A Lightbulb Worm?

A teardown of the Philips Hue.

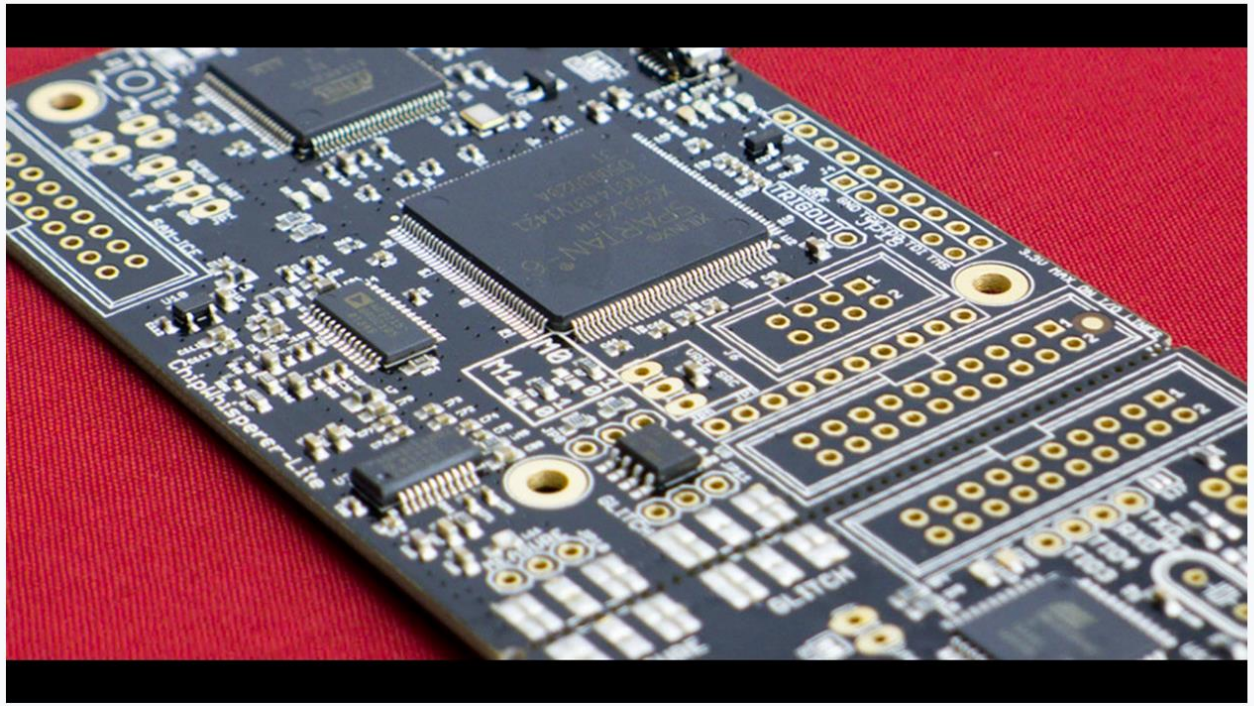
Colin O'Flynn

(with special appearance by Eyal Ronen)

black hat USA 2016

About Me

ChipWhisperer-Lite: A New Era of Hardware Security Research



Embedded security - is it an oxymoron? Learn the truth through a series of hands-on labs targeting computer and electrical engineers.

Created by

Colin O'Flynn

331 backers pledged $88,535 CAD to help bring this project to life.

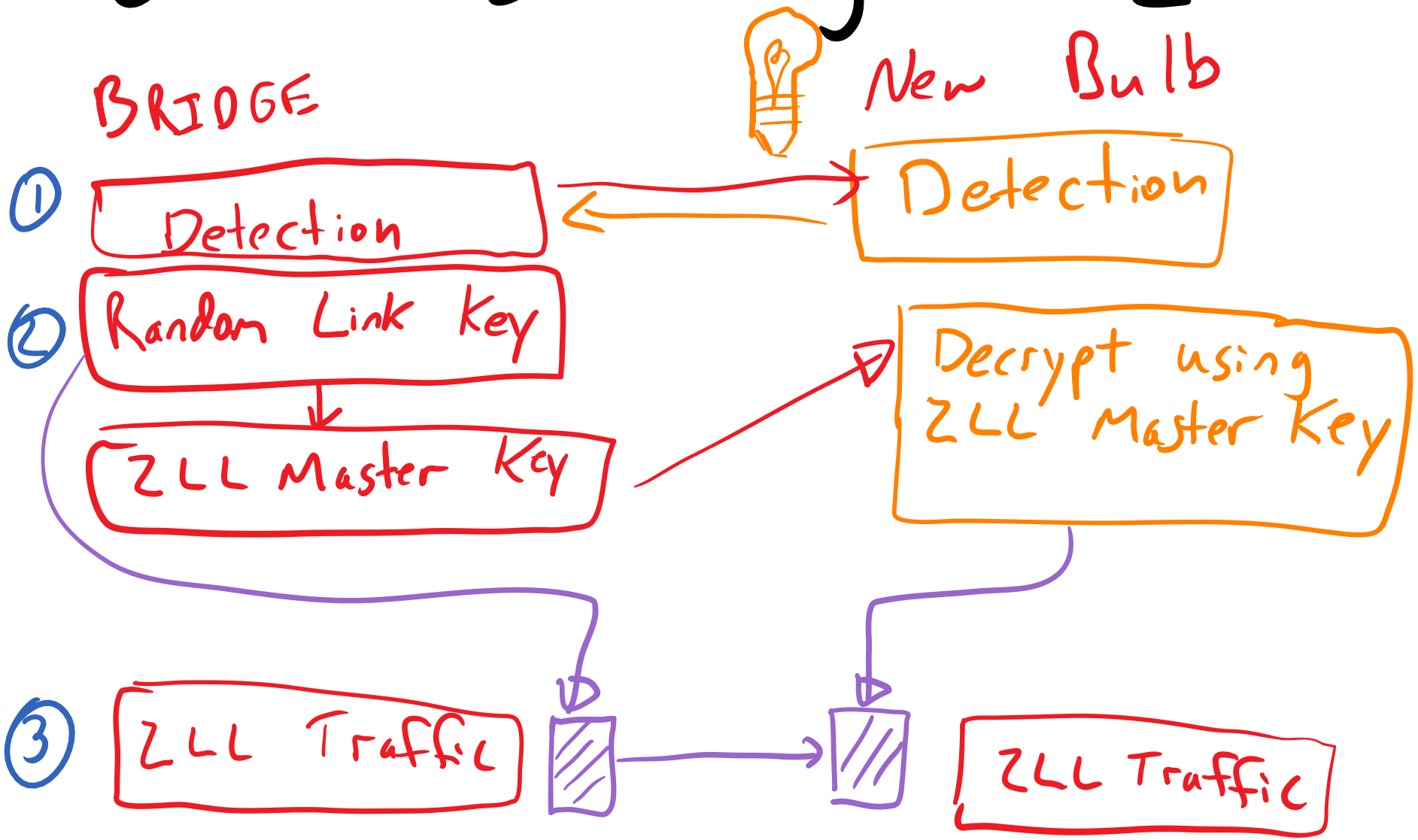

PHILIPS
I WANT / JE VEUX
hue
Available on the App Store


PHILIPS

HACKS?

1. Brick light-bulb by OTA firmware update.
2. Move bulb onto unavailable network, or control bulb.
3. Hack <u>into</u> bridge, access ethernet.
4. Malware in bulbs to do #3?

Understanding ZLL



IMPORTANT NOTE

The following 7 slides are work that Colin O'Flynn HAS NOT contributed too.

If you are crediting the takeover attack, please reference the names in the slides, do not add me (Colin). I'm only talking about the teardown & some rooting stuff.

Taking full control of already installed smart lights from long distance

Eyal Ronen

Joint work with Prof. Adi Shamir and Mr. Achi-Or Weingarten

Weizmann Institute of Science

ZLL Touchlink mechanism

- Allows adding a light to the network
- Enforces a distance check mechanism (10 – 20 cm)

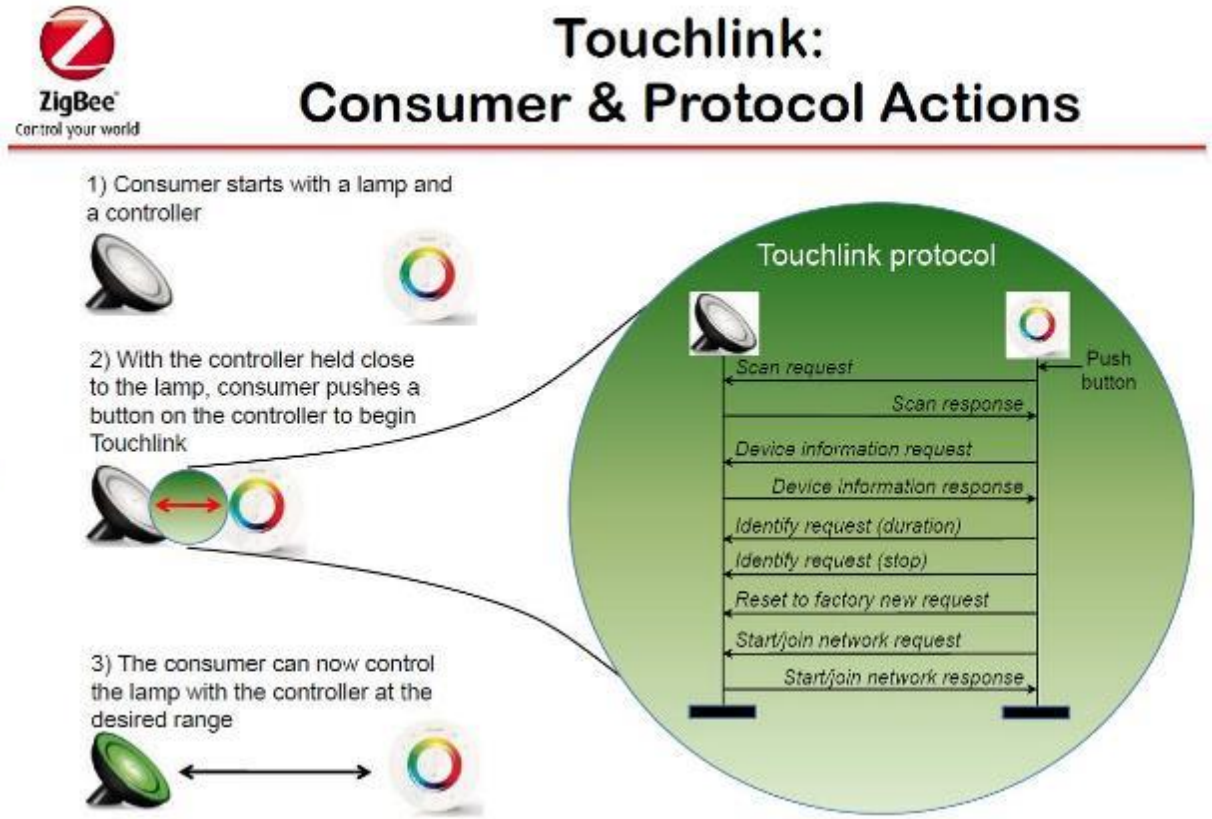
How can I connect multiple lamps to a Hue dimmer switch?

Date published 10/1/2015 - Last edited 7/4/2016

Power up the lamp you want to connect. Hold the Hue dimmer switch close to the lamp (<15cm) and press and hold the ON button until the LED on the front turns green (during this process lamp will blink several times).

Did this answer your question?

Yes | No



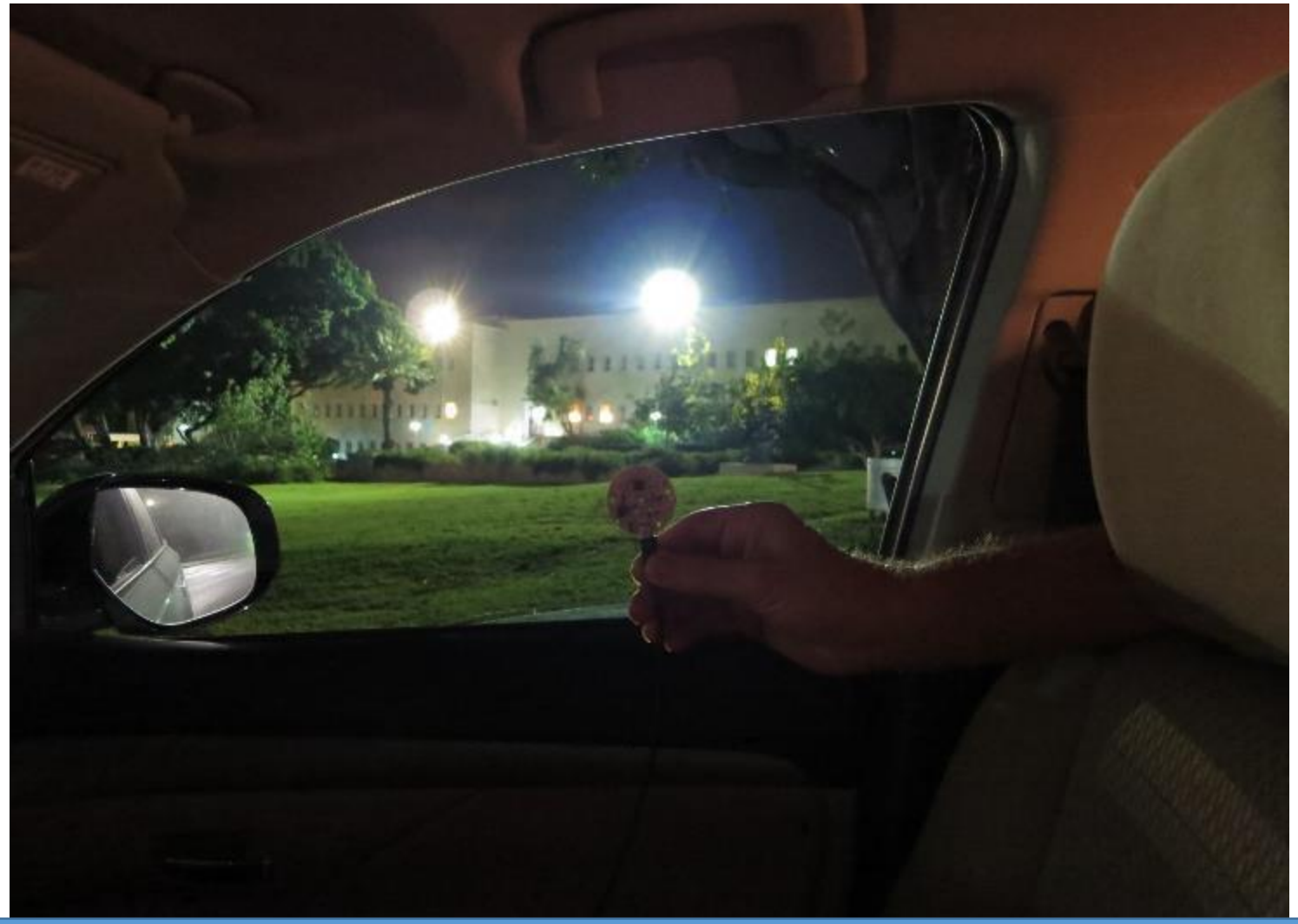
Work by Eyal Ronen et al (NOT Colin O'Flynn)

Our Attack

- Our attack
  - Take full control over Philips Hue lights from large distance
  - Fully autonomous attack kit
    - Of the shelf ZigBee evaluation board (TI cc2531 Zlight2)
    - USB power bank
  - War-driving and War-flying demonstrated

- Ethical disclosure
  - Full disclosure to Philips Lightning
  - Currently under internal investigation
  - Will allow more time before publishing the details

Zigbee War-driving

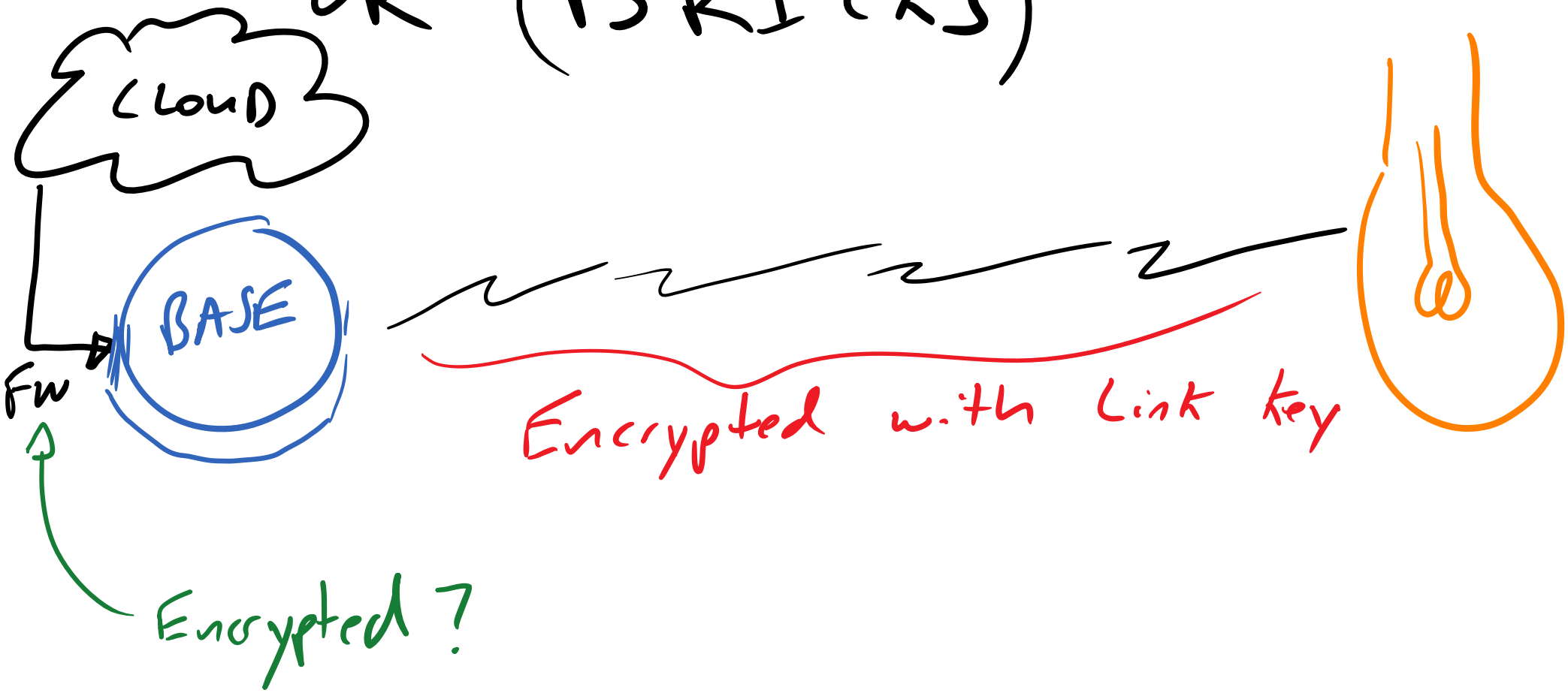


Work by Eyal Ronen et al (NOT Colin O’Flynn)

ZigBee War-flying

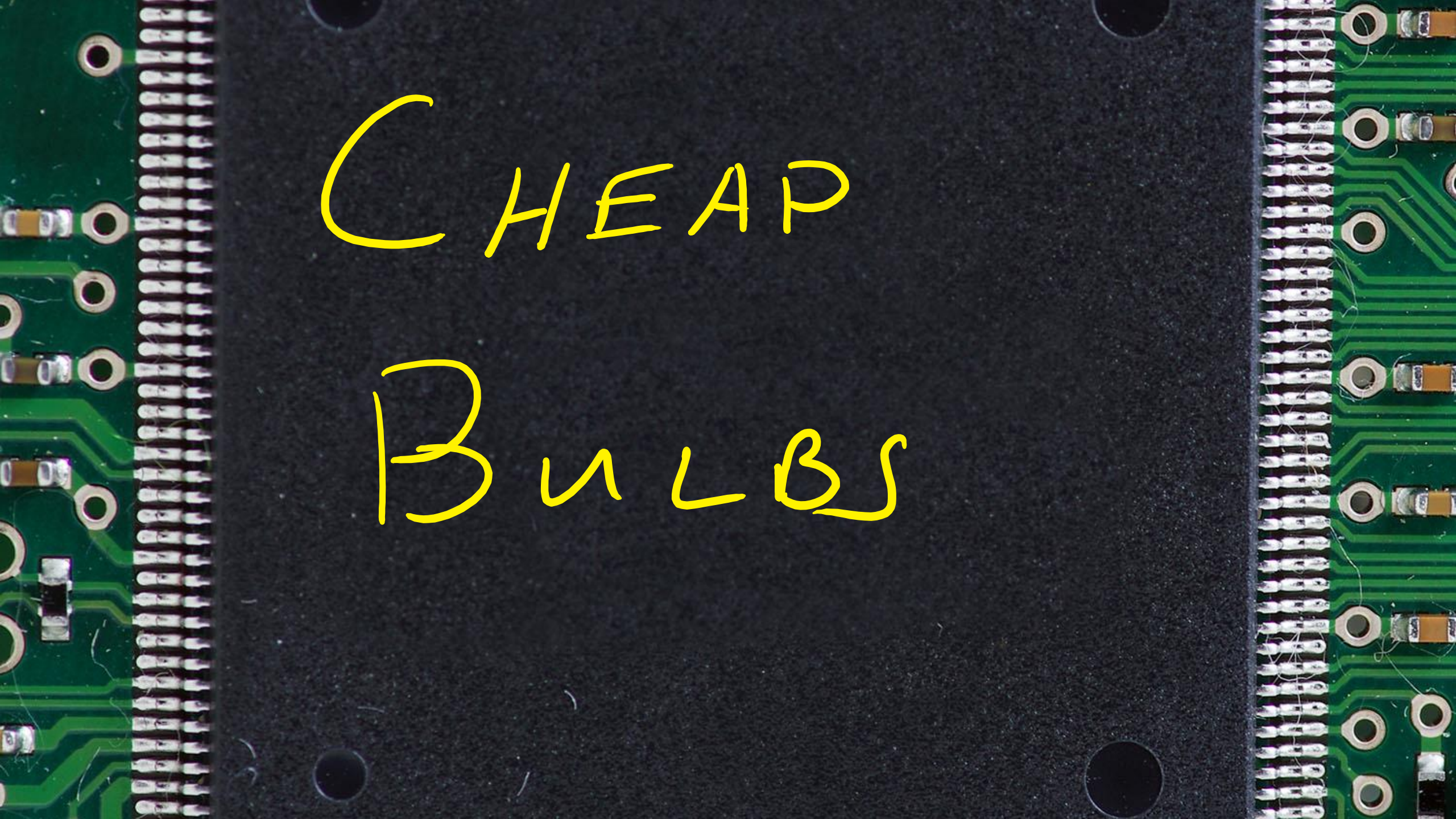
Work by Eyal Ronen et al (NOT Colin O'Flynn)

LIGHT BULB MALWARE OR (BRICKS)

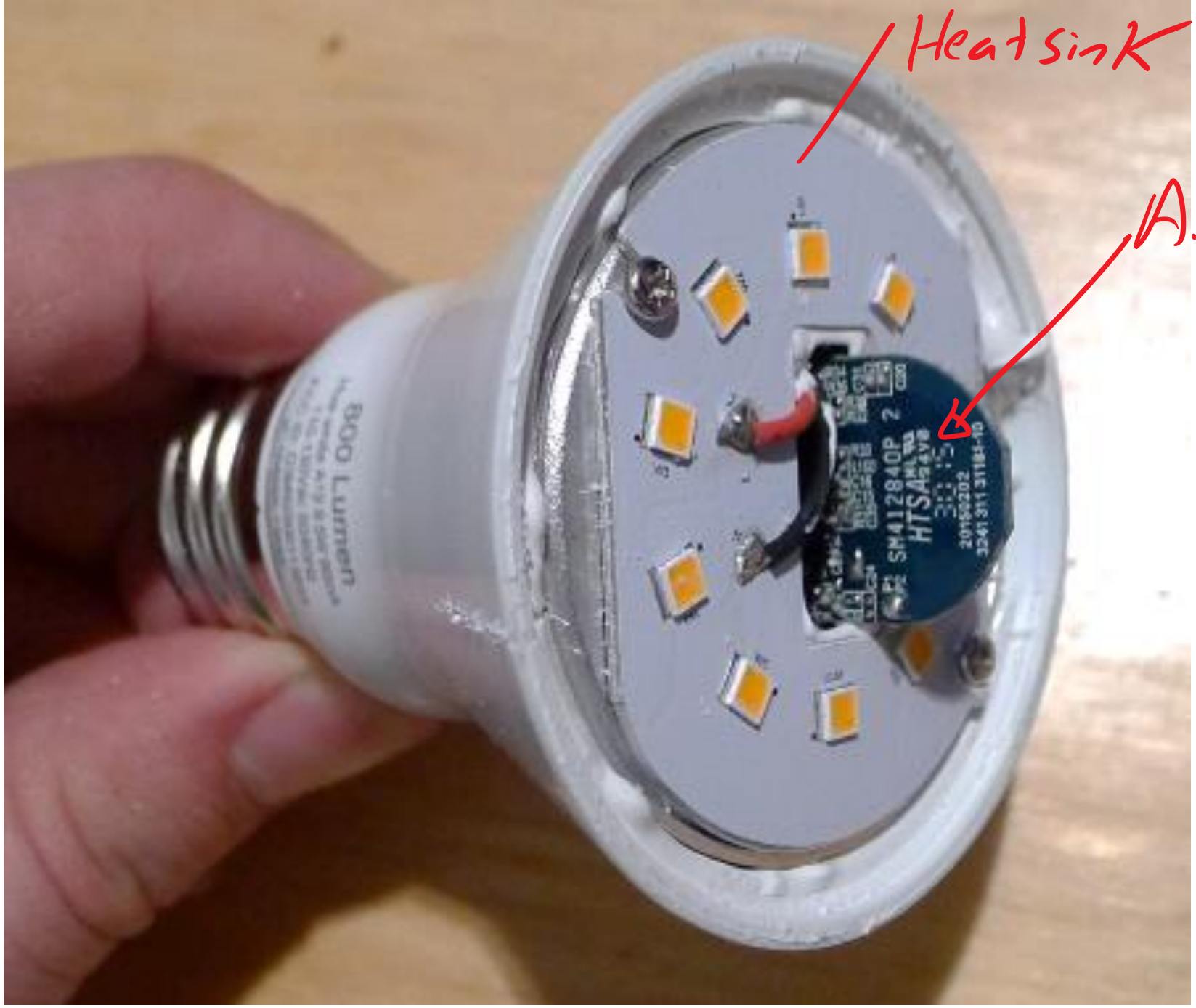


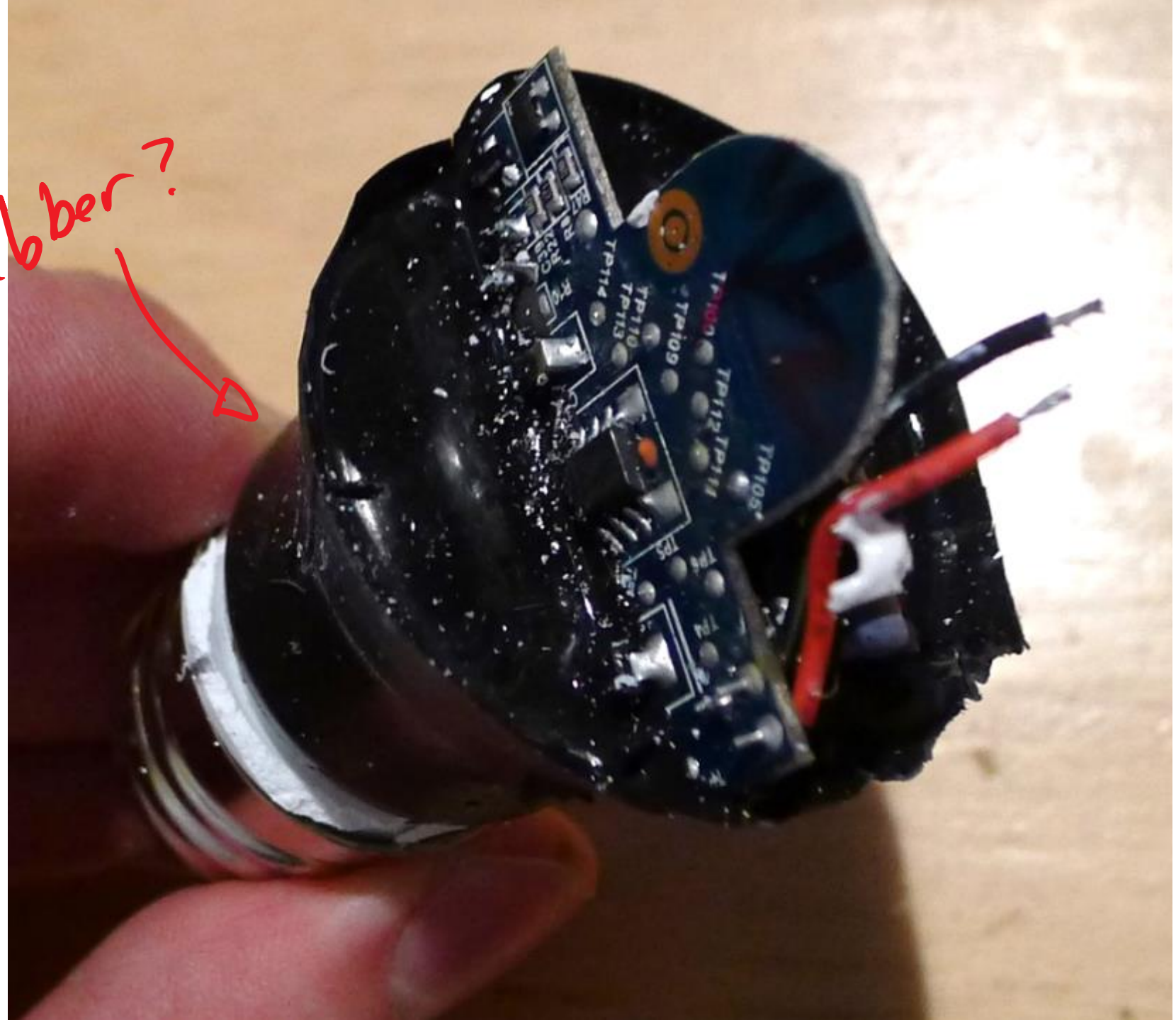
LIGHTBULB MALWARE

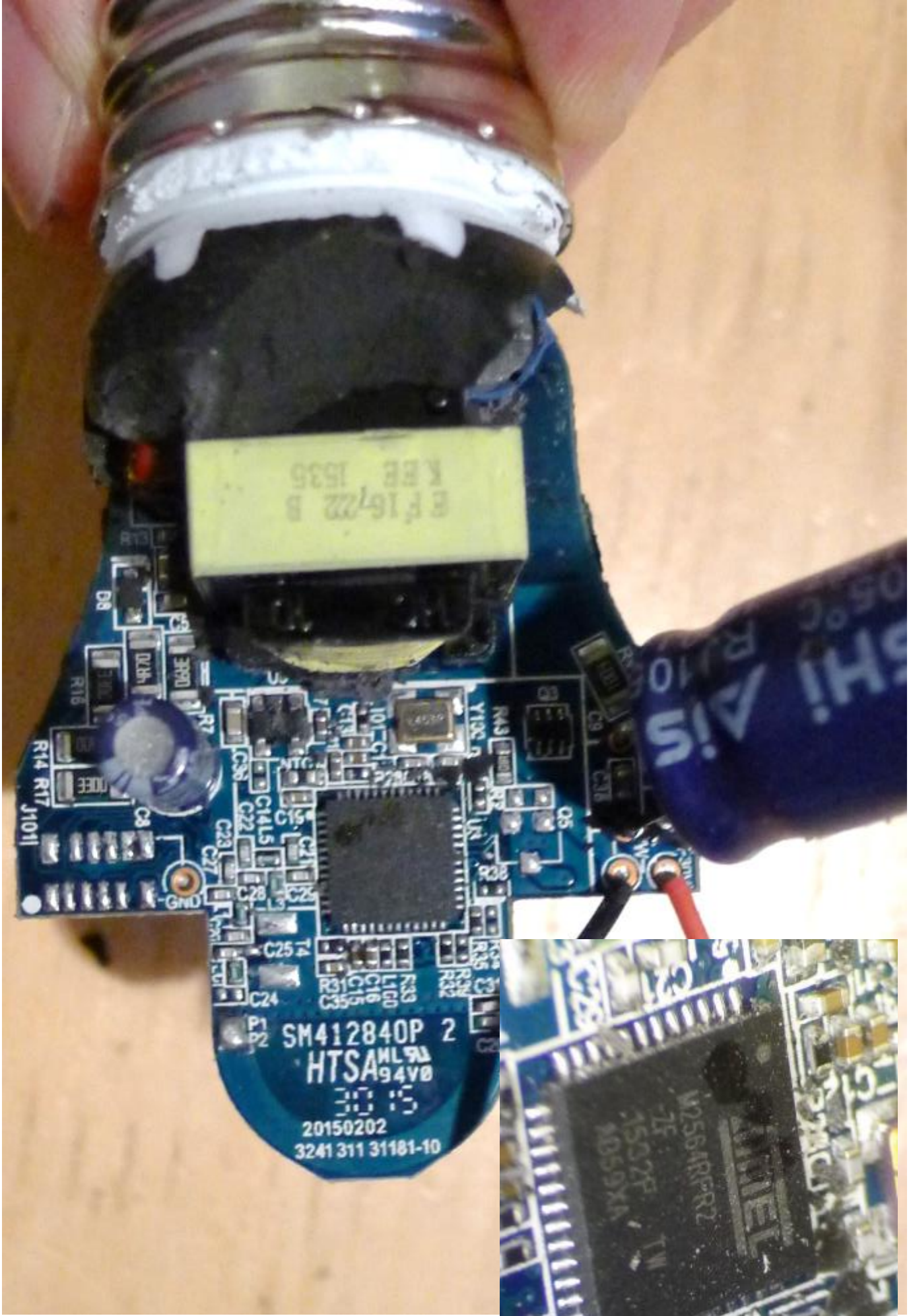
1) ZLL key leaked. We know it's possible to "steal" bulbs.
2) Custom FW on bulbs could turn bulb into "bridge" that searchs for & steals nearby bulbs.
3) If could cause other bulbs to perform OTA FW update → WORM

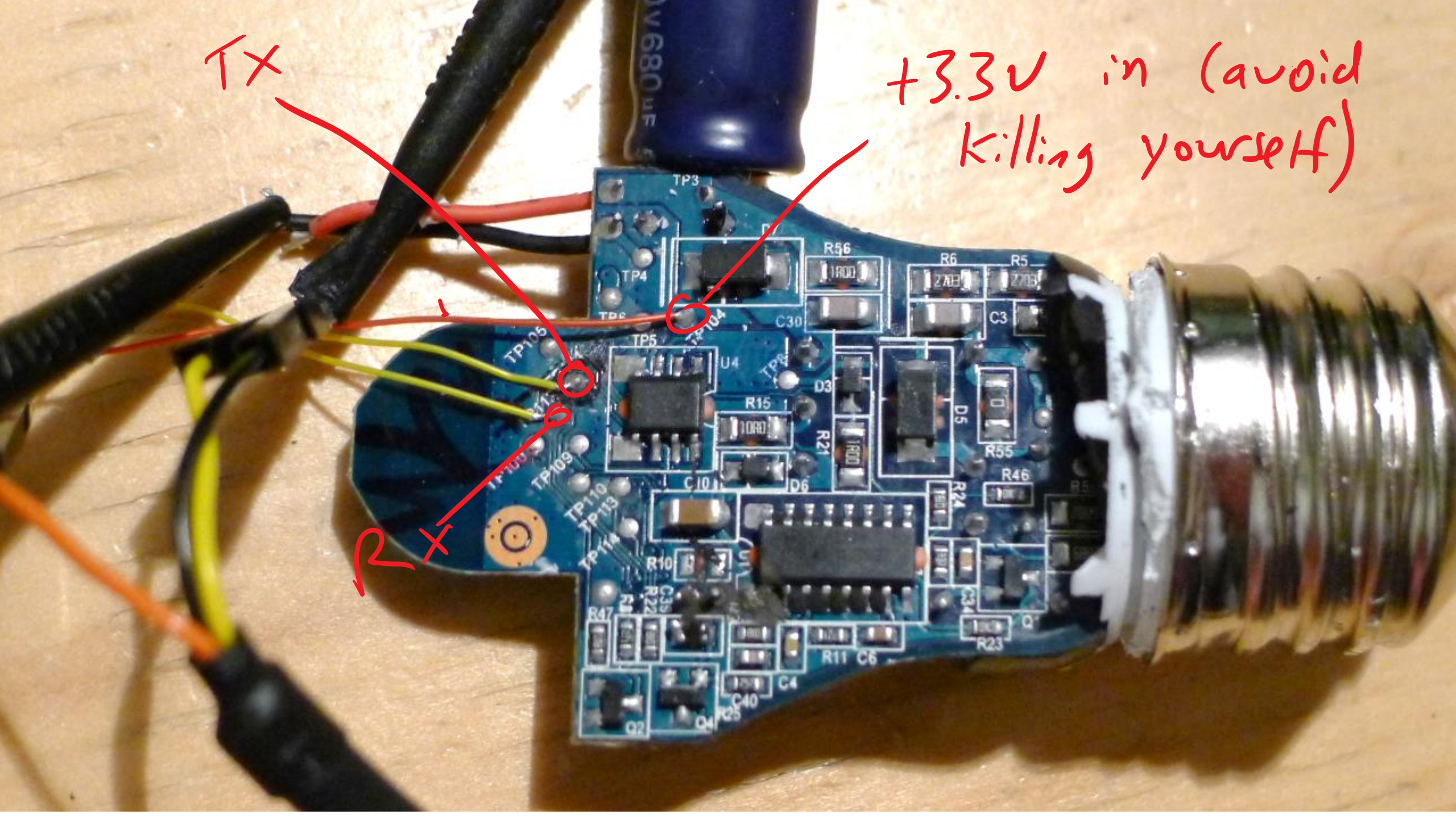

CHEAP
BULBS

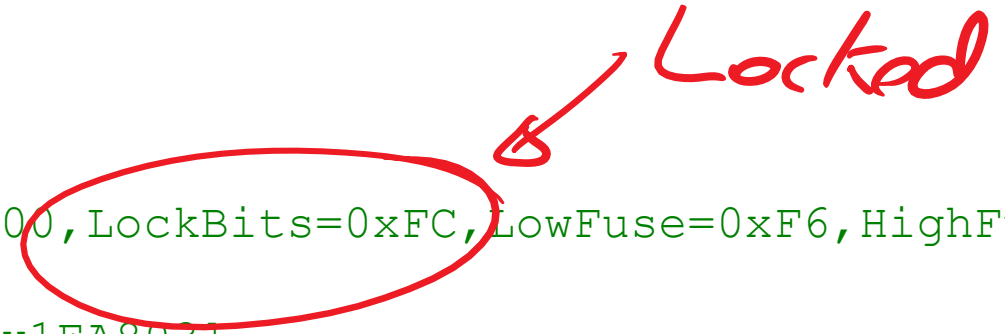

800 Lumen
Hue white A19 9.5W 90mA
110-130Vac 50/60Hz
FCC ID: O3M9290011369X
IC:10469A-1369X
Model: 9290011369


Heatsink
Antenna
800 Lumen

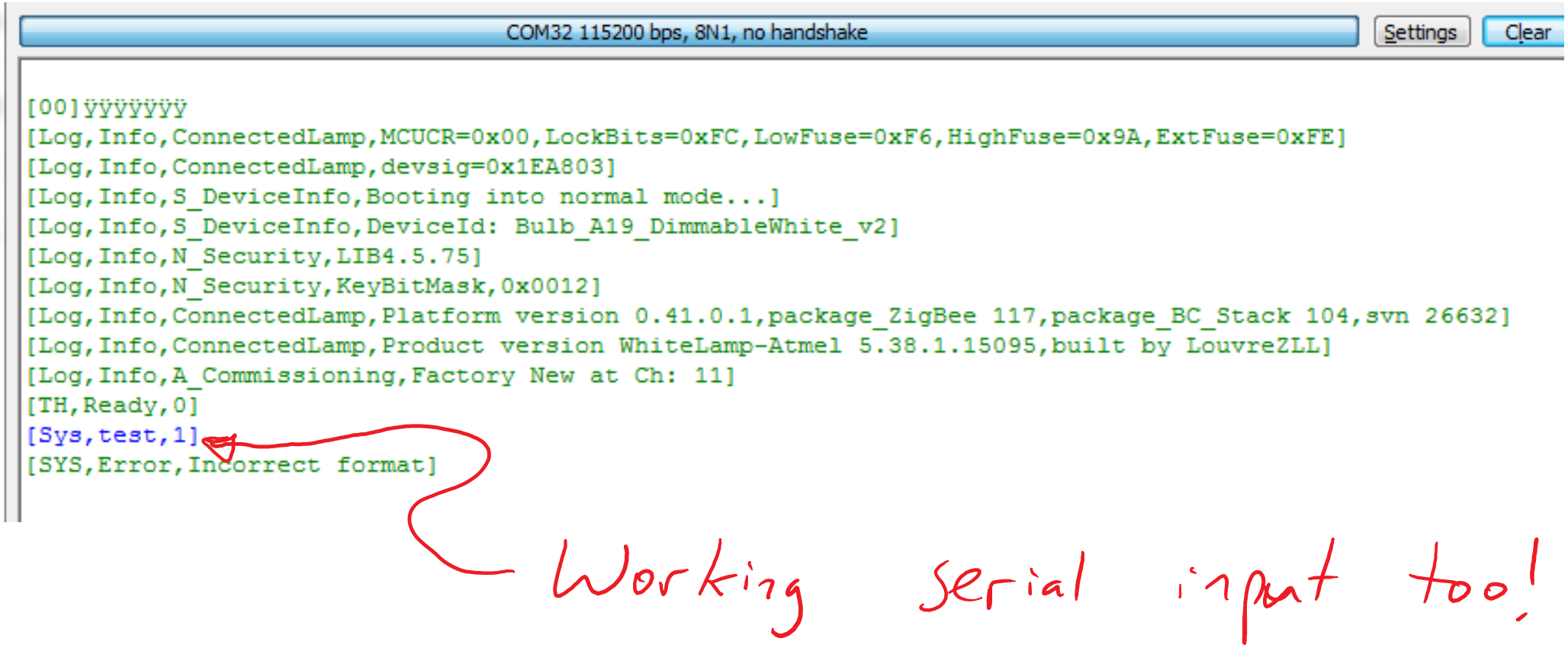

Rubber?

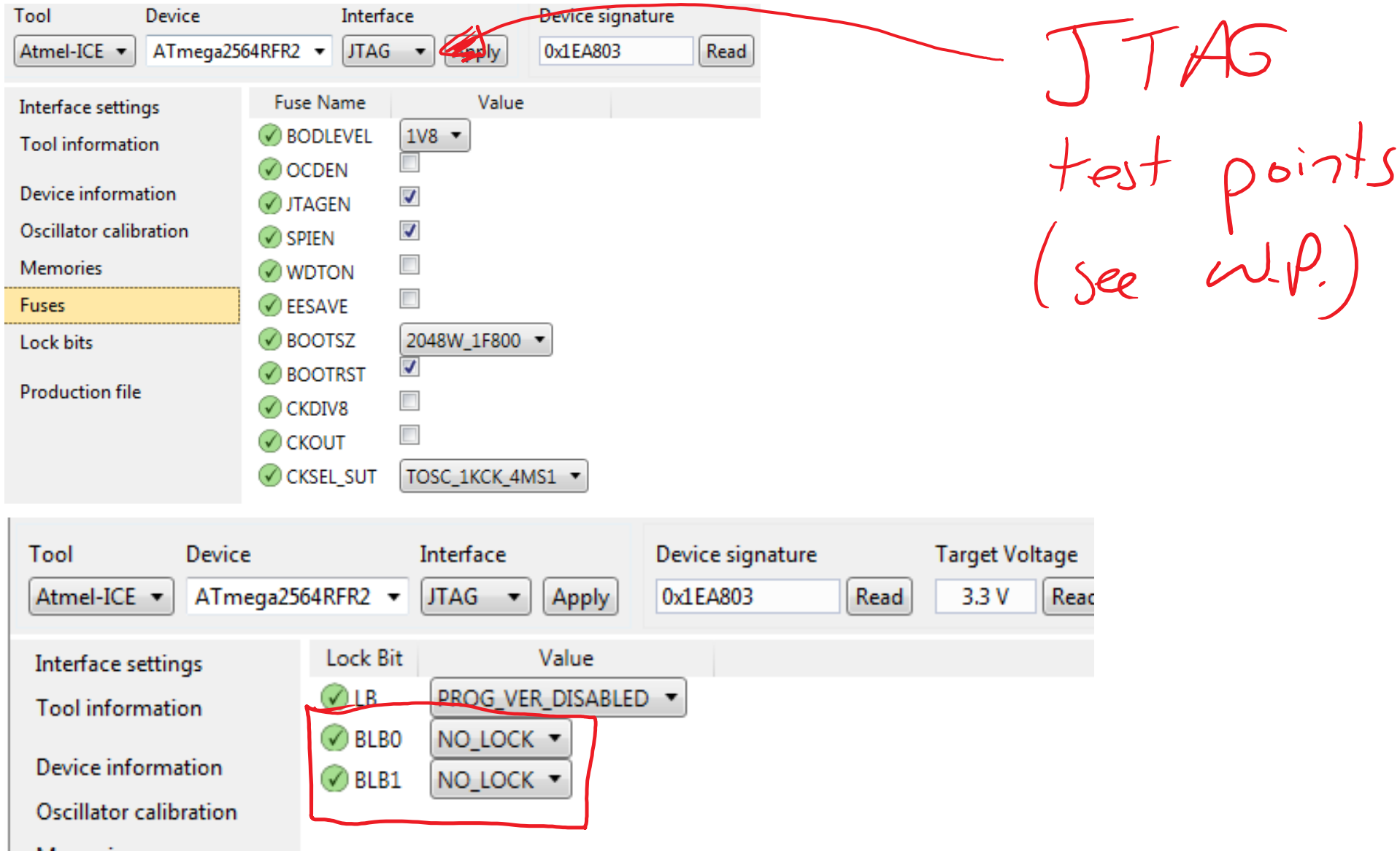

R14 R17
J1011
GND
C25
C24
P1
P2
SM412840P 2
HTSA
20150202
3241 311 31181-10


TX
+3.3V in (avoid killing yourself)
RX
TP3
TP4
R56
R6
R5
C30
C3
TP5
U4
D3
R15
R21
D5
R55
C10
D6
R46
R24
R10
R47
R11
C6
C4
C40
Q2
Q4
Q1
R23




```
[Log,Info,ConnectedLamp,MCUCR=0x00,LockBits=0xFC,LowFuse=0xF6,HighFuse=0x9
A,ExtFuse=0xFE]
[Log,Info,ConnectedLamp,devsig=0x1EA803]
[Log,Info,S_DeviceInfo,Booting into normal mode...]
[Log,Info,S_DeviceInfo,DeviceId: Bulb_A19_DimmableWhite_v2]
[Log,Info,N_Security,LIB4.5.75]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,ConnectedLamp,Platform version 0.41.0.1,package_ZigBee
117,package_BC_Stack 104,svn 26632]
[Log,Info,ConnectedLamp,Product version WhiteLamp-Atmel 5.38.1.15095,built
by LouvreZLL]
[Log,Info,A_Commissioning,Factory New at Ch: 11]
[TH,Ready,0]
```


COM32 115200 bps, 8N1, no handshake
Settings
Clear
[00]ÿÿÿÿÿÿÿ
[Log,Info,ConnectedLamp,MCUCR=0x00,LockBits=0xFC,LowFuse=0xF6,HighFuse=0x9A,ExtFuse=0xFE]
[Log,Info,ConnectedLamp,devsig=0x1EA803]
[Log,Info,S_DeviceInfo,Booting into normal mode...]
[Log,Info,S_DeviceInfo,DeviceId: Bulb_A19_DimmableWhite_v2]
[Log,Info,N_Security,LIB4.5.75]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,ConnectedLamp,Platform version 0.41.0.1,package_ZigBee 117,package_BC_Stack 104,svn 26632]
[Log,Info,ConnectedLamp,Product version WhiteLamp-Atmel 5.38.1.15095,built by LouvreZLL]
[Log,Info,A_Commissioning,Factory New at Ch: 11]
[TH,Ready,0]
[Sys,test,1]
[SYS,Error,Incorrect format]
Working serial input too!



See white-paper for JTAG pin-out connections.

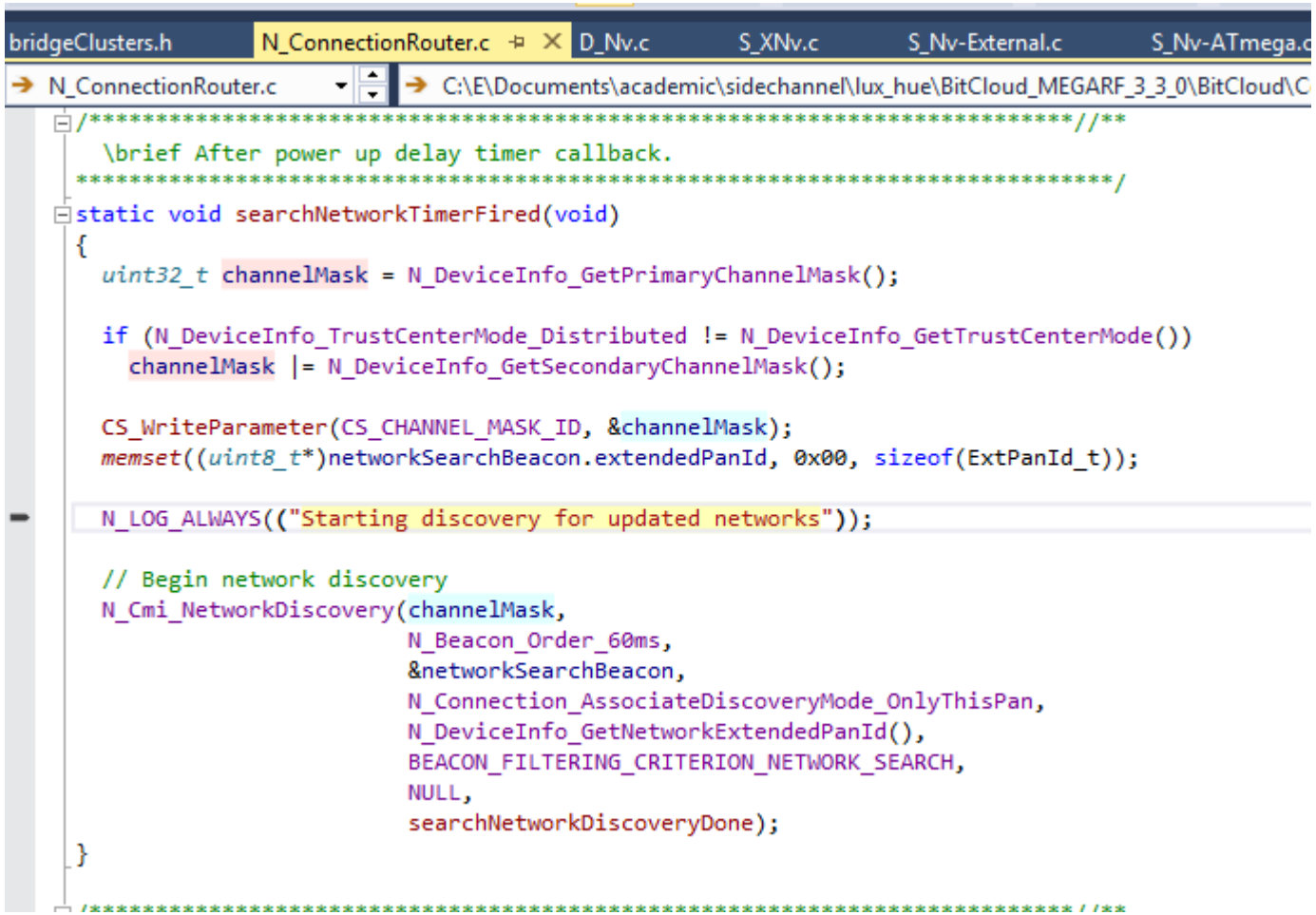
a.Hold SPI line low, notice ASSERT printed matches same name-types used (NVs)

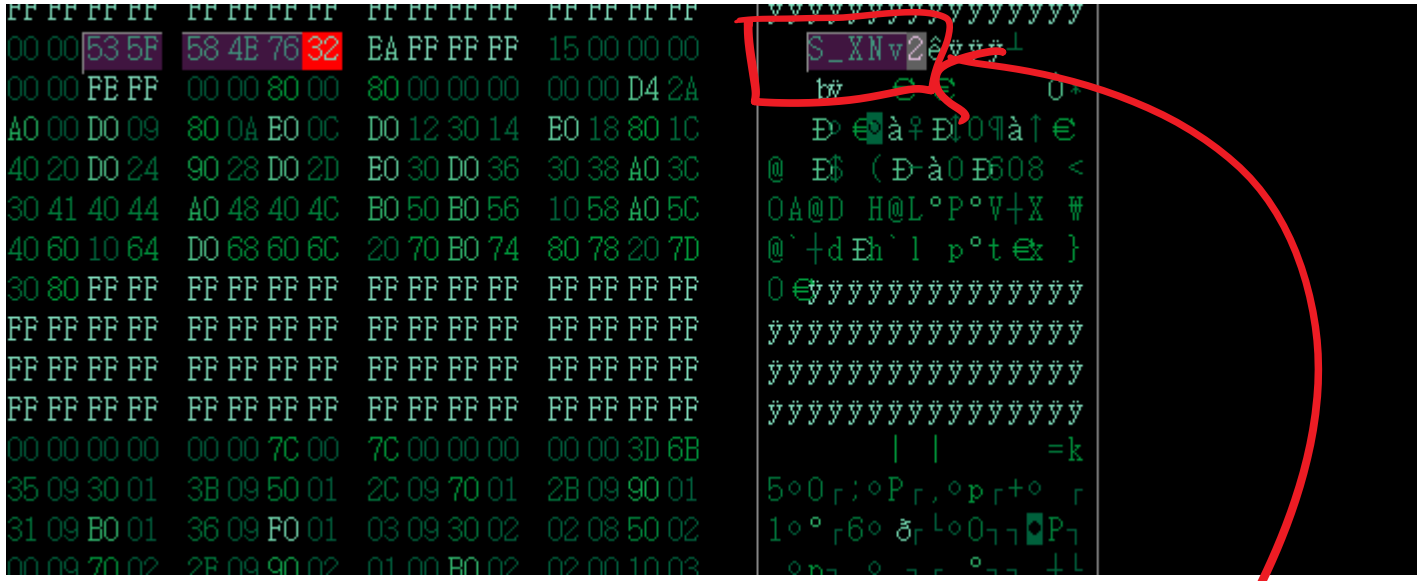
b.Can find same print statements

[TH,Ready,0]

[Log,Info,N_Connection,Starting discovery for updated networks]

[Log,Info,N_Connection,Discovery for updated networks completed]

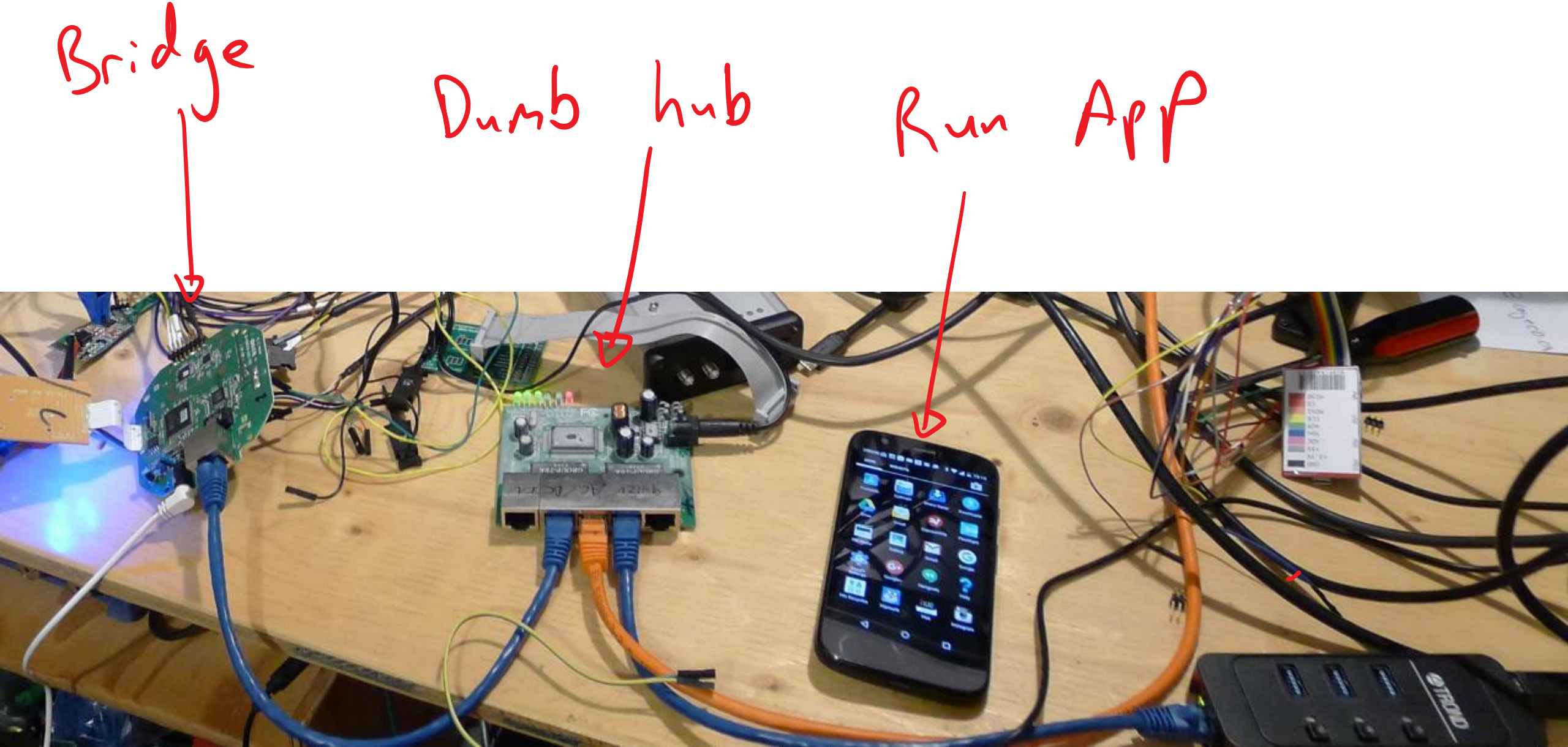


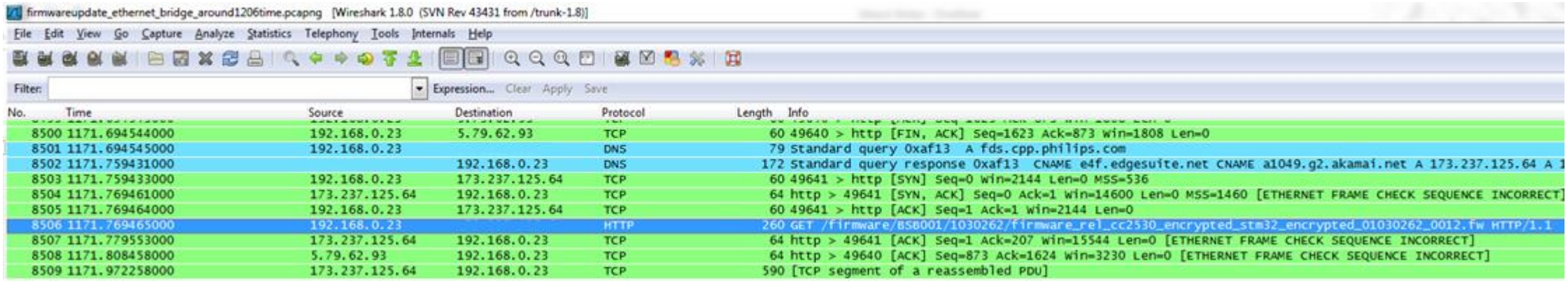



```c
/** 16 byte sector header used in flash located at the start of the active sector. */
typedef struct SectorHeader_t
{
    /** Is this sector active. Written with 0x0000 at the end of the compact operation. */
    uint16_t isActive;
    /** Signature to detect valid sectors. Must have the value "S_XNv2". */
    uint8_t signature[6];
    /** Counter, decreased each time a new sector becomes the active sector. */
    uint32_t sequenceNumber;
    /** Parity bits for the sequenceNumber field = sequenceNumber ^ 0xFFFFFFFFuL. */
    uint32_t sequenceParity;
} SectorHeader_t;
```


BRIDGE
1.0

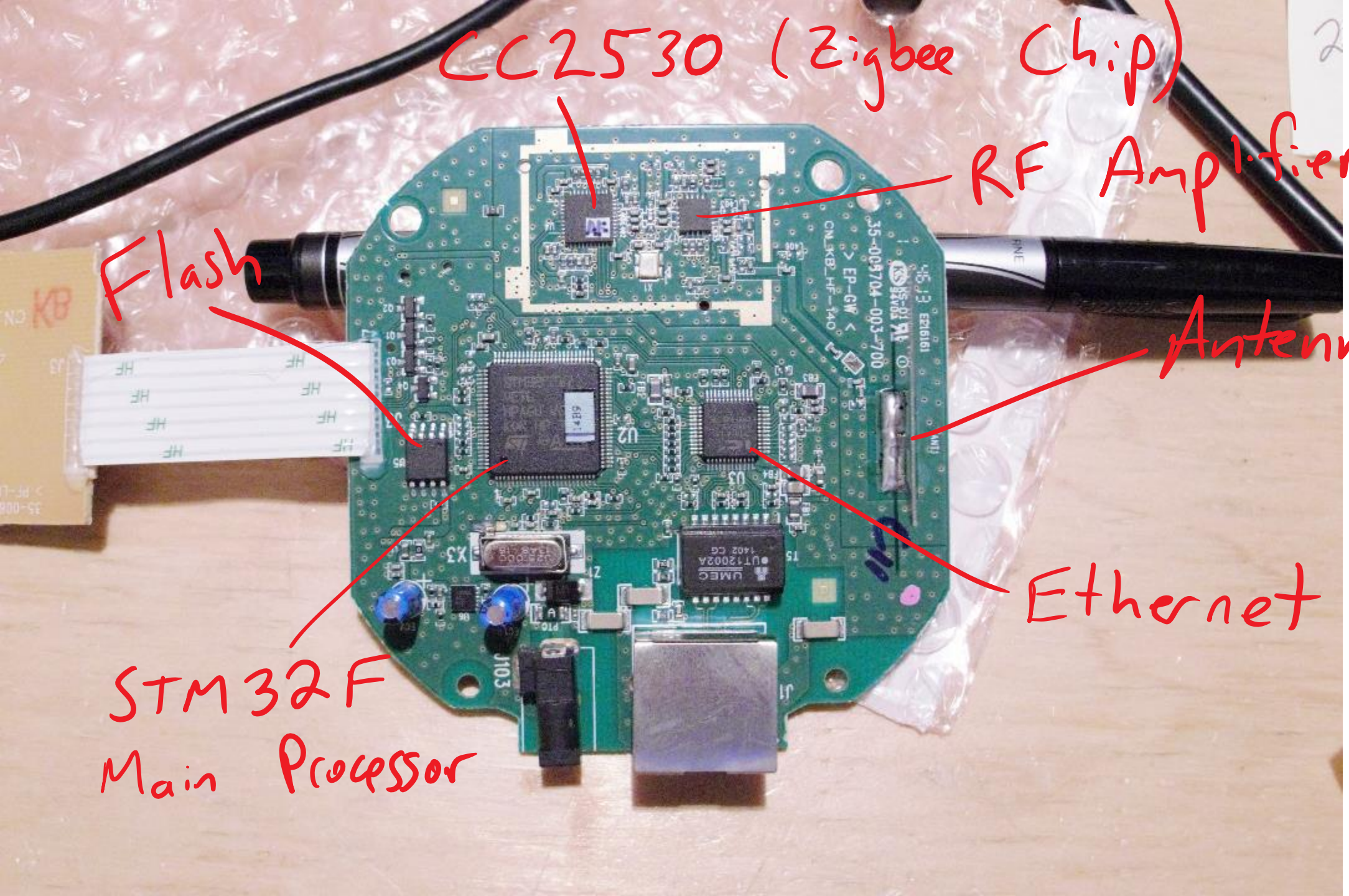
BRIDGE 1.0 HACKING





http://xxx/firmware/HUE0100/66013452/ConnectedLamp-Target_0012_13452_8D.sbl-ota

http://xxx/firmware/BSB001/1030262/firmware_rel_cc2530_encrypted_stm32_encrypted_01030262_0012.fw


CC2530 (Zigbee Chip)
RF Amplifier
Flash
Antenna
Ethernet
STM32F
Main Processor


20706 105C 60V VW-1 -HF-
20706 105C 60V VW-1 -HF-
20706 105C 60V VW-1 -HF-
20706 105C 60V VW-1 -HF-
TP35
TP46
TP6
TP41
TP24
TP25
TP22
TP23
TP26
TP40
TP21
TP19
TP18
TP27
TP34
TP37
TP2
TP7
TP32
TP35
TP5
TP17
TP16
TP20
TP3
TP39
TP4
TP9
TP28
TP31
SW1
TP10
TP29
TP15
TP14
TP419
35-00870-4
> EP-GW <

Output from CC2530

```
[Log,Info,S_DeviceInfo,Booting into normal mode...]
[Log,Info,S_DeviceInfo,DeviceId: IpBridge]
[Log,Info,N_Security,LIB4.4.52]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,A_Bridge,Platform version 0.25.0,package_ZigBee 8720,package_Z_Stack
8720,built by LouvreZLL]
[Log,Info,A_Bridge,Product version 5.7.1,SmartBridge 11393,built by LouvreZLL]
[Bridge,Version,5.7.1,SmartBridge 11393,built by LouvreZLL]
[Bridge,GroupRange,0x5357,0x5367]
[Log,Info,D_Led,dc 16]
[Bridge,NetworkSettings,False,0xB163,26DF52A183D85889,11,0,S=0x0001]
[Log,Info,A_Bridge,NwkAddr: 0x0001, Ch: 11, Pan: 0xB163, NwkUpdId: 0,
ExtPanID:26:DF:52:A1:83:D8:58:89]
[Log,Info,D_Led,dc 16]
[TH,Ready,0]
[Connection,A]
[Connection,GetAddress,L=00:17:88:01:01:07:BF:FC,S=0x0001.0]
[Bridge,StoreGroupRange,0]
[Log,Info,N_ConnectionRouter,Startup network discovery...]
```

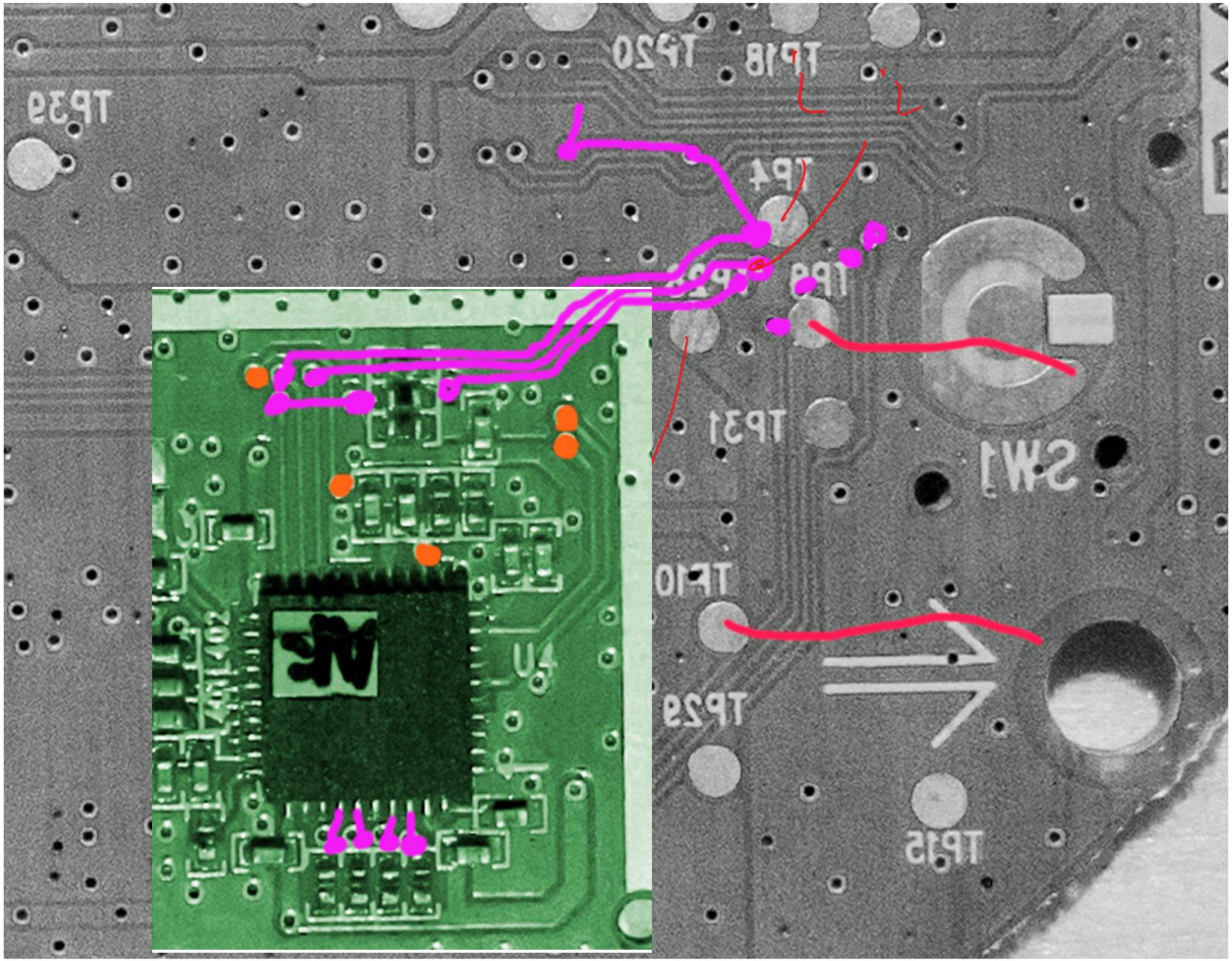
Input to CC2530

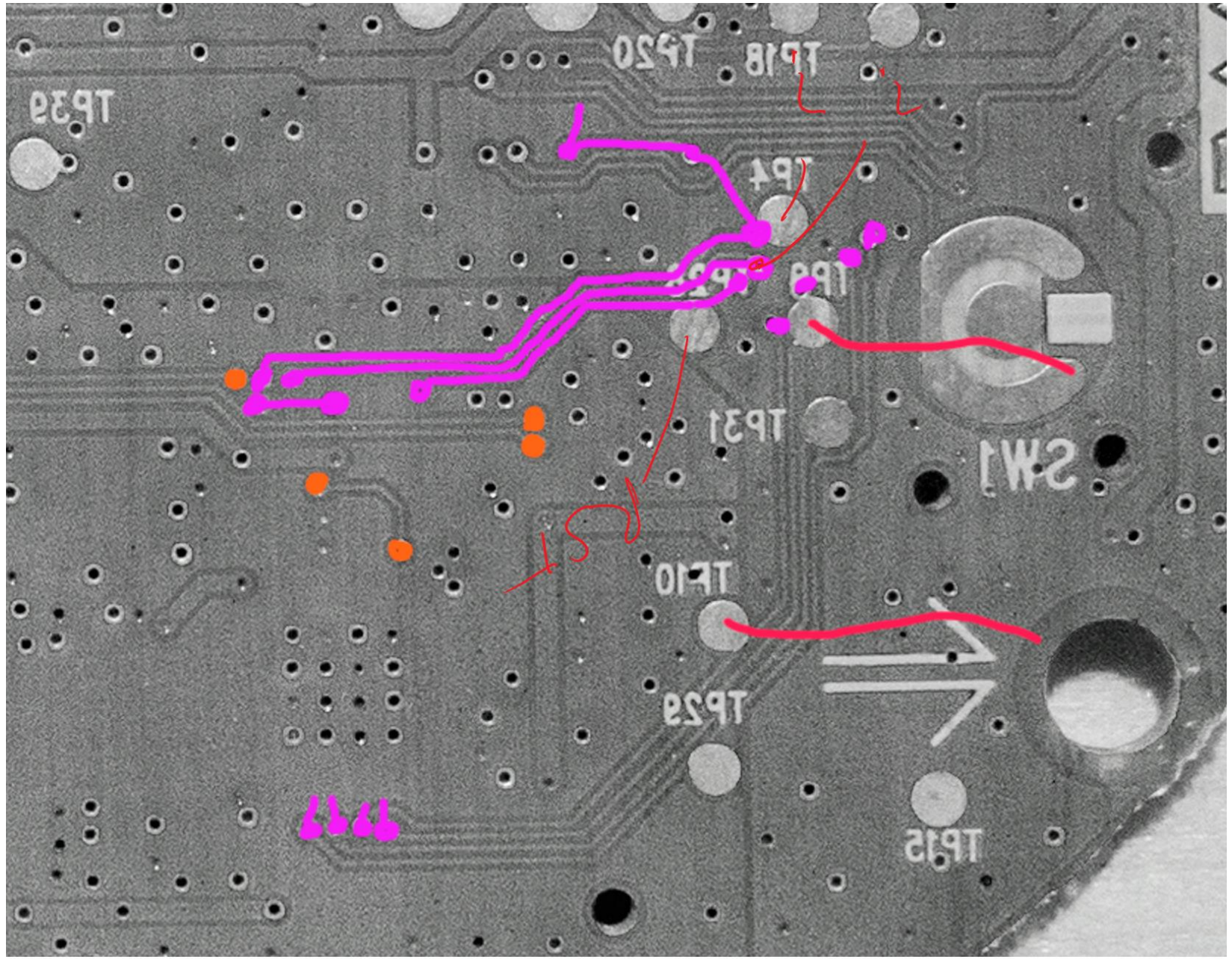
```
[Connection,GetAddress]
[Bridge,StoreGroupRange,0x5357,0x5367]
[Zcl,S,S=0x0002.11,6,0000000000]
[Routing,ClearEntry,1]
[Routing,SendMtoRR,True]
[Zcl,S,S=0x0003.11,6,0001000000]
[Routing,ClearEntry,2]
[Routing,SendMtoRR,True]
[Zcl,S,S=0x0002.11,6,0002000000]
[Zcl,S,S=0x0003.11,6,0003000000]
[Zcl,S,S=0x0002.11,6,0004000000]
```

Bypass ZLL Key?

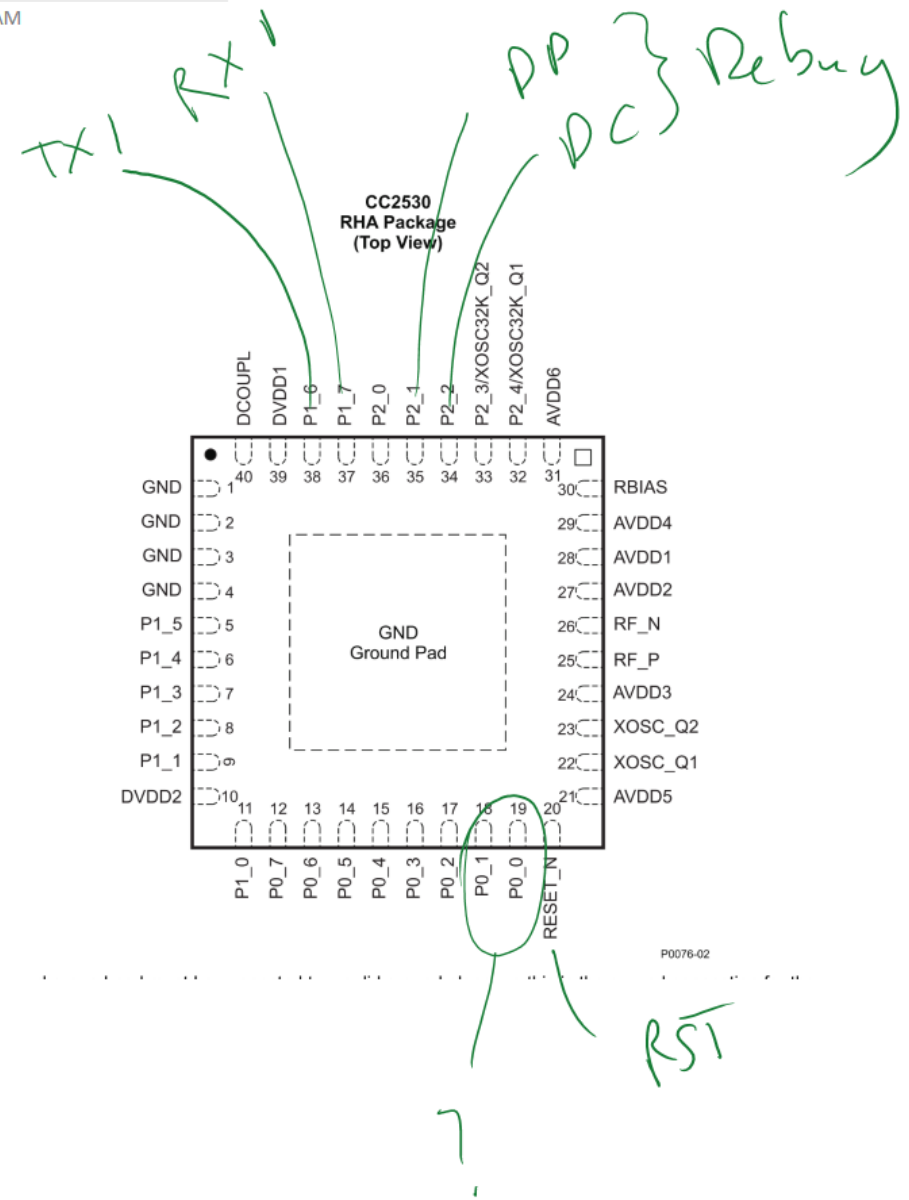
```
[Zcl,S,S=0x0004.11,6,001a000000,64]
[Zcl,S,S=0x0004.11,8,001b000000,64]
[Zcl,S,S=0x0004.11,768,001c000100004002400300040007000800,64]
[Zcl,S,S=0x0004.11,25,190d05000b1000018c340042ae0100002b97098fac0320f2f31e3c8fd035a34097da5018feb50a2e8b40d3678aa57c866a47122020a3a86220a25c93,64]
[Zcl,S,S=0x0004.11,25,190e05000b1000018c340042d90100002b2d3d4e5d25c0622af60856c62900d59f71b104541e744b3657ebc32286f3e635474145d3189f7deca60cd9,64]
[Zcl,S,S=0x0003.11,6,001d000000,64]
[Zcl,S,S=0x0004.11,25,190f05000b1000018c340042040200002b92c84eb5d02416e5153d8aa6a944b0dd7c9796547fa4f63793ea06c100f2c3293c87a425cd5279a8765d3d,64]
[Zcl,S,S=0x0003.11,8,001e000000,64]
[Zcl,S,S=0x0004.11,25,191005000b1000018c3400422f0200002ba0b3d5e50a0e550e48f25a6125d1aea4fca962453c7f718f05ec20c7875e799ae71b45cd7fc74b3e436094,64]
[Zcl,S,S=0x0004.11,25,191105000b1000018c3400425a0200002b2956b4f0014e7f7aa0ba92c6cb8ed7ddd6d67c114bd4396d5e03f65105ab62da87dac1c7d344e73ea4c901,64]
[Zcl,S,S=0x0002.11,6,001f000000,64]
[Zcl,S,S=0x0004.11,25,191205000b1000018c340042850200002be080f0a5152a9d4c0f7eed933a2a3262474106f57b2947d1c44121c326e1c8bfbaea0a925ed58e5a9290a1,64]
3,64]
a691ef28438fee5be91305000b1000018c340042b00200002b965b229d29d5cf2c0f7eed933a2a3262474106f57b2947d1c44121c326e1c8bfbaea0a925ed58e5a9290a1,64]
[Zcl,S,S=0x0004.11,25,191405000b1000018c340042db0200002ba14b4f7686df97989d0371c2c4358733cd9e9361bc90de747f9ec249c2fe86b90f2430595cc5ba87bde7c0,64]
[Routing,SendMtoRR,True]
[Zcl,S,S=0x0004.11,25,191505000b1000018c340042060300002be67a19318a005a40204ccbc012695198270 9f080f5806e33d478efd8dcda9e79303ad662ddcfa822316b03,64]
```

B3 D5 E5 0A	0E 55 0E 48	F2 5A 61 25	D1 AE A4 FC	³ÕåUHòZa%Ñ®¤ü
A9 62 45 3C	7F 71 8F 05	BC 20 C7 87	5E 79 9A	©bE<q ¼ Ç^yš
1B 45 CD 7E	C7 4E 3E 43	60 94 29 56	B4 F0 01	←EÍ~ÇK>C`")V´ð
77 7A A0 BA	92 C6 CB 8E	D7 DD D6 D6	7C 11 4B D4	wz º'ÆËŽ×ÝÖÖ|◀KÔ
39 6D 5E 03	F0 51 05 AB	62 DA 87 DA	C1 C7 D3 44	9m^ðQ«bÚ‡ÚÁÇÓD
E7 3E A4 C9	01 B0 80 F0	A5 15 2A 9D	4C 0F 7E ED	ç>¤É°€ð¥*L~í
93 3A 2A 32	62 47 41 06	F5 7B 29 47	D1 C4 41 21	"*2bGA-õ{)GÑÄA!
C3 26 E1 C8	BF BA EA 0A	92 5E D5 8E	5A 92 90 A1	Ã&áÈ¿ºê'^ÕŽZ' ¡
96 5B 22 9D	29 D5 CF 2E	C8 1C 7E B2	7D 98 84 96	-[")ÕÏ.È ~²}~„-
DC 79 66 A9	7D 4E E7 41	B2 67 E4 CB	C1 C1 B2 BA	Üyf©}NçA²gäËÁÁ²º

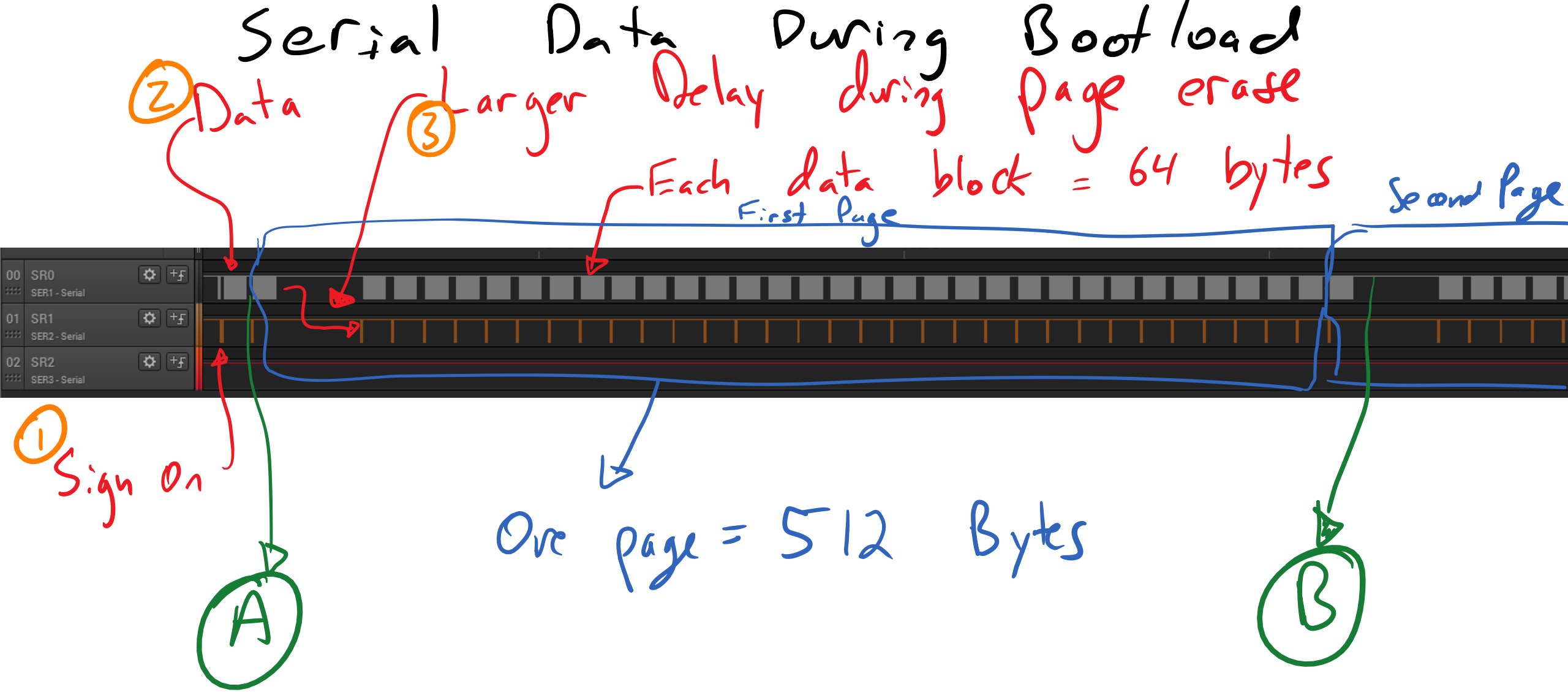

TP39
TP20
TP18
TP4
TP8
TP31
SW1
TP10
TP29
TP15
U4


TP20
TP18
TP39
TP4
TP8
TP31
SW1
TP10
TP29
TP15

:41 AM


TX1
RX1
DP
DC
Reboy
CC2530
RHA Package
(Top View)
DCOUPL
DVDD1
P1_6
P1_7
P2_0
P2_1
P2_2
P2_3/XOSC32K_Q2
P2_4/XOSC32K_Q1
AVDD6
GND
GND
GND
GND
P1_5
P1_4
P1_3
P1_2
P1_1
DVDD2
GND
Ground Pad
RBIAS
AVDD4
AVDD1
AVDD2
RF_N
RF_P
AVDD3
XOSC_Q2
XOSC_Q1
AVDD5
P1_0
P0_7
P0_6
P0_5
P0_4
P0_3
P0_2
P0_1
P0_0
RESET_N
P0076-02
RST

Serial Data During Bootload



Extracting Keys from Second Generation Zigbee Chips

Travis Goodspeed
1933 Black Oak Street
Jefferson City, TN, USA
travis@radiantmachines.com

ABSTRACT

First generation Zigbee chips were SPI slaves with no internal processing beyond cryptographic acceleration. Extracting a key was as simple as spying on the SPI transactions. The second generation chips, typified by the CC2430 from Texas Instruments and the EM250 from Ember, contain both a microcontroller and a radio, making the SPI sniffing attack all but irrelevant. Nevertheless, both chips are vulnerable to local key extraction. This paper describes techniques for doing so, focusing on the CC2430 as the EM250 has no protection against outside access. Recommendations are made for defending CC2430 firmware by using compiler directives to place sensitive information in flash memory, rather than in RAM. All Chipcon radios with 8051 cores released prior to the publication of this paper are expected to be vulnerable.

Keywords

Zigbee, CC2430, EM250, System on a Chip (SoC)

1. GENERATIONS

First generation Zigbee chips, such as the CC2420, were simply digital radios with SPI interfaces and a bit of hardware-accelerated cryptography. They could not run a Zigbee stack themselves, but rather relied upon an external microcontroller cores were added for convenience, not security, as will be explained below.

The third generation of chips will include more powerful microprocessors and–hopefully–a lot more security. The offering from Texas Instruments is the CC430 family, based upon the MSP430X2 processor. Ember will be using the Arm Cortex M3 in its EM300 series. These chips are out of scope for this paper, as they are not yet commercially available. Also, Freescale's line of radios have not yet been examined by the author, but they will be in the near future.

2. CONCERNING THE EM250

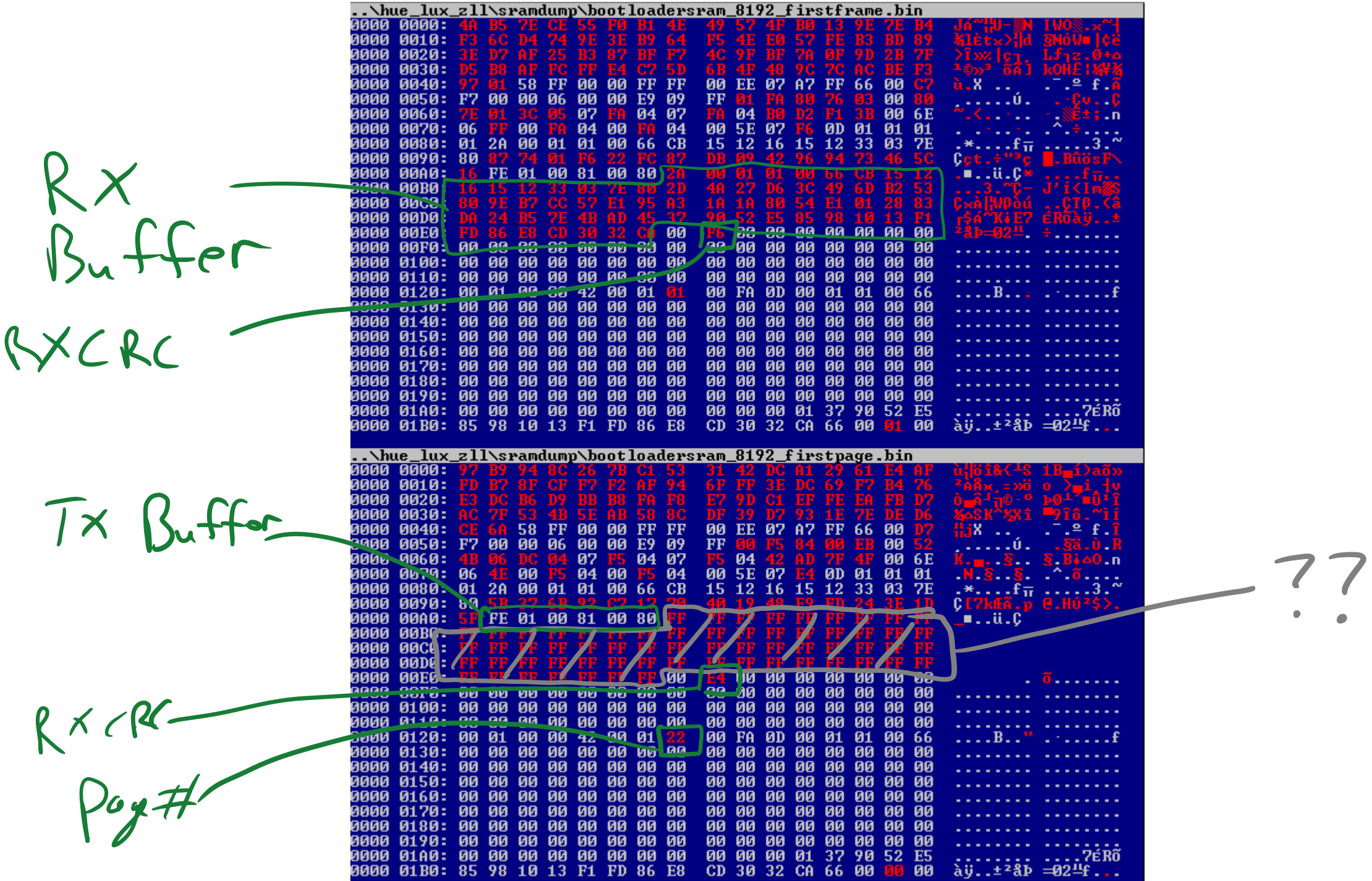
The Ember EM250 contains a 16-bit XAP2b microprocessor from Cambridge Consultants Ltd.[3] Debugging support is provided by that firm's proprietary SIF protocol, with hardware and software available only through Ember. SIF itself is a variant of JTAG.

While the datasheet and various piece of marketing literature claim "The EM250 employs a configurable memory protection scheme usually found on larger microcontrollers.", this refers not to a debugging fuse or bootloader password, but rather to protection from accidental self-corruption of memory. This is in the form of Application/System separation, allowing the EmberZNet stack to defend certain regions

Good Things

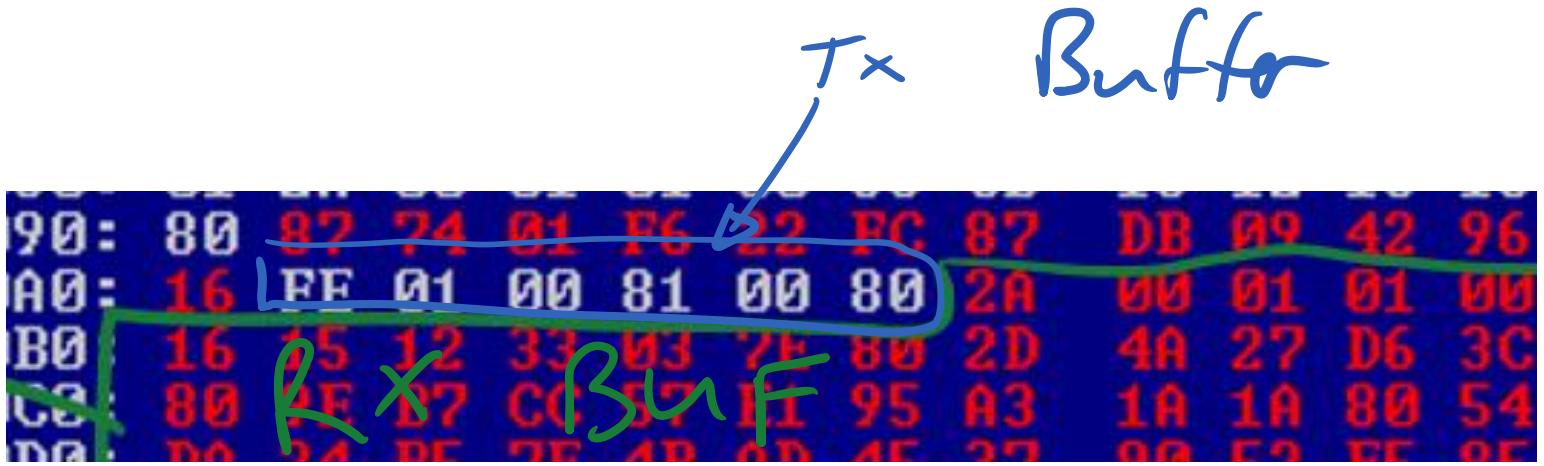
- ZLL master key not in regular SRAM
- Tried AES-128 CBC to decrypt bootloader image, where key = {every possible 16-byte block}

  ↳ No success, key not in SRAM?


..\hue_lux_zll\sramdump\bootloadersram_8192_firstframe.bin
..\hue_lux_zll\sramdump\bootloadersram_8192_firstpage.bin
Rx Buffer
RXCRC
Tx Buffer
RxCRC
Pay#
??

TX BUFFER ATTACK

```
for(uint8 i=0; i < data_to_send; i++){
    uart_write(tx_buf[i]);
}
```

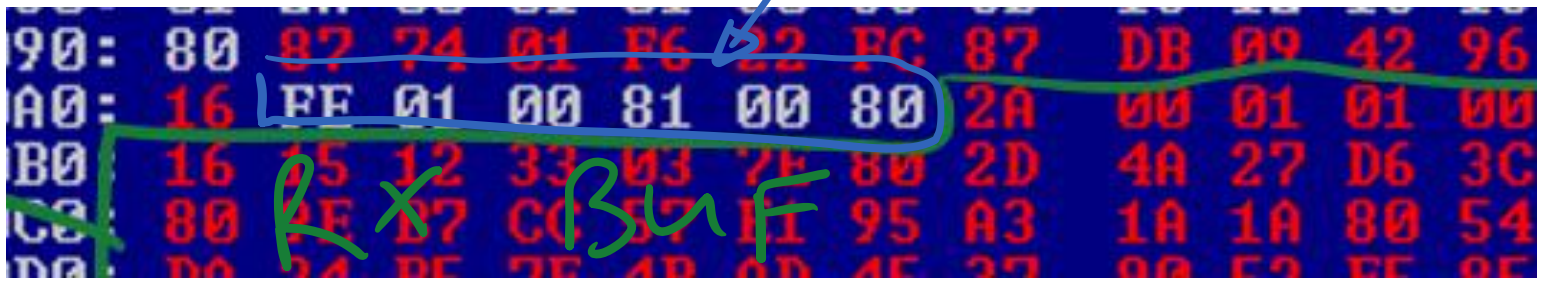


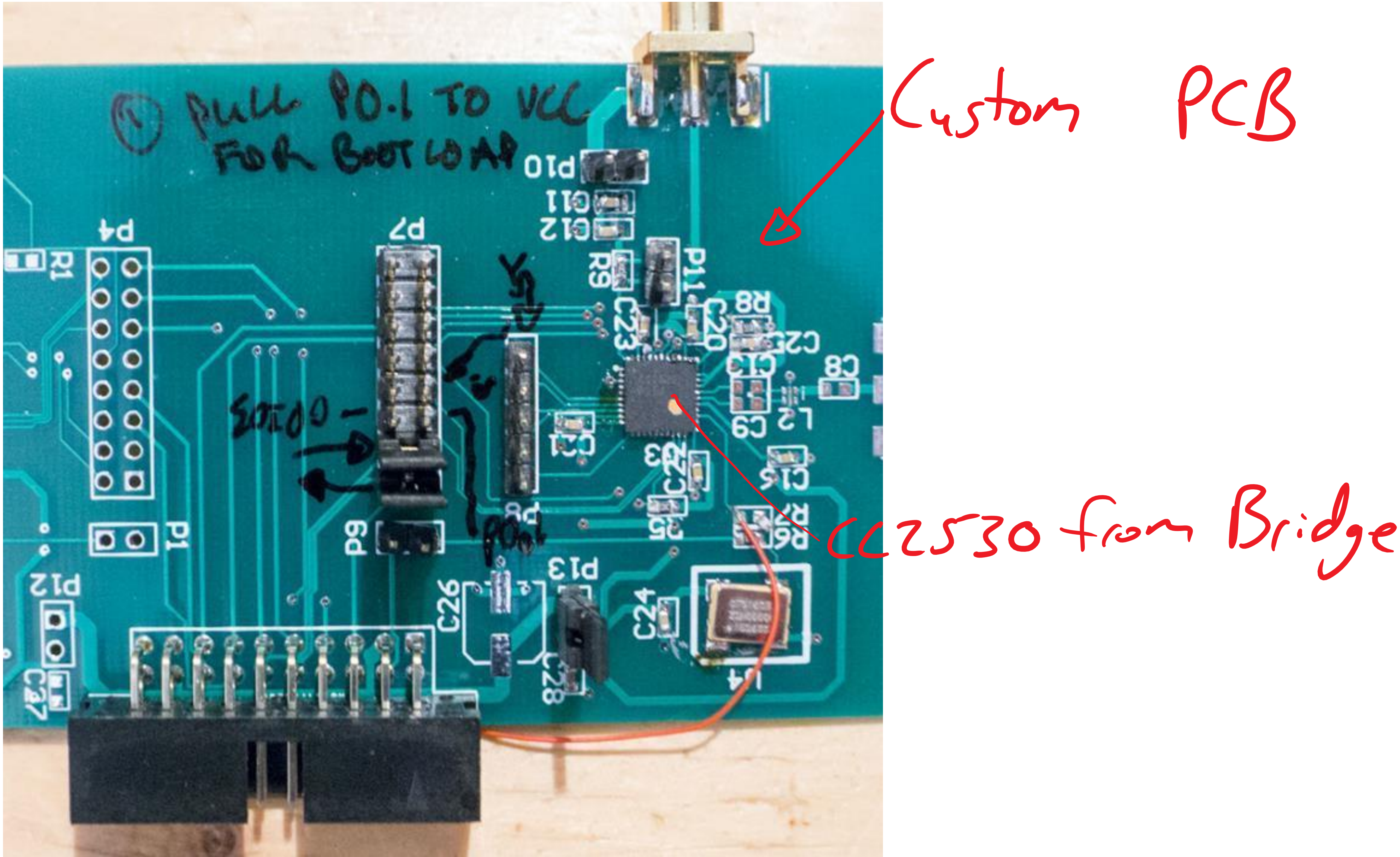
TX Buffer Attack

```
for(uint8 i=0; i < data_to_send; i++){
    uart_write(tx_buf[i]);
}
```

Glitch Attack!

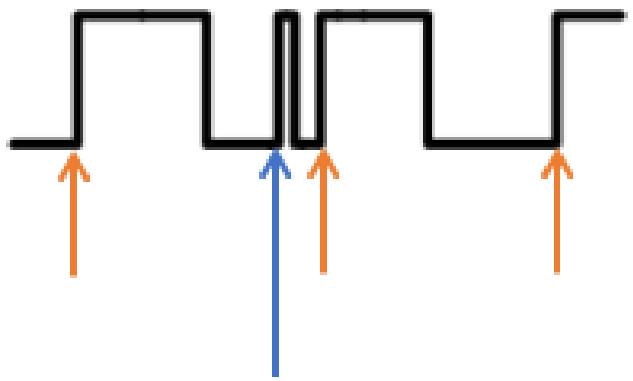
Tx Buffer

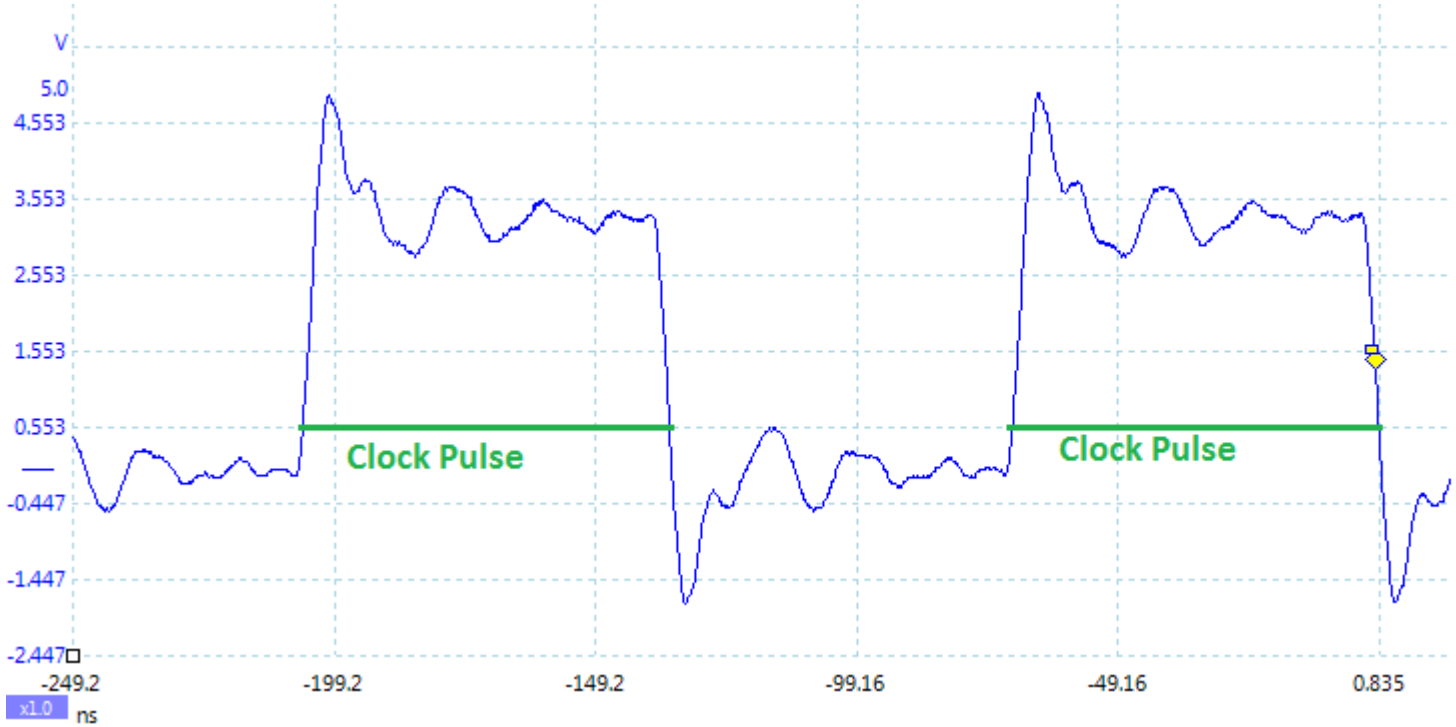




Custom PCB
CC2530 from Bridge
pull P0.1 to VCC FOR BOOTLOAD

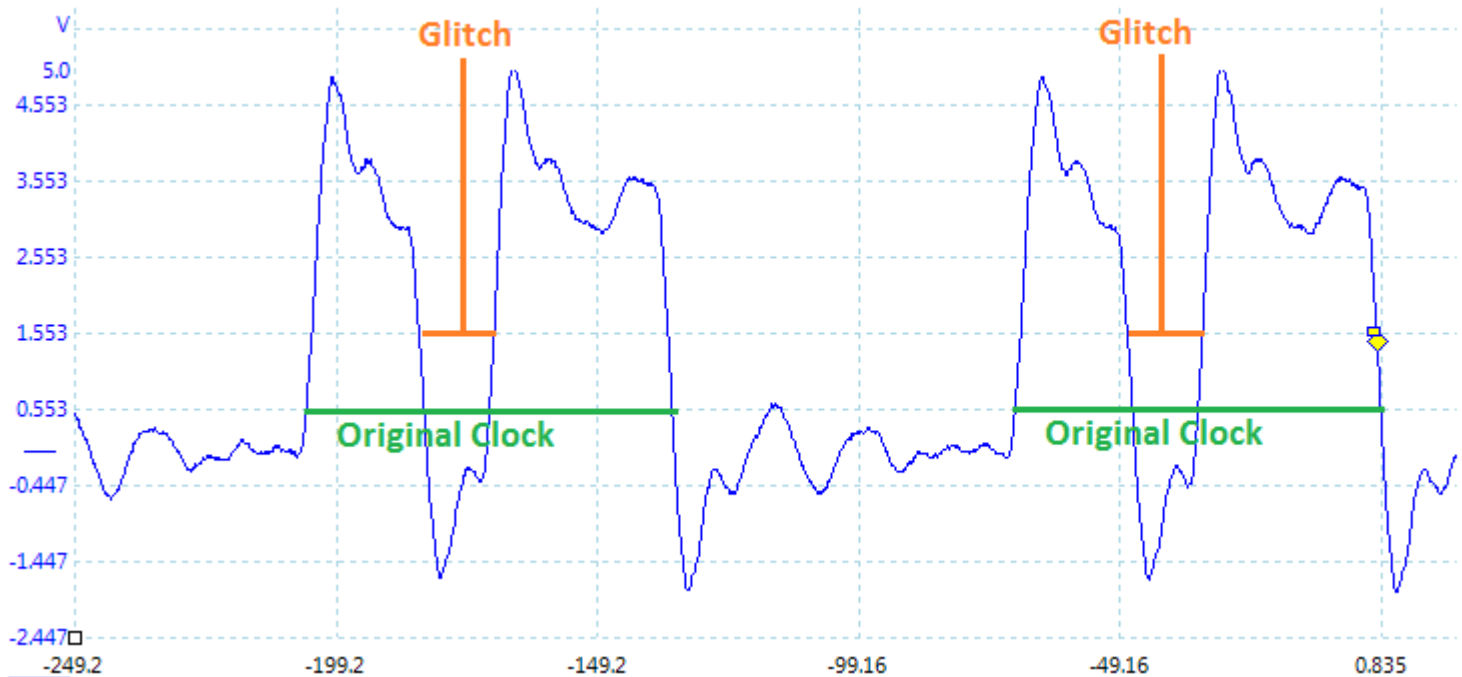
Clock Glitching





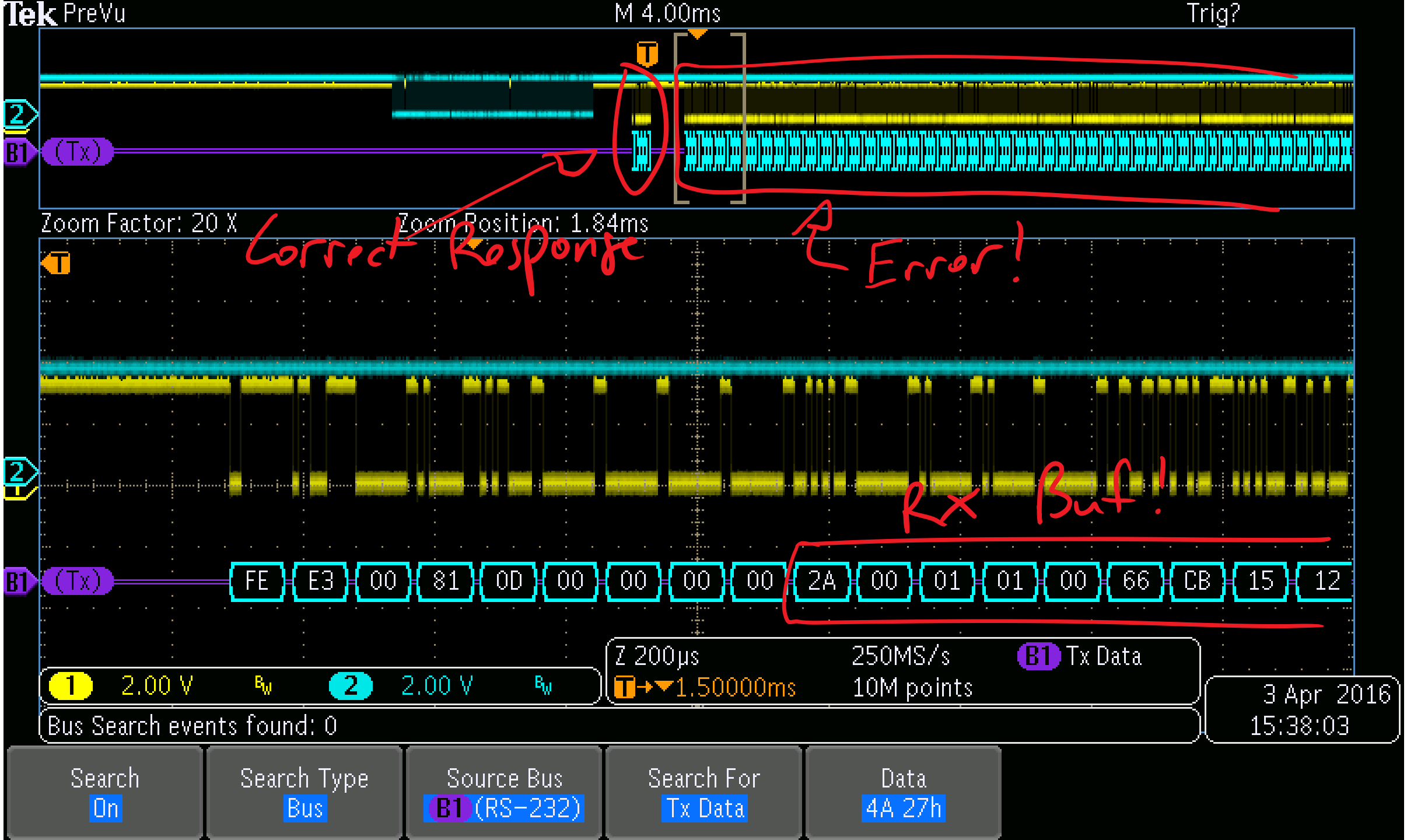
Original Clock

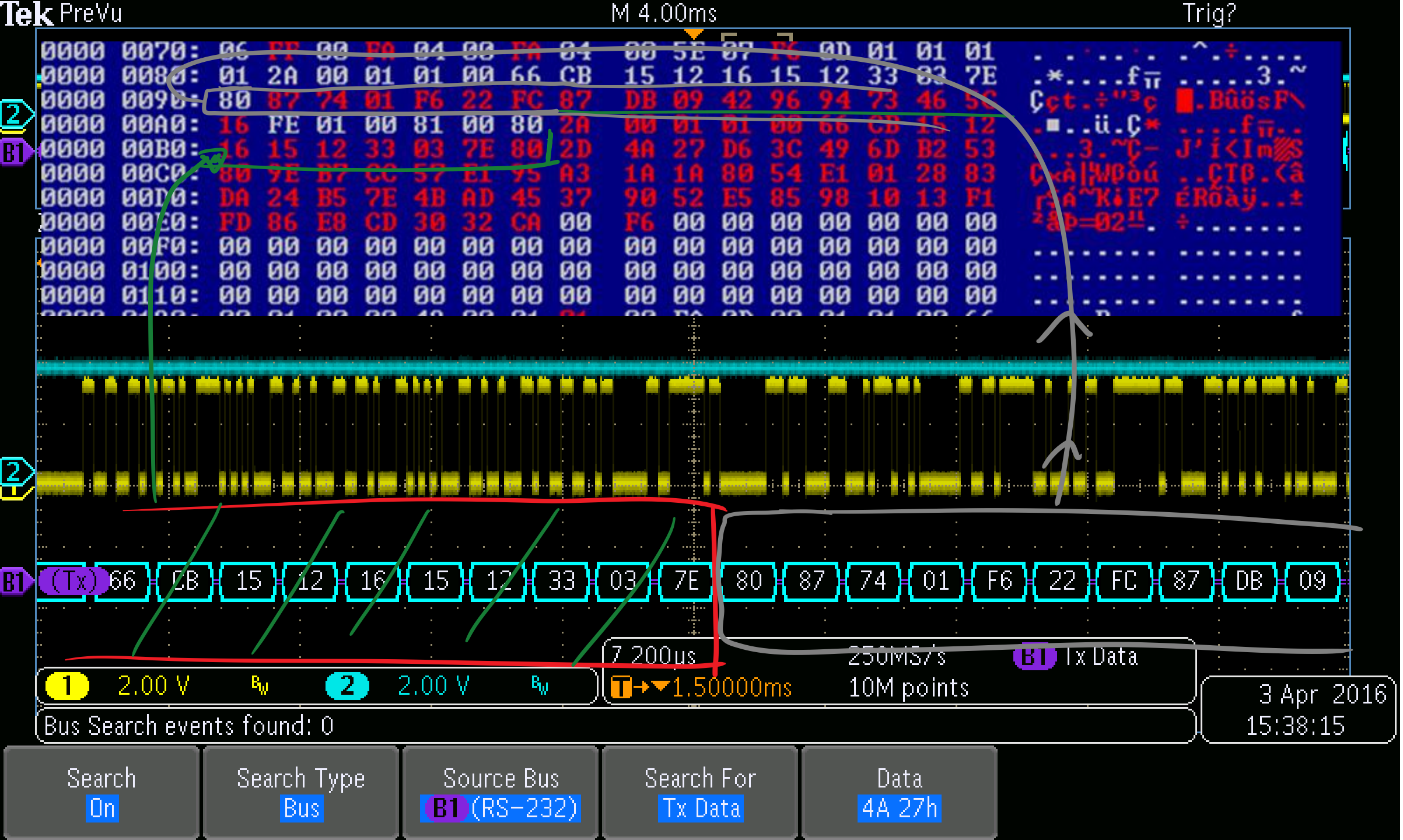
7.37 MHz

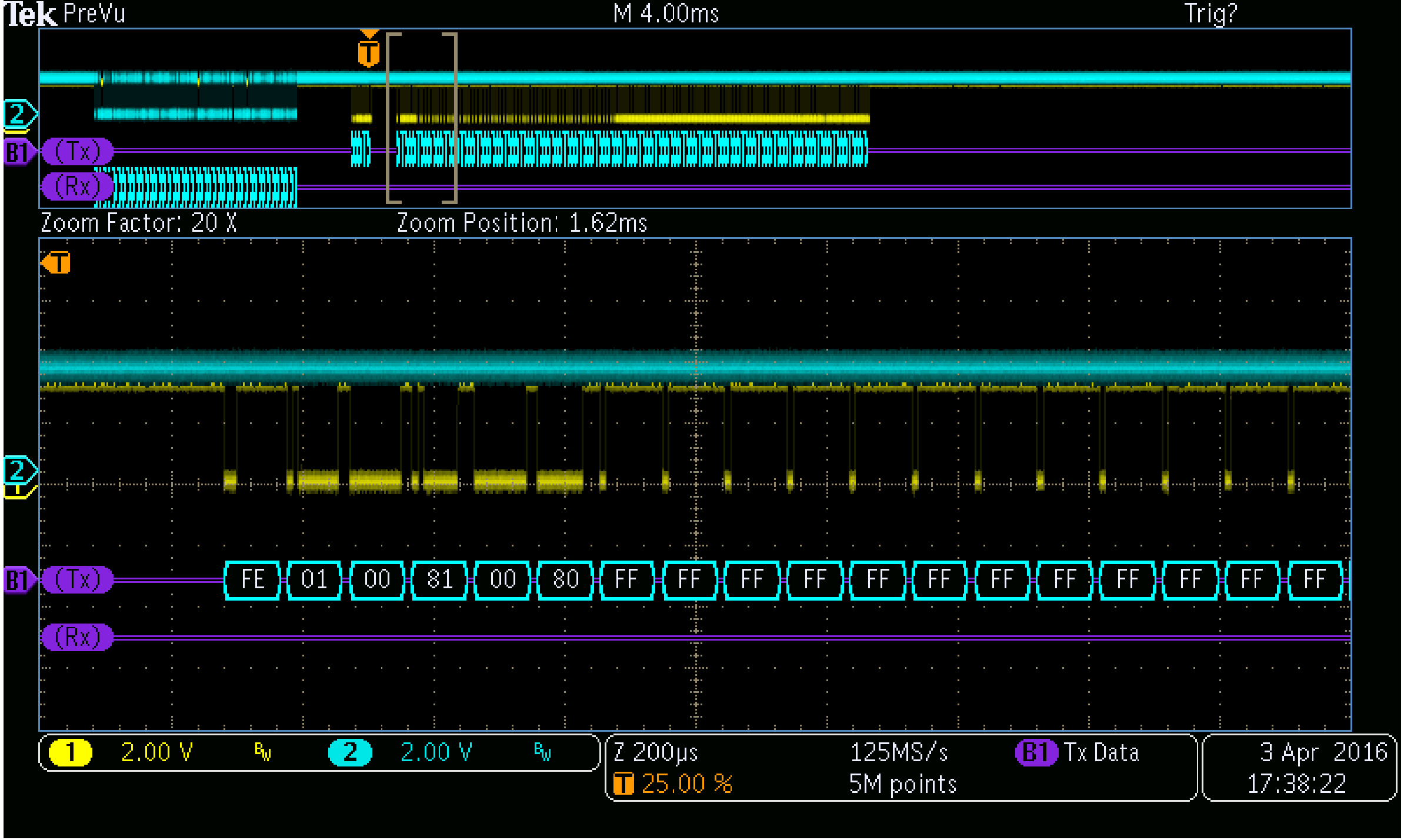


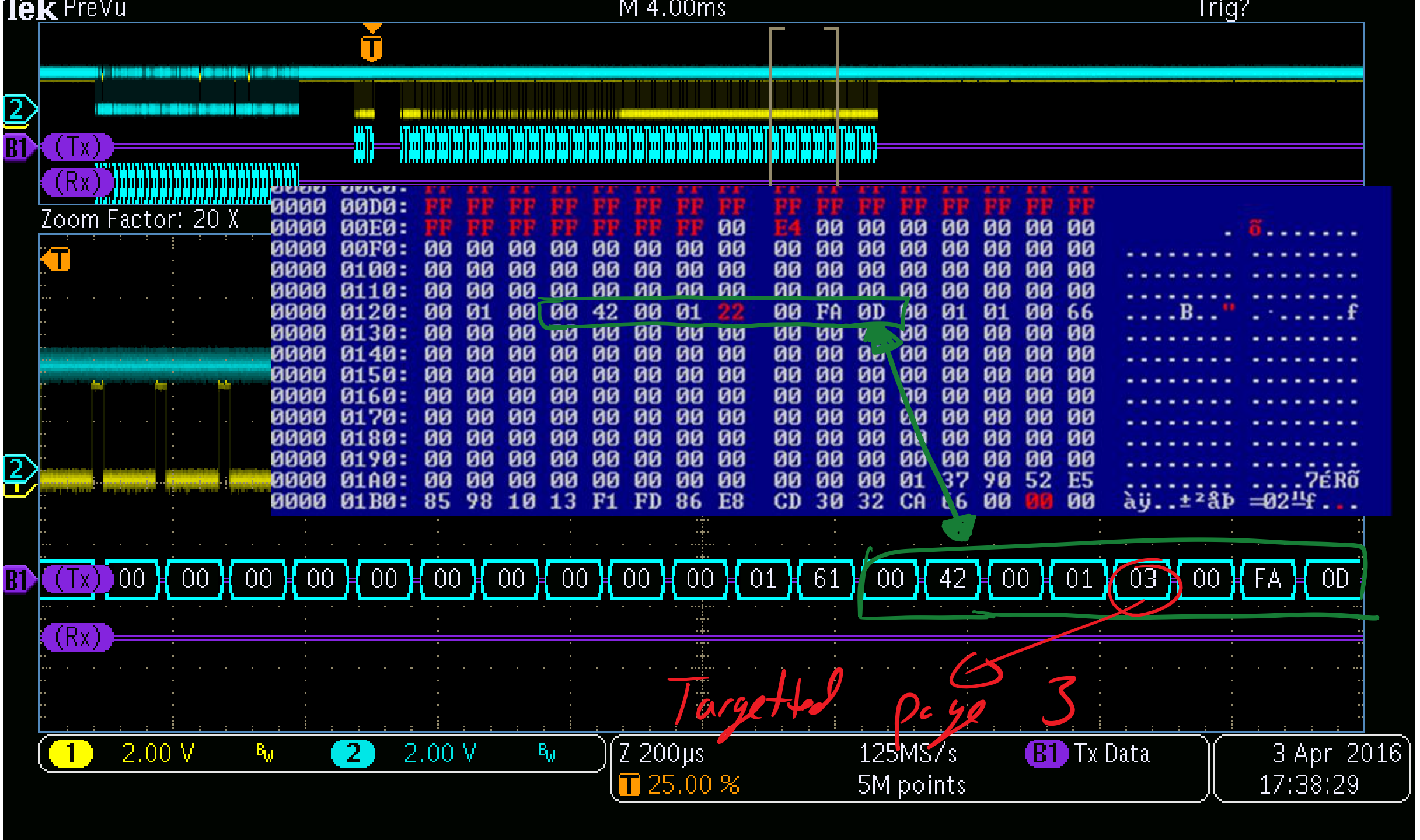
Width = 10%
Offset = +15%

Clock XORd with Glitch


Tek PreVu
M 4.00ms
Trig?
Zoom Factor: 20 X
Zoom Position: 1.84ms
Correct Response
Error!
Rx Buf!
B1 (Tx)
FE E3 00 81 0D 00 00 00 00 2A 00 01 01 00 66 CB 15 12
1 2.00 V BW
2 2.00 V BW
Z 200µs 250MS/s
T→ 1.50000ms 10M points
B1 Tx Data
Bus Search events found: 0
3 Apr 2016
15:38:03
Search On
Search Type Bus
Source Bus B1 (RS-232)
Search For Tx Data
Data 4A 27h


Tek PreVu
M 4.00ms
Trig?
0000 0070: 06 FF 00 FA 04 00 FA 04 00 5E 07 FC 0D 01 01 01
0000 0080: 01 2A 00 01 01 00 66 CB 15 12 16 15 12 33 03 7E
0000 0090: 80 87 74 01 F6 22 FC 87 DB 09 42 96 94 73 46 5C
0000 00A0: 16 FE 01 00 81 00 80 2A 00 01 01 00 66 CB 15 12
0000 00B0: 16 15 12 33 03 7E 80 2D 4A 27 D6 3C 49 6D B2 53
0000 00C0: 80 9E B7 CC 57 E1 95 A3 1A 1A 80 54 E1 01 28 83
0000 00D0: DA 24 B5 7E 4B AD 45 37 90 52 E5 85 98 10 13 F1
0000 00E0: FD 86 E8 CD 30 32 CA 00 F6 00 00 00 00 00 00 00
0000 00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(Tx) 66 CB 15 12 16 15 12 33 03 7E 80 87 74 01 F6 22 FC 87 DB 09
7.200µs
250MS/s
B1 Tx Data
1 2.00 V BW
2 2.00 V BW
T→▼1.50000ms
10M points
Bus Search events found: 0
3 Apr 2016
15:38:15
Search
On
Search Type
Bus
Source Bus
B1 (RS-232)
Search For
Tx Data
Data
4A 27h


Tek PreVu
M 4.00ms
Trig?
2
B1
(Tx)
(Rx)
Zoom Factor: 20 X
Zoom Position: 1.62ms
FE 01 00 81 00 80 FF FF FF FF FF FF FF FF FF FF FF
1 2.00 V BW
2 2.00 V BW
Z 200µs
T 25.00 %
125MS/s
5M points
B1 Tx Data
3 Apr 2016
17:38:22


Tek PreVu
M 4.00ms
Trig?
Zoom Factor: 20 X
0000 00D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0000 00E0: FF FF FF FF FF FF FF 00 E4 00 00 00 00 00 00 00
0000 00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0120: 00 01 00 00 42 00 01 22 00 FA 0D 00 01 01 00 66
0000 0130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 0190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000 01A0: 00 00 00 00 00 00 00 00 00 00 00 01 37 90 52 E5
0000 01B0: 85 98 10 13 F1 FD 86 E8 CD 30 32 CA 66 00 00 00
B1 (Tx) 00 00 00 00 00 00 00 00 00 00 01 61 00 42 00 01 03 00 FA 0D
(Rx)
Targetted page 3
1 2.00 V BW
2 2.00 V BW
Z 200µs
T 25.00 %
125MS/s
5M points
B1 Tx Data
3 Apr 2016
17:38:29

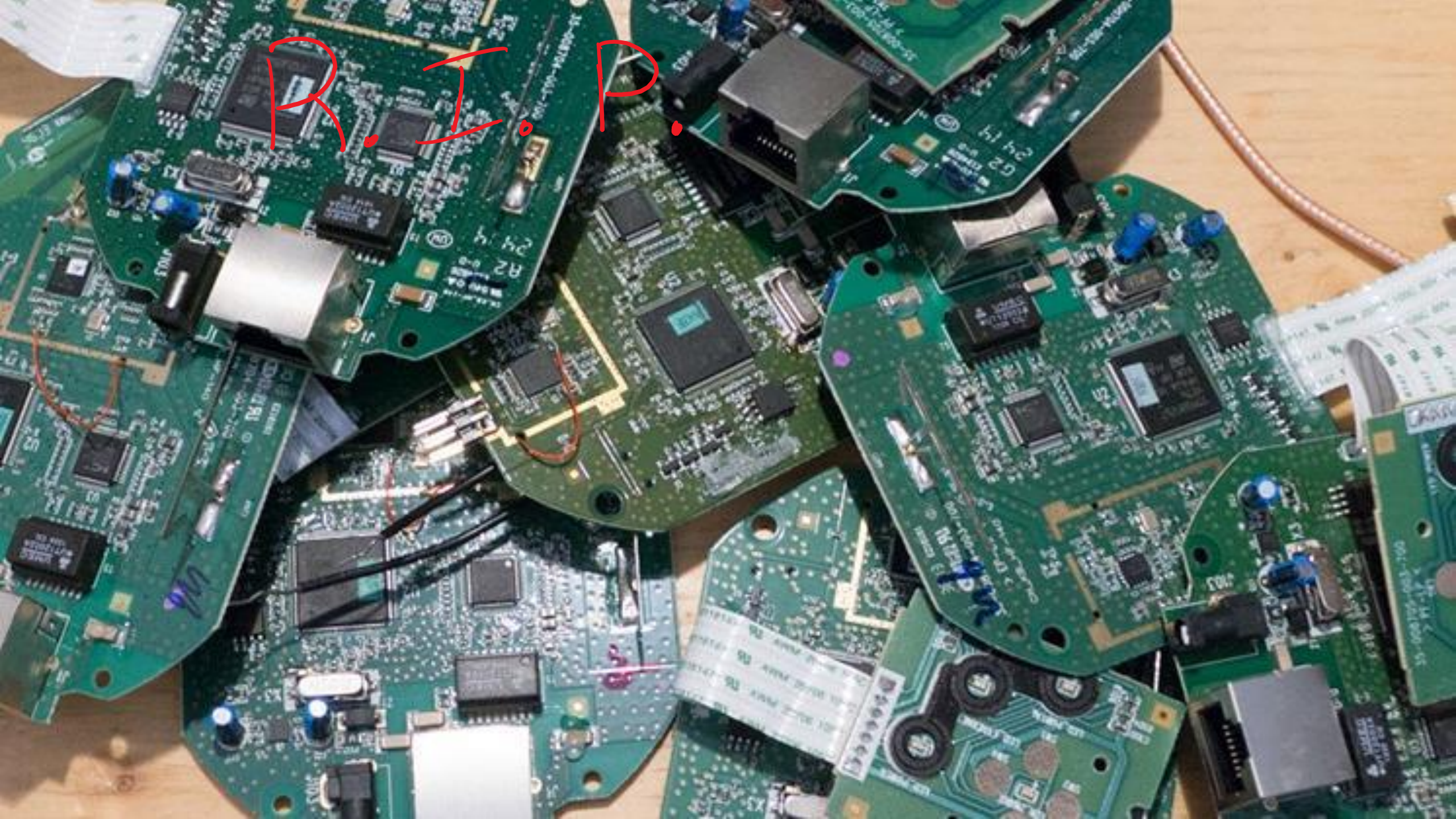
Appears section of SRAM is erased after use.

↳ This is good practice!

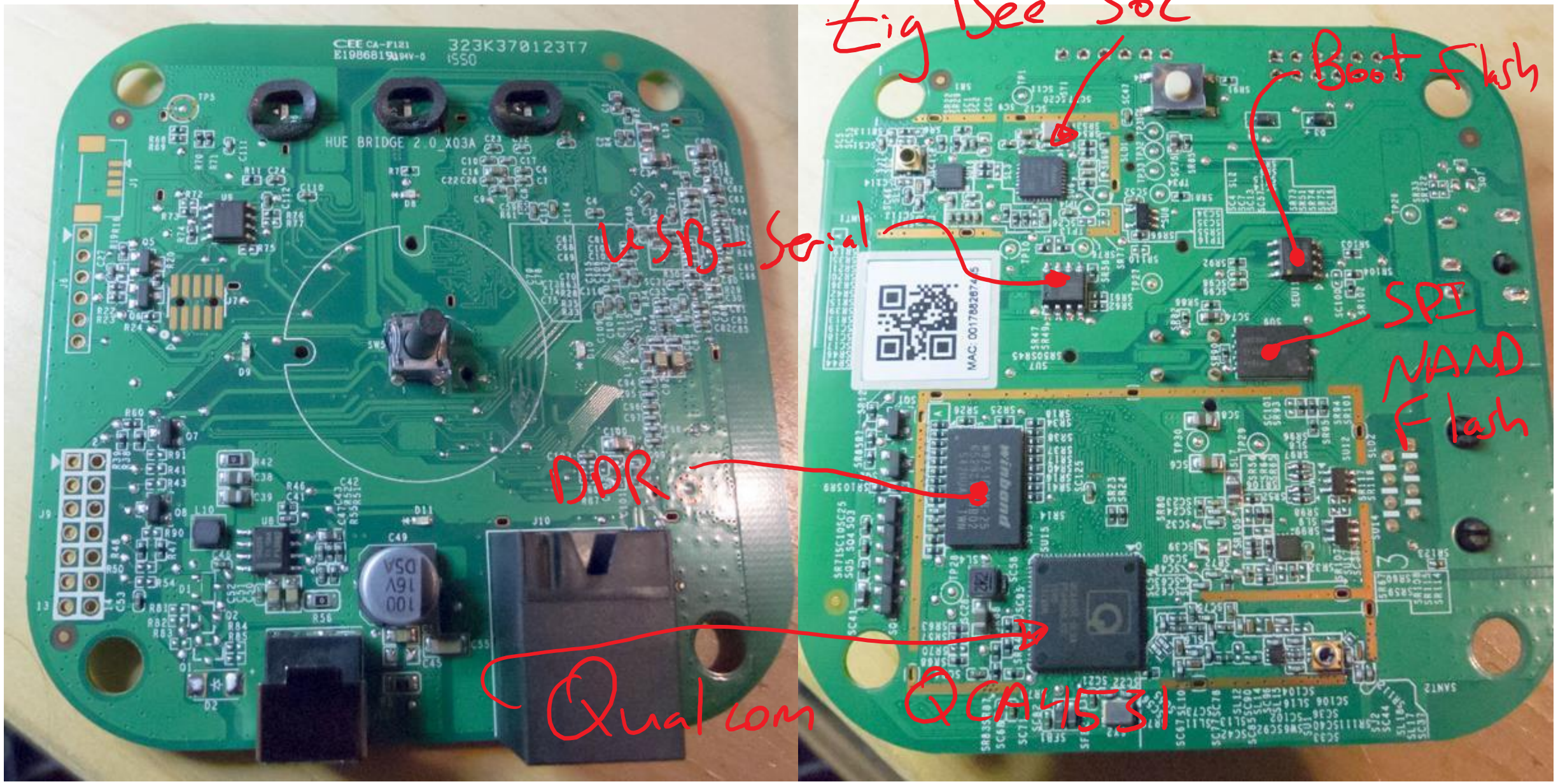
↳ May be possible with more glitches.

Glitch Attacks To Firmware

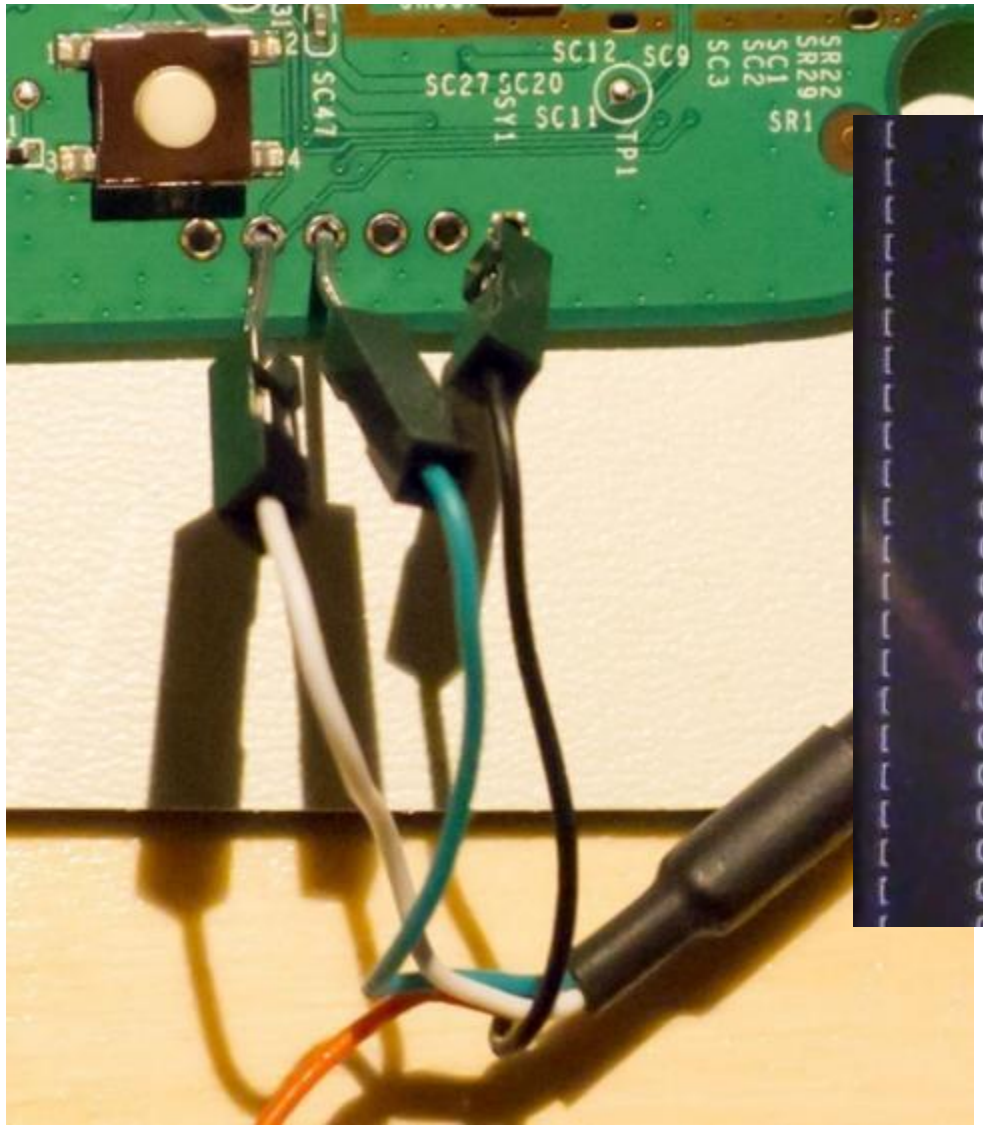
- Appears we can use glitching to dump SRAM.
- Careful timing required to get decrypted data.


R.I.P.


BRIDGE
2.0


HUE BRIDGE 2.0_X03A
323K370123T7
ISSO
MAC: 0017882E745
Zig Bee SoC
Boot Flash
USB-Serial
SPI NAND Flash
DDR
Qualcom
QCA4531

HACKING TOOLS

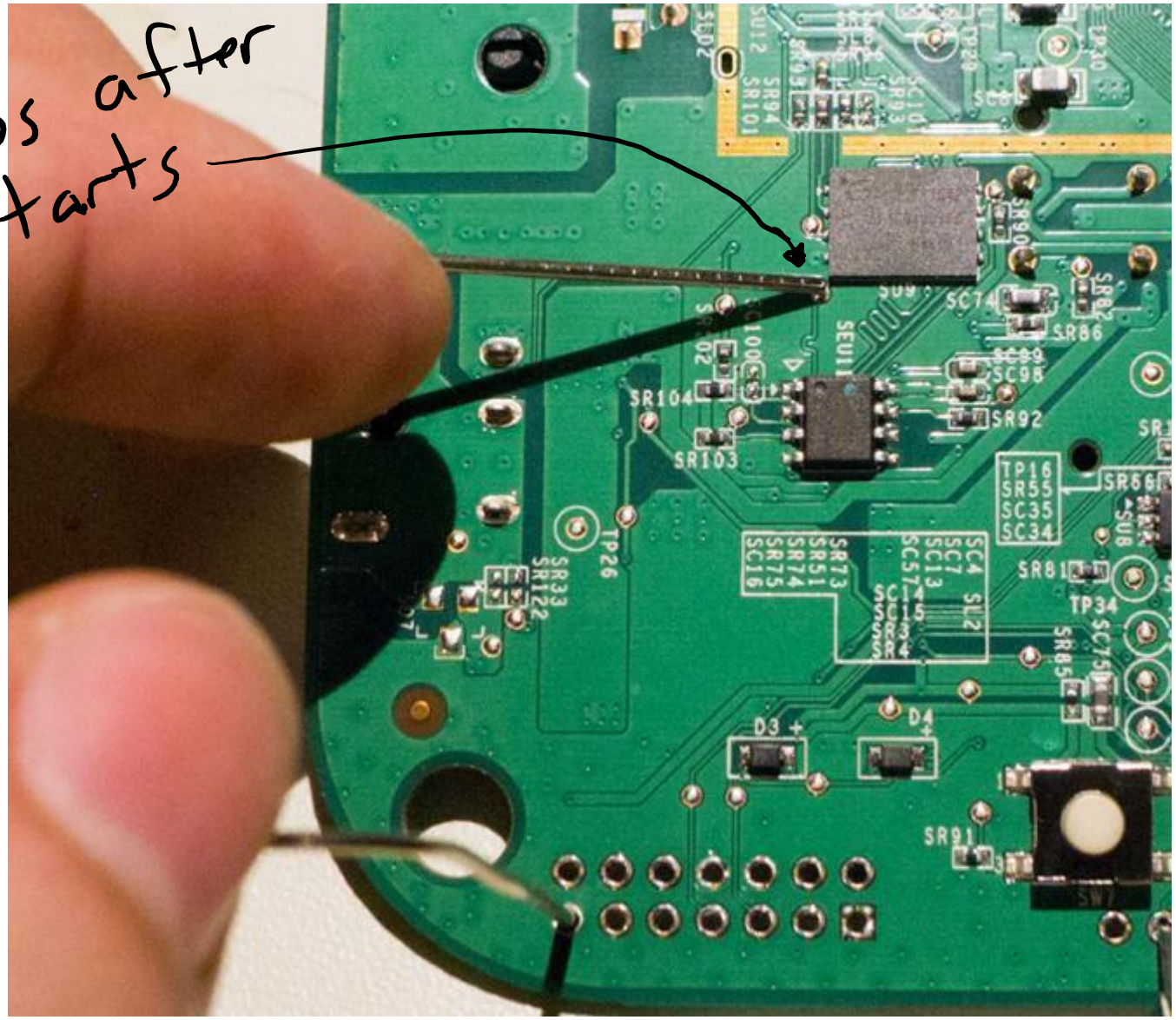



```
[    0.600000] io scheduler noop registered
[    0.600000] io scheduler deadline registered (default)
[    0.610000] Serial: 8250/16550 driver, 1 ports, IRQ sharing disable
[    0.630000] serial8250.0: ttyS0 at MMIO 0x18020000 (irq = 11, base_
[    0.640000] console [ttyS0] enabled
[    0.640000] console [ttyS0] enabled
[    0.650000] bootconsole [early0] disabled
[    0.650000] bootconsole [early0] disabled
[    0.660000] m25p80 spi0.0: found gd25d40, expected m25p80
[    0.670000] m25p80 spi0.0: gd25d40 (512 Kbytes)
[    0.670000] 4 cmdlinepart partitions found on MTD device spi0.0
[    0.680000] Creating 4 MTD partitions on "spi0.0":
[    0.680000] 0x000000000000-0x000000040000 : "u-boot"
[    0.690000] 0x000000040000-0x000000060000 : "u-boot-env"
[    0.690000] 0x000000060000-0x000000070000 : "reserved"
[    0.700000] 0x000000070000-0x000000080000 : "art"
[    0.710000] nand: device found, Manufacturer ID: 0xc8, Chip ID: 0xb
[    0.720000] nand: Giga Device GD5F1GQ4U 1G 3.3V 8-bit
[    0.730000] nand: 128MiB, SLC, page size: 2048, OOB size: 128
[    0.730000] Scanning device for bad blocks
[    0.900000] Bad eraseblock 768 at 0x000006000000
```

https://www.youtube.com/watch?v=hi2D2MnwiGM

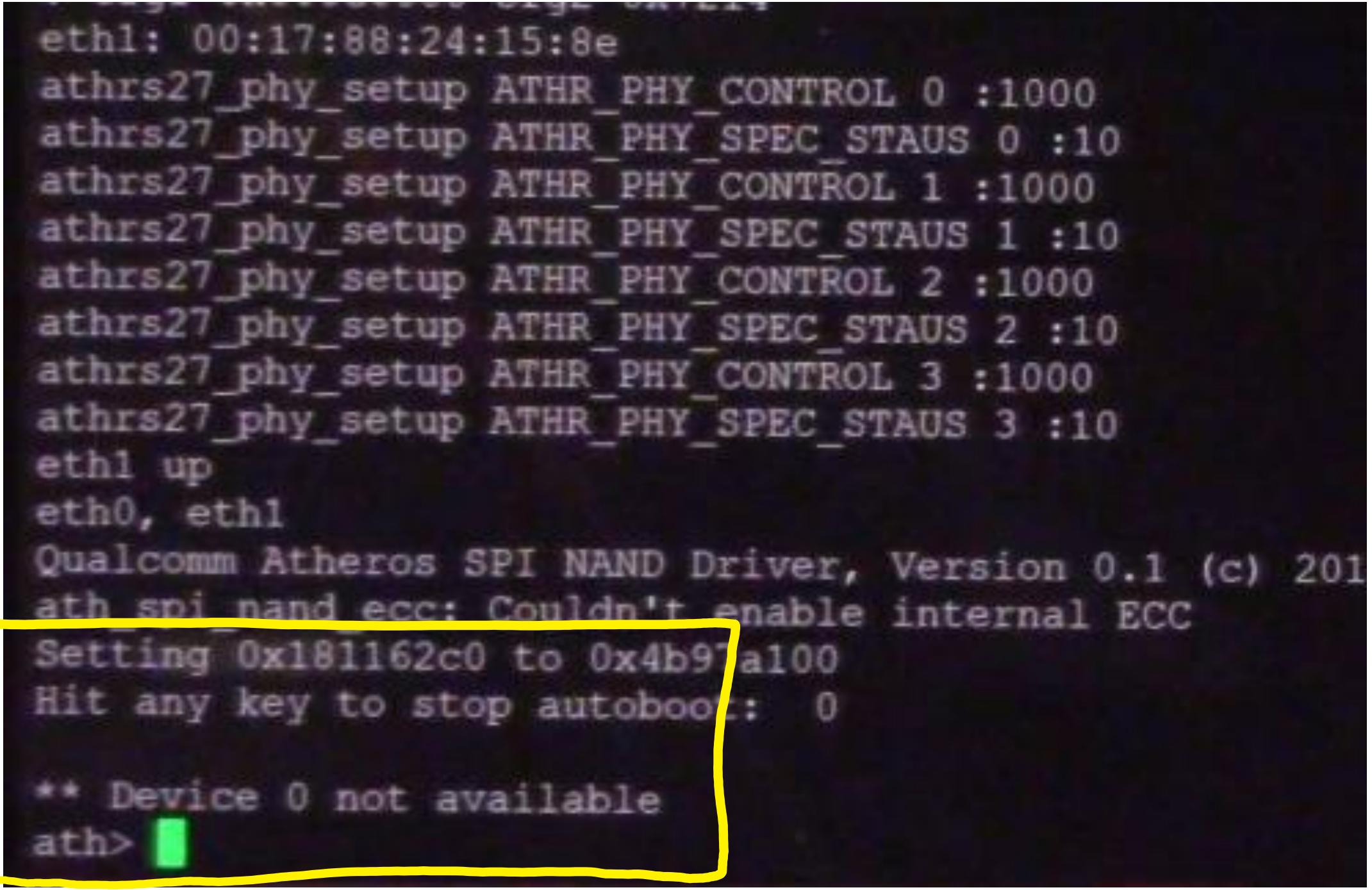
Or: http://www.oflynn.com

Short boot ~3s after starts



https://www.youtube.com/watch?v=hi2D2MnwiGM

Or: http://www.oflynn.com


eth1: 00:17:88:24:15:8e
athrs27_phy_setup ATHR_PHY_CONTROL 0 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 0 :10
athrs27_phy_setup ATHR_PHY_CONTROL 1 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 1 :10
athrs27_phy_setup ATHR_PHY_CONTROL 2 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 2 :10
athrs27_phy_setup ATHR_PHY_CONTROL 3 :1000
athrs27_phy_setup ATHR_PHY_SPEC_STAUS 3 :10
eth1 up
eth0, eth1
Qualcomm Atheros SPI NAND Driver, Version 0.1 (c) 201
ath_spi_nand_ecc: Couldn't enable internal ECC
Setting 0x181162c0 to 0x4b97a100
Hit any key to stop autoboot: 0
** Device 0 not available
ath>

Use "mkpasswd"

```
ath> setenv bootdelay 3
ath> printenv security
```

*****COPY THE DEFAULT VALUE THAT WAS PRINTED & SAVE THIS SOMEWHERE****

```
ath> setenv security '$5$wbgtEC1iF$ugIfQUoE7SNg4mplDI/7xdfLC7jXoMAkupeMsm10hY9'
ath> printenv security
security=$5$wbgtEC1iF$ugIfQUoE7SNg4mplDI/7xdfLC7jXoMAkupeMsm10hY9
ath> saveenv
ath> reset
```

https://www.youtube.com/watch?v=hi2D2MnwiGM

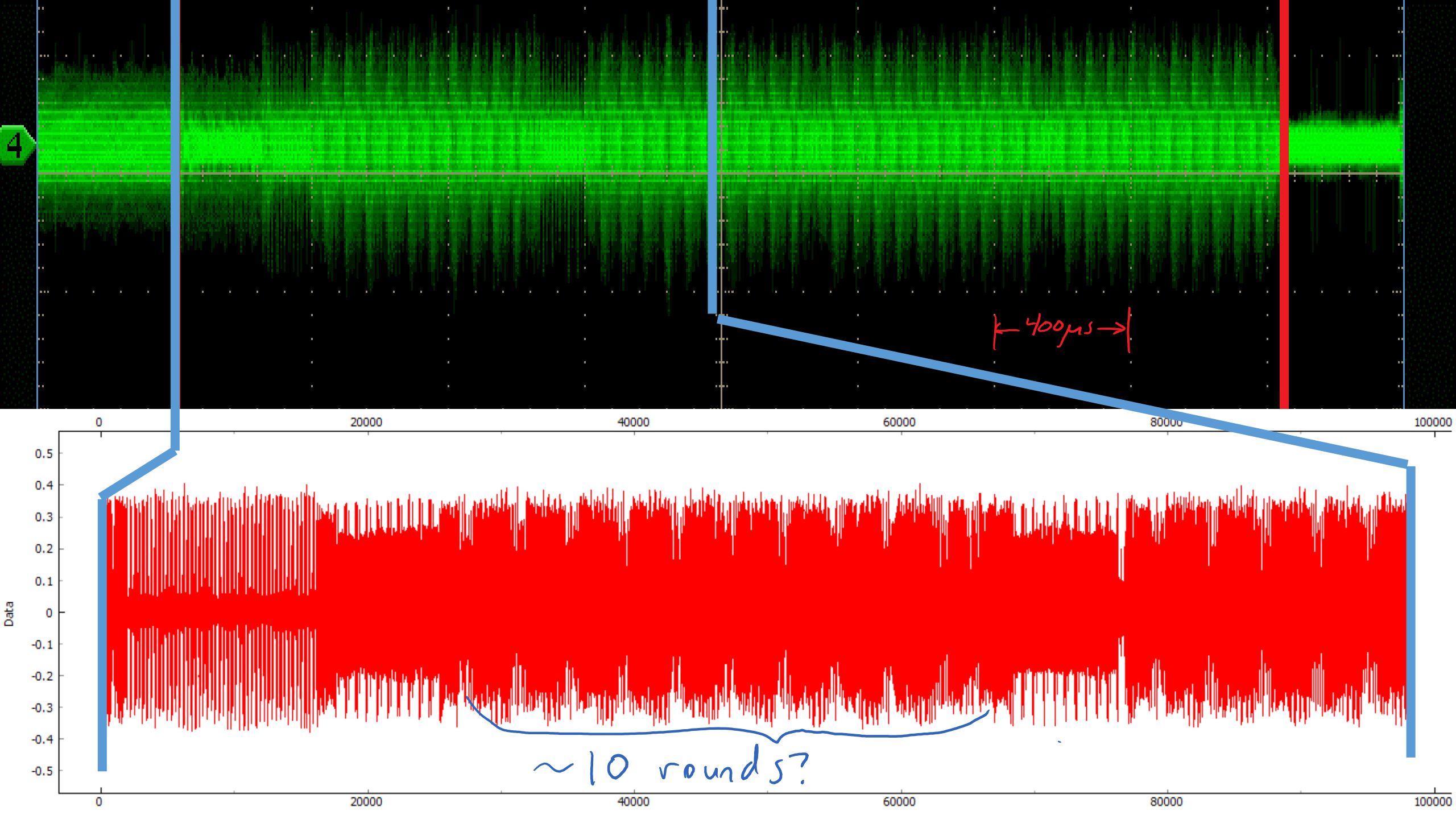
http://colinoflynn.com/?p=706

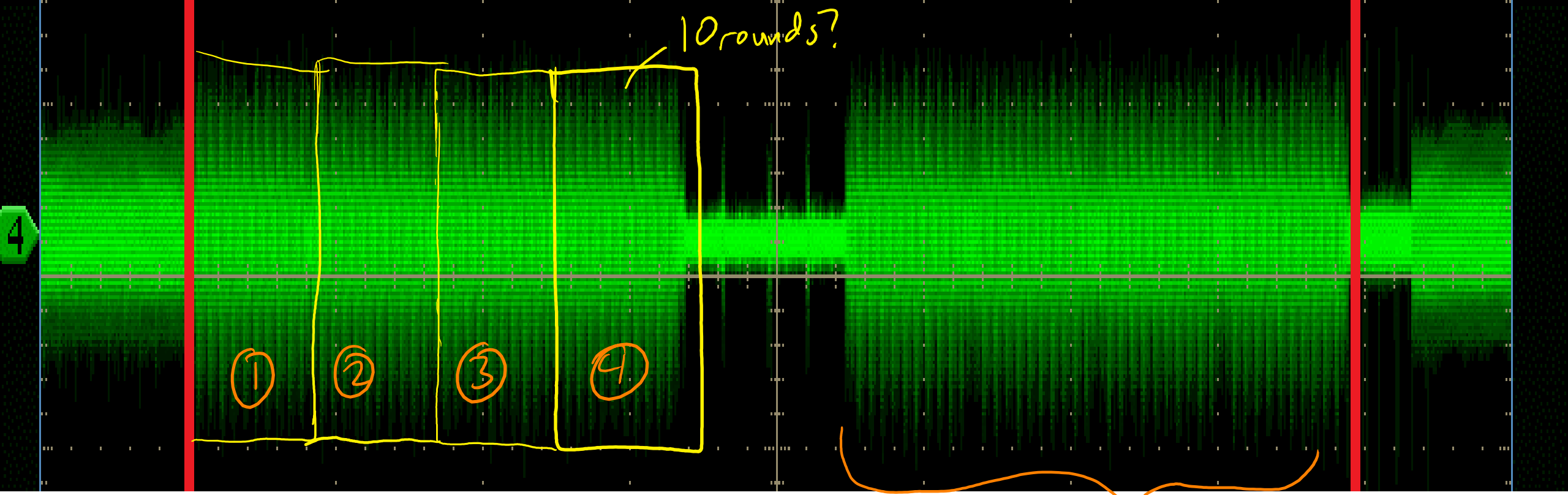
- Master binary seems to “do it all” (webserver, parsing requests, etc.) at `/usr/sbin/ipbridge`

- FW Update routine at /`usr/sbin/swupdate`
  - References AES-CBC-256 decryption routine, which references encryption key at `/home/swupdate/certs/enc.k`

  - Two different bridges used same AES key (not really a big deal, as we already have unencrypted binaries since we have root).


TX/RX
nRST
Power Analysis
CLK-IN


4
←400μs→
0
20000
40000
60000
80000
100000
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Data
~10 rounds?



① – ④ = AES-128?
Would explain 64 bytes.

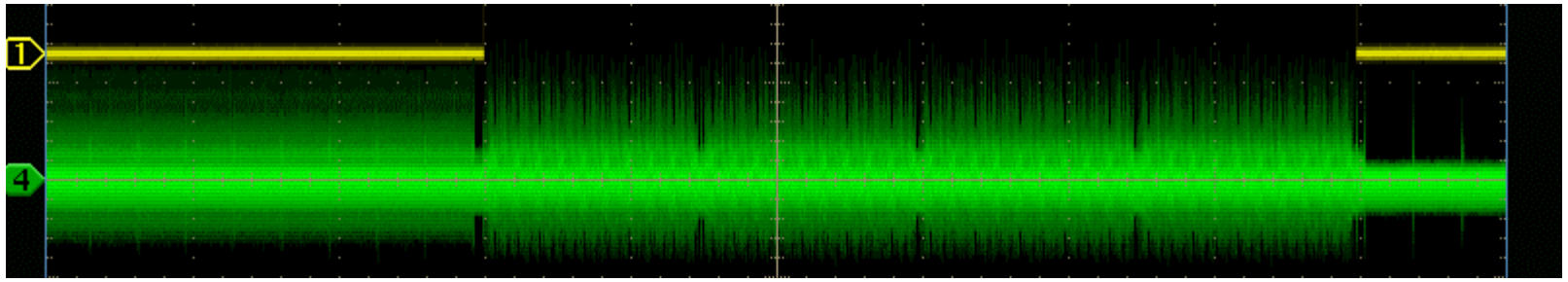
Again!?
↳ Maybe signature?

Previous slide: power signature of first 64-byte block sent (sign-on info?).
This slide: Power signature for remaining 64-byte blocks (delay varies).

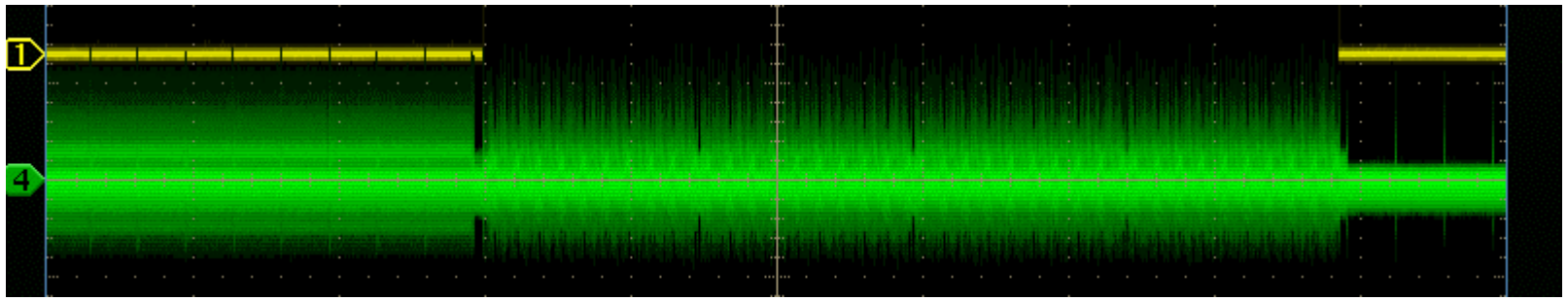
SAM D20 Xplained Pro *(109)*

SAM D21 Xplained Pro *(232)*

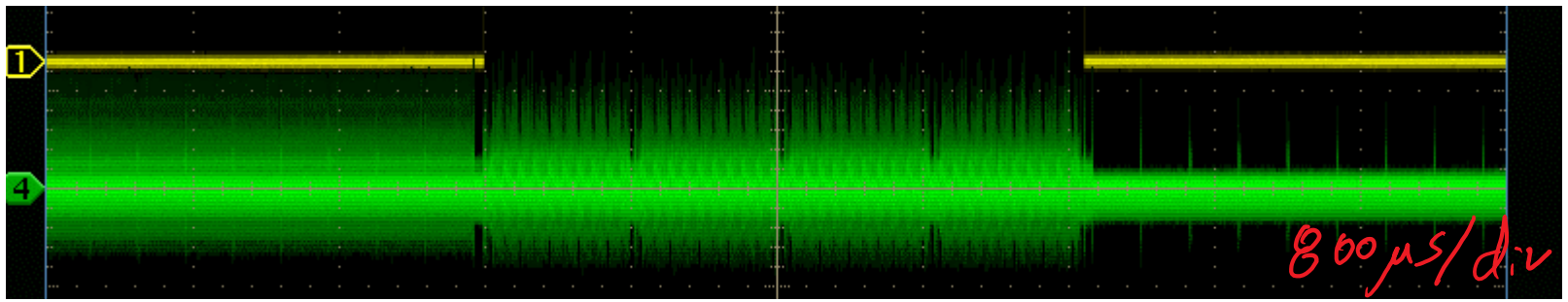
- 8MHz Oscillator Calibration Application - SAM D21 Xplained Pro
- ADP example application - SAM D21 Xplained Pro
- AES Software Library Demo - SAM D21 Xplained Pro
- Alert Notification Client Application - SAM D21 Xplained Pro



ECB



CBC

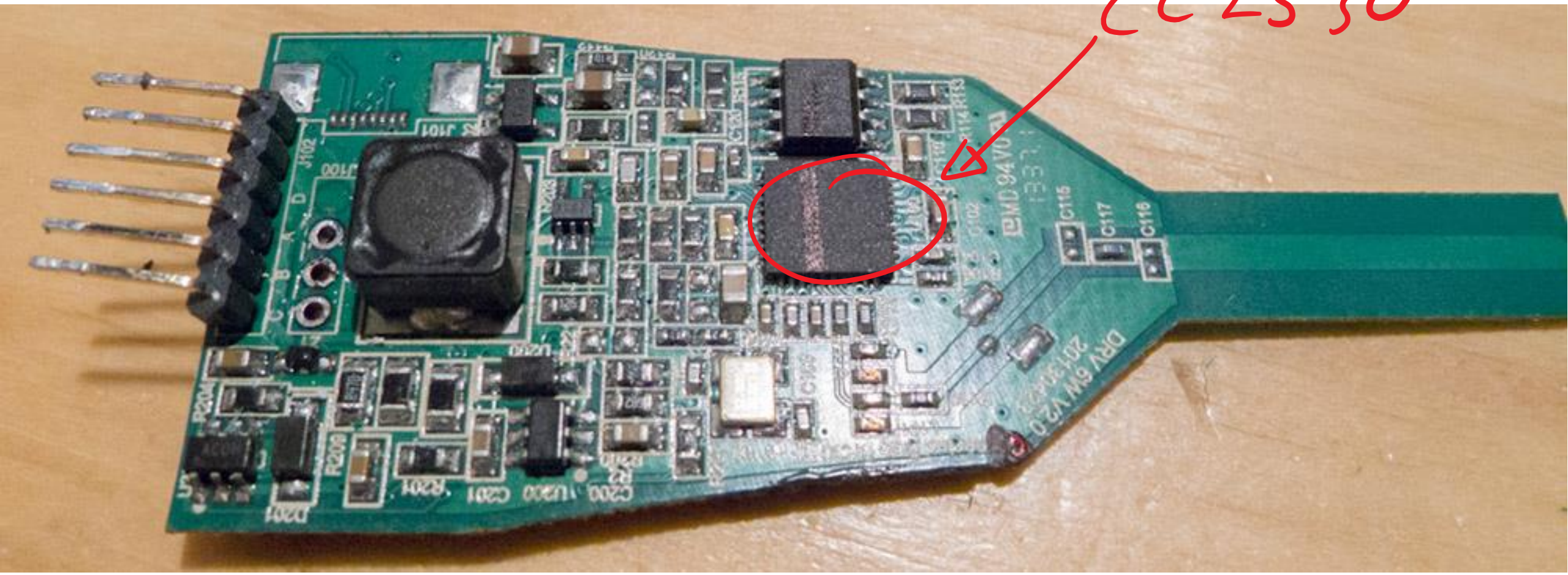


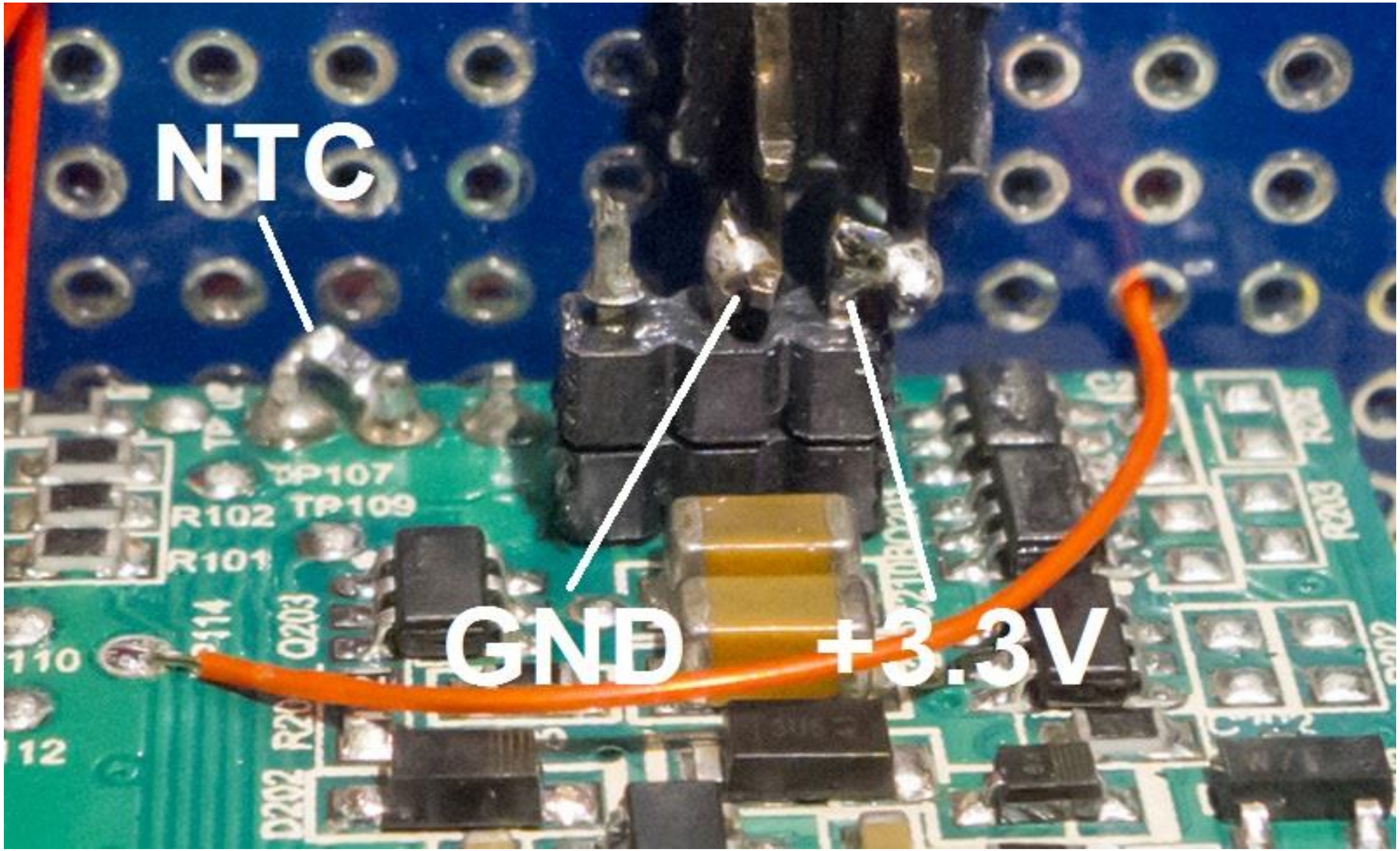
CTR

64 BYTE DECRYPTION

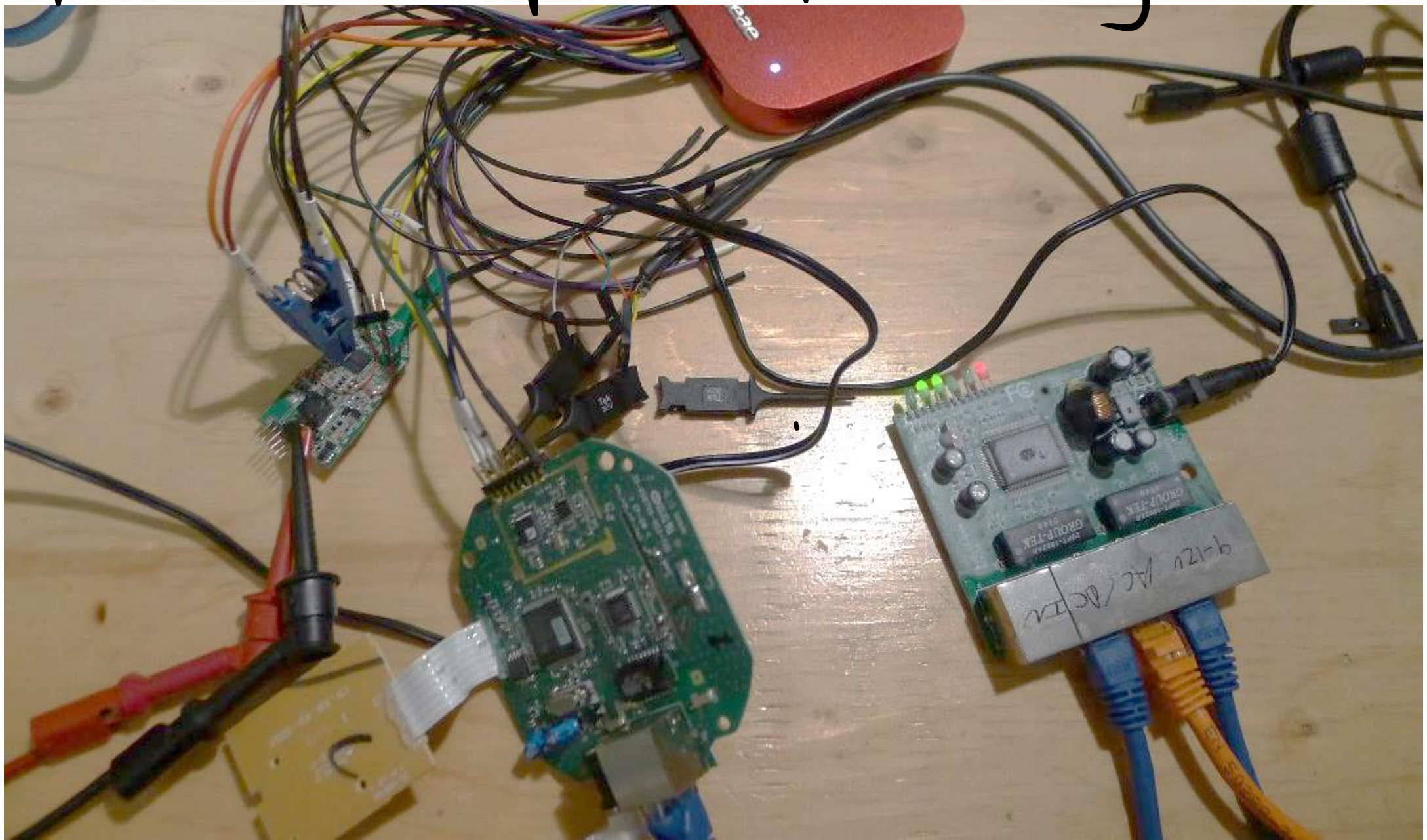

BR30
DOWNLIGHT

CC2530 Based


CC2530

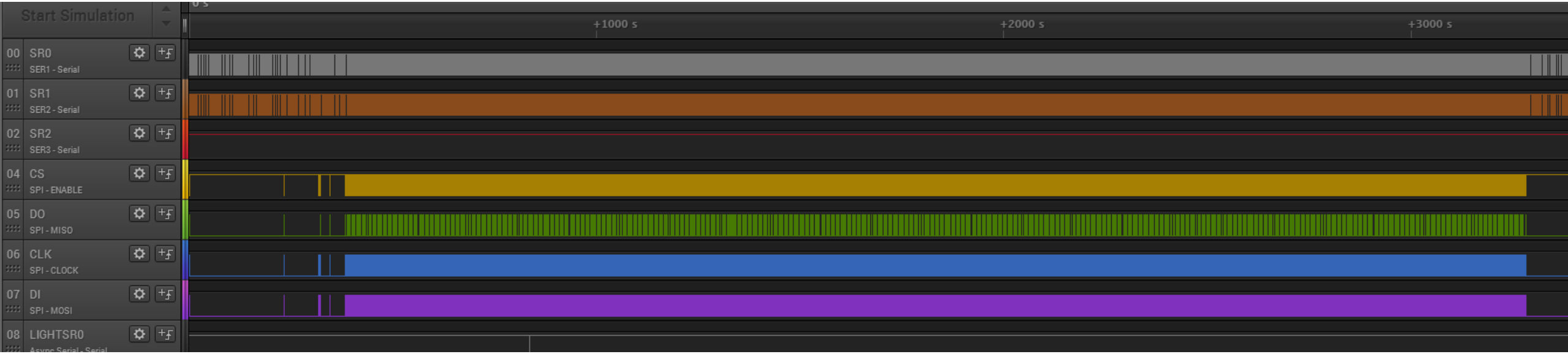

NTC
GND
+3.3V
TP107
TP109
R102
R101
Q203
D202

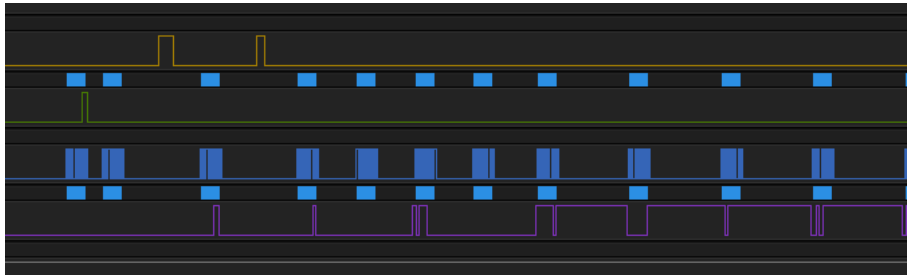
Firmware Update Hacking

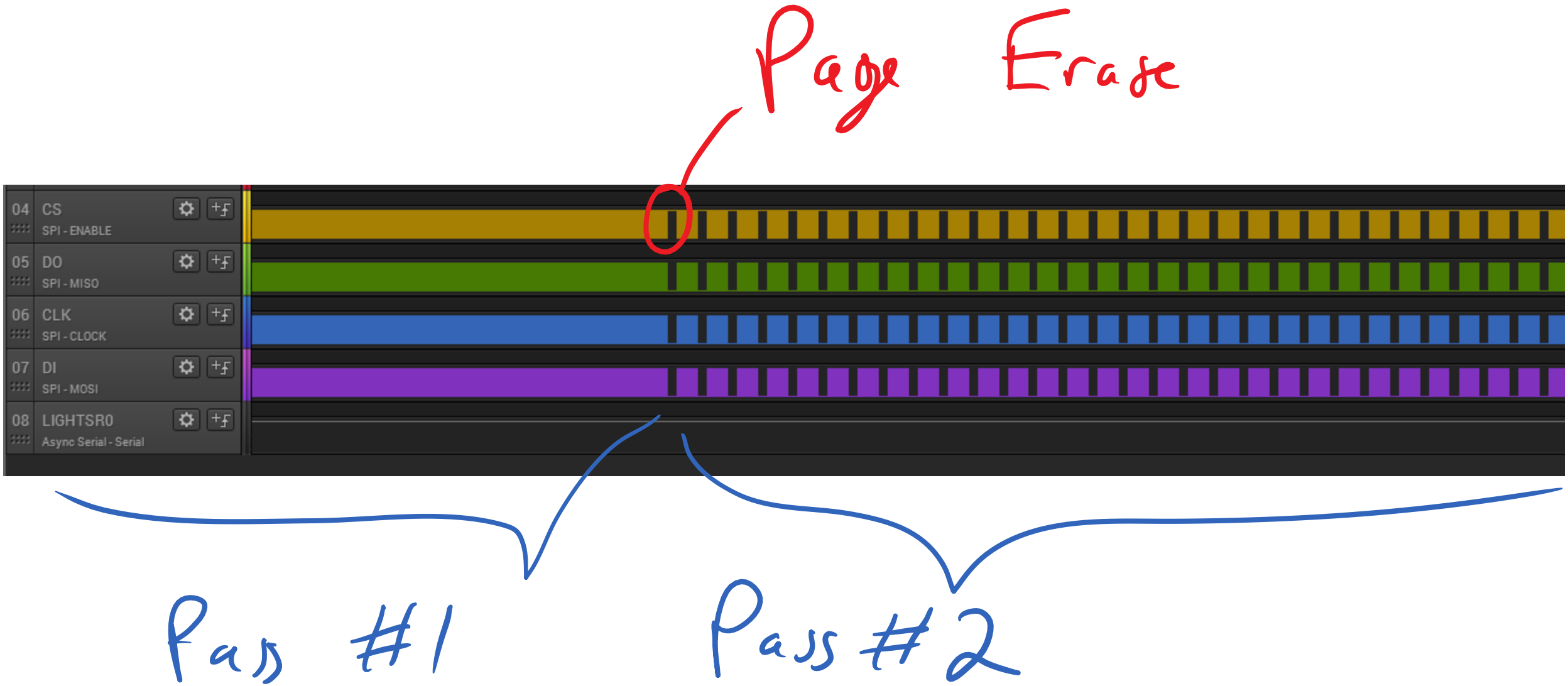


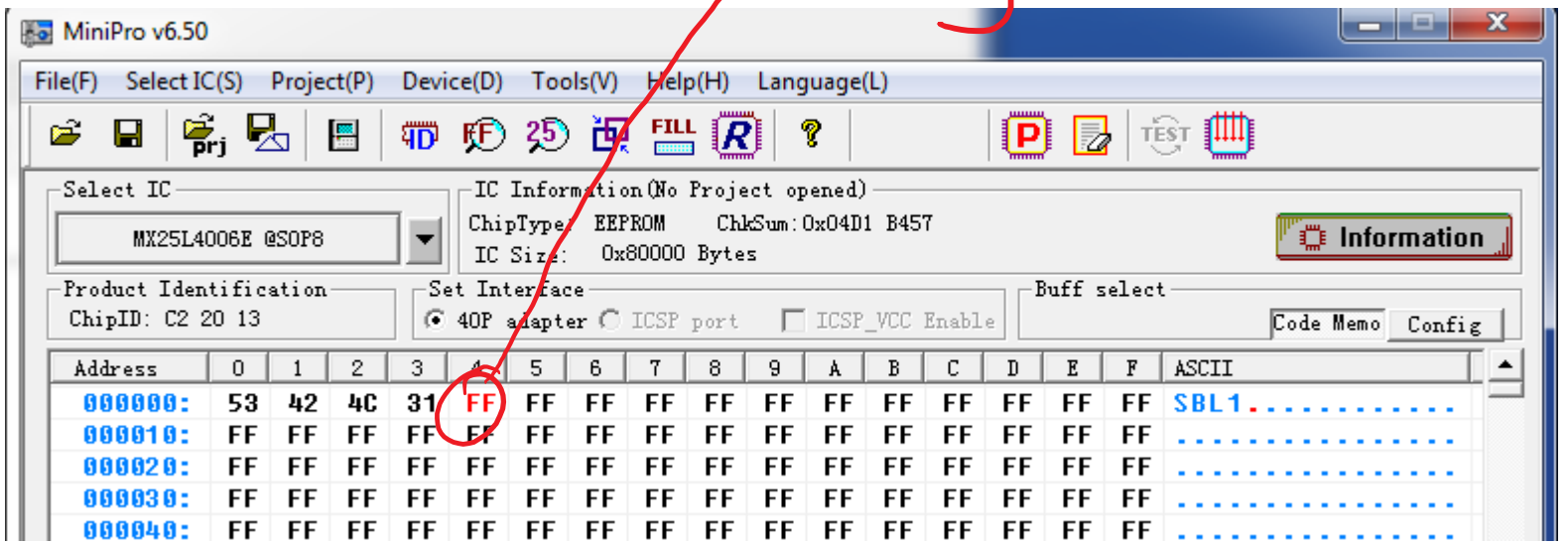
Using Salae Pro-16

↳ Capture 8MHz SPI Traffic






Page Erase
04 CS
SPI - ENABLE
05 DO
SPI - MISO
06 CLK
SPI - CLOCK
07 DI
SPI - MOSI
08 LIGHTSR0
Async Serial - Serial
Pass #1
Pass #2



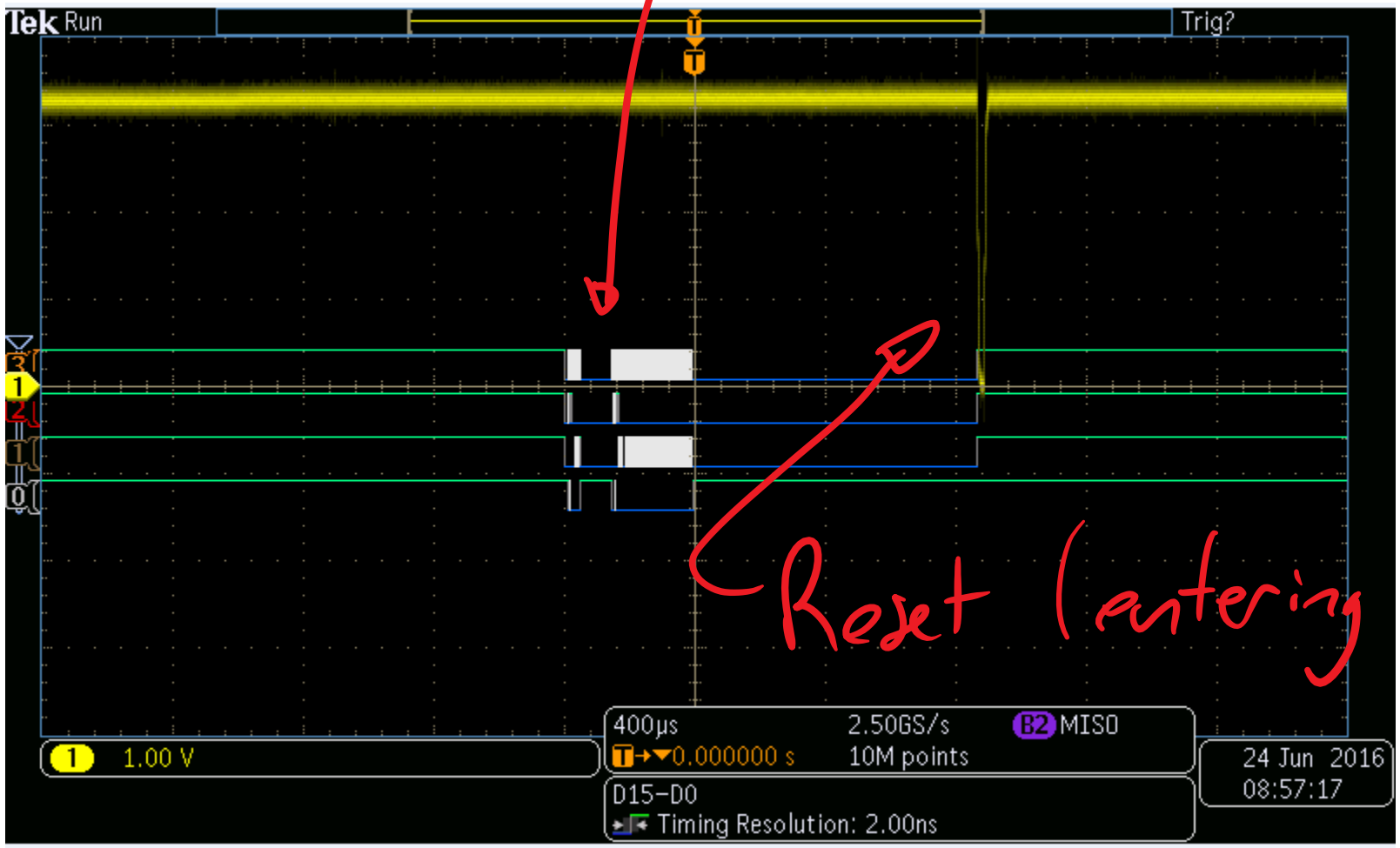
First block Sent

```
000780: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
000790: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0007A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0007B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0007C0: 2A 00 01 00 00 66 52 14 10 02 17 30 39 03 EF 40 *....fR....09..@
0007D0: 2E 37 0B 25 EC C0 47 65 CB E1 1E 0E 74 F7 A1 14 .7.%..Ge....t...
0007E0: EE 6B 58 B5 2F F3 0D 83 68 12 67 71 4C 7A 75 20 .kX./...h.gqLzu
0007F0: 4D 08 E0 74 95 54 CE AB 23 72 2B 80 AB 46 46 CD M..t.T..#r+..FF.
000800: 77 CF AC 2E 8C 58 9E 75 8C 1D 77 43 D5 A2 28 5C w....X.u..wC..(\
000810: 4E 94 CC F9 C8 C5 5B 62 E7 09 8B E3 6A 3A 0C 07 N.....[b....j:..
000820: 86 27 80 7A 76 91 90 AA 1E 8F 40 FD 35 96 CC C0 .'.zv.....@.5...
000830: BF 53 2D F0 88 7E 28 ED F3 B7 96 AF 65 8C 8A 1D .S-..~(.....e...
000840: D6 8B 07 49 EE 8C B7 49 54 D9 D9 62 94 62 65 0C ...I...IT..b.be.
000850: 99 E4 B8 4A CE 17 26 28 A8 FF F3 4C 48 45 B0 A0 ...J..&(...LHE..
000860: 2E 29 3D 2A 4E 1D 40 42 C3 8A 9D E0 D6 6E 47 98 .)=*N.@B.....nG.
000870: D3 42 47 CF 29 EC BC 88 CB FB 35 15 CD DB 8A FE .BG.).....5.....
```

SRAM Dump

↳ During Bootload

That block from previous page.



Reset (entering debug)


Block
Address of SPI
0
??
First 16 bytes of block
What's this?
SRAM Dump
IV?
Signature?


Security
Conclusions

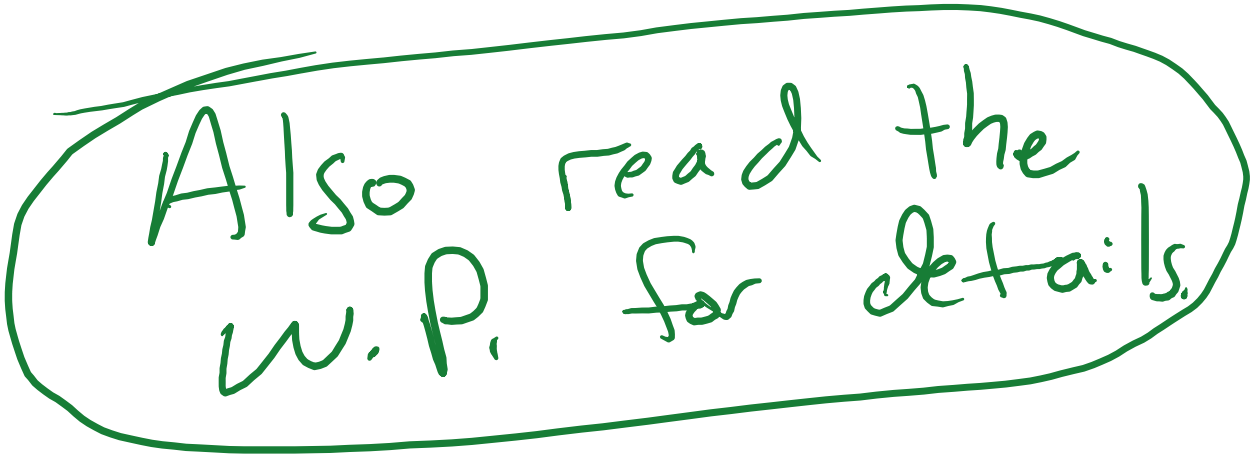
1. Huge risk to Philips if worm designed.
2. Good security practices in place to prevent this:
   - Encrypted FW
   - Signed FW (Linux only)
   - Keep keys out of SRAM
   - Clear memory when done.
3. Trade-offs may cause future problems:
   - Same key decrypts FW updates across many devices.
   - ZLL master key leak opens up lamp-stealing.
   - Huge Linux binary does a lot, vulns?
   - See White Paper for more!

Eyal Ronen

http://www.wisdom.weizmann.ac.il/~eyalro/
http://www.wisdom.weizmann.ac.il/~eyalro/PhilipsHueTakeOver/
@eyalr0
eyal.ronen@weizmann.ac.il

Colin O'Flynn

@colinoflynn
oflynn.com
newae.com
coflynn@newae.com



http://colinoflynn.com/2016/08/philips-hue-r-e-whitepaper-from-black-hat-2016/