperer.com