A Lightbulb Worm?

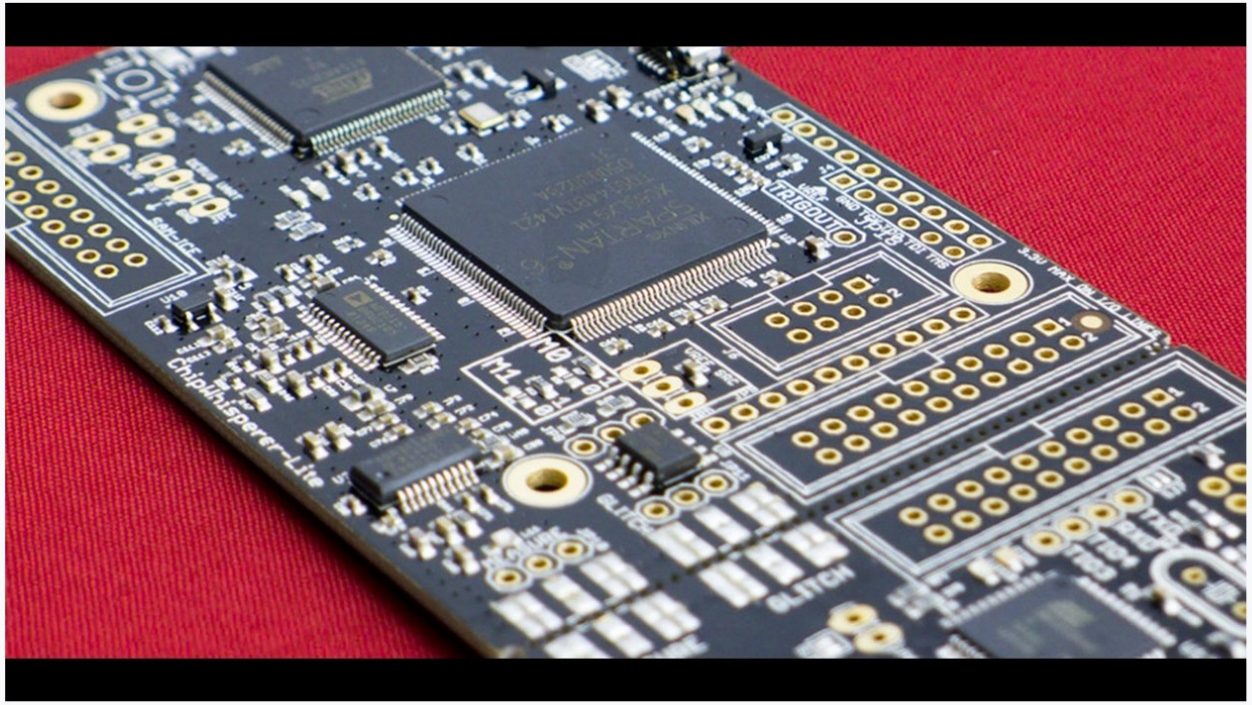
A teardown of the Philips Hue.

Colin O'Flynn

(with special appearance by Eyal Ronen)

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.

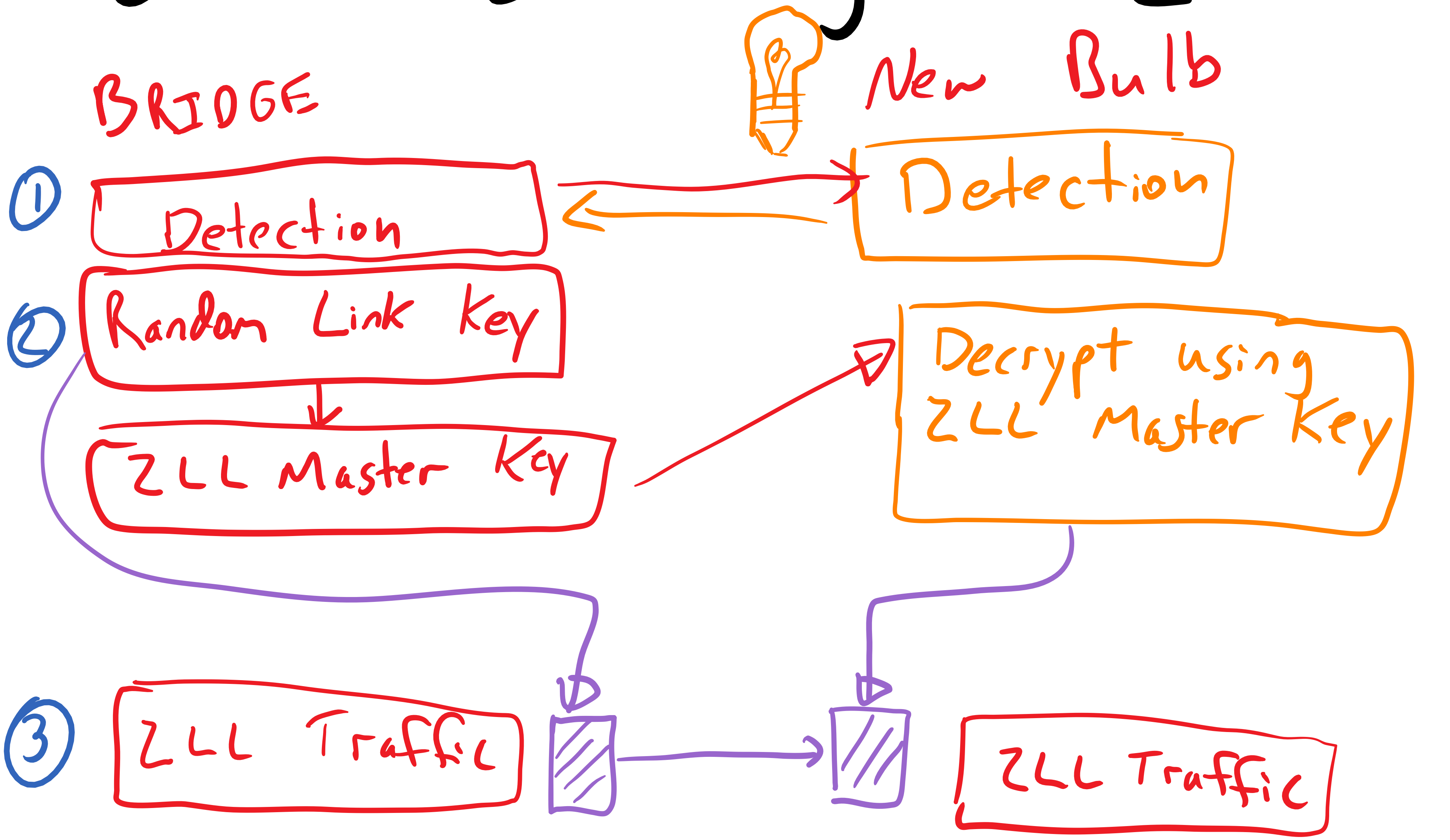

PHILIPS
I WANT / JE VEUX
hue
Available on the
App Store


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



Light Bulb Theft

6.4.4 Stealing a Node

A node that is already part of a ZLL network can be taken or 'stolen' by another ZLL network using Touchlink (in which case, the stolen node will cease to be a member of its previous network). This transfer can only be performed on a node which supports one or more Lighting devices (and not Controller devices).

The node is stolen using an initiator in the new network, e.g. from a remote control unit. The 'stealing' process is as follows:

1. The initiator sends a Scan Request to nodes in its vicinity. The required function is:

   eCLD_ZllCommissionCommandScanReqCommandSend()

2. A receiving ZLL node replies to the Scan Request by sending a Scan Response. The required function is:

   eCLD_ZllCommissionCommandScanRspCommandSend()

3. The initiator receives Scan Responses from one or more nodes and, based on these responses, selects a node (containing a Lighting device) that is already a member of another ZLL network.

4. The initiator then sends a Reset To Factory New Request to the desired node. The required function is:

   eCLD_ZllCommissionCommandFactoryResetReqCommandSend()

5. On receiving this request on the target node, the event E_CLD_COMMISSION_CMD_FACTORY_RESET_REQ is generated and the function **ZPS_eAplZdoLeaveNetwork()** should be called. In addition, all persistent data should be reset.

6. The node can then be commissioned into the new network by following the process in Section 6.4.2 from Step3.

Reply based on signal power, should only work at short distances.

8.1.2 Channels

A ZLL device shall be able to operate on all channels available at 2.4GHz, numbered from 11 to 26. When operating on channel 26, the transmission power may be reduced in order to comply with FCC regulations.

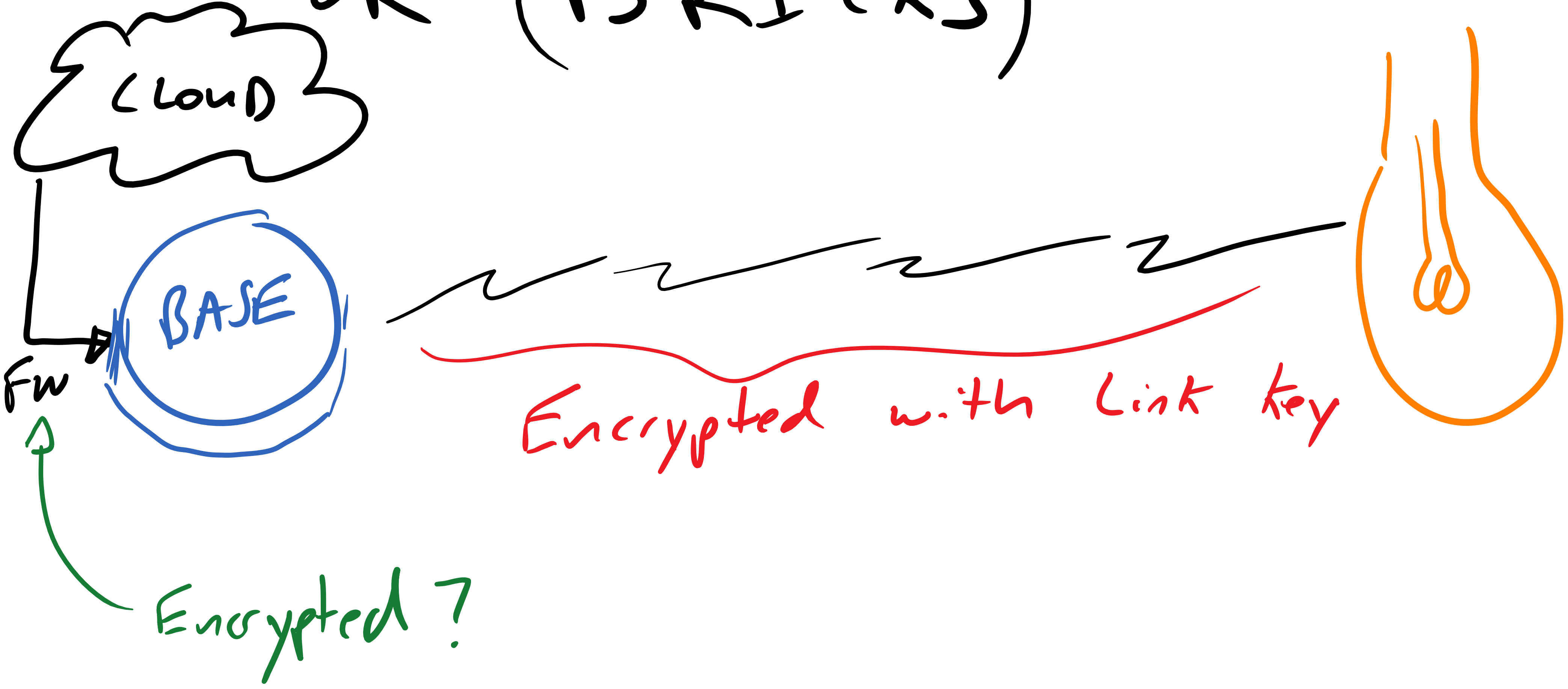
Within this range, two sets of channels shall be defined. The ***primary*** ZLL channel set shall consist of channels 11, 15, 20 and 25 and shall be used in preference for commissioning and normal operations. The ***secondary*** ZLL channel set shall consist of channels 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24 and 26 and can be used as a backup to allow the ZLL device to connect to a non-ZLL network.

Demo by Eyal Ronen

See http://www.wisdom.weizmann.ac.il/~eyalro/

Light Bulb Malware

or (Bricks)

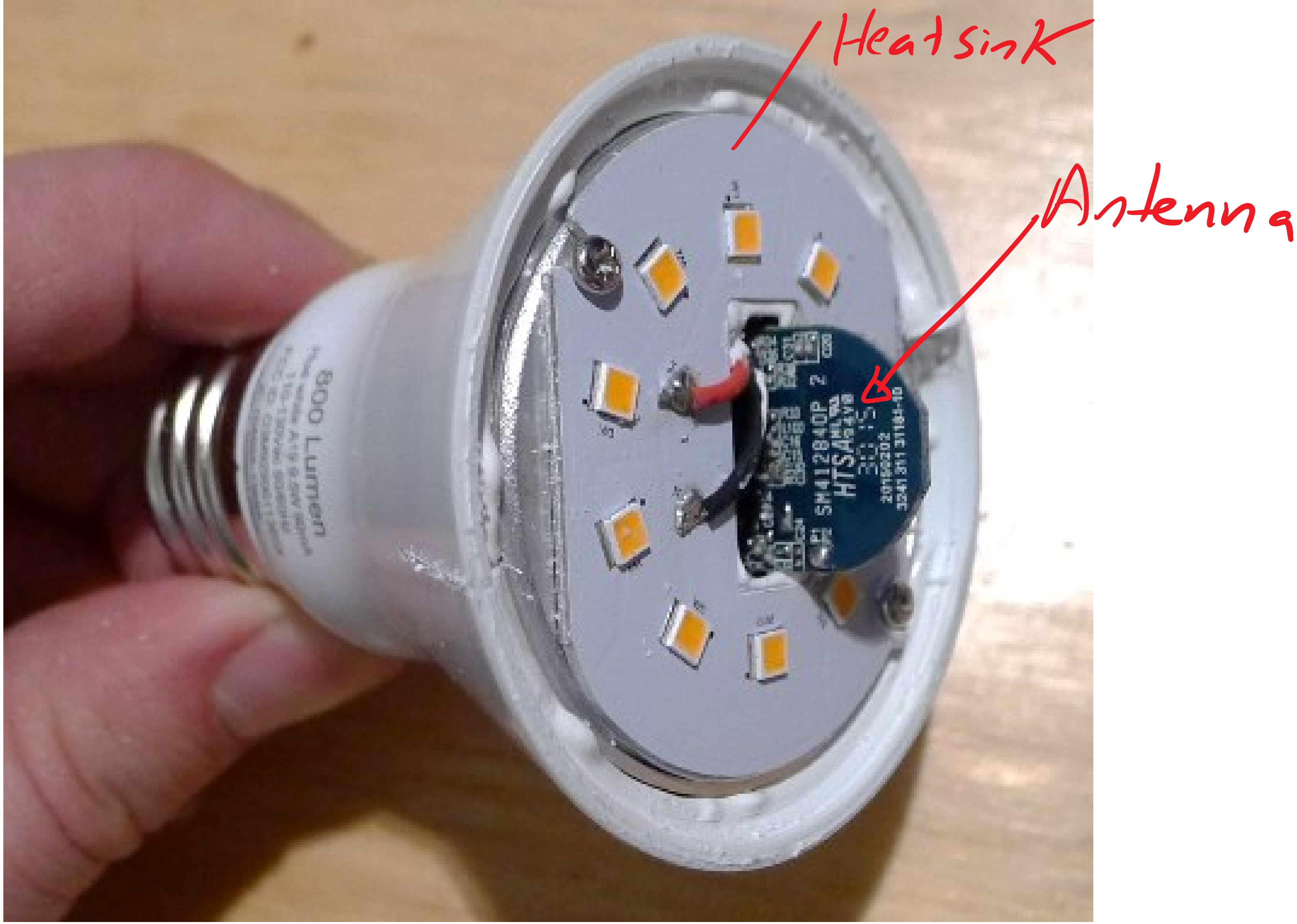


LIGHTBULB MALWARE

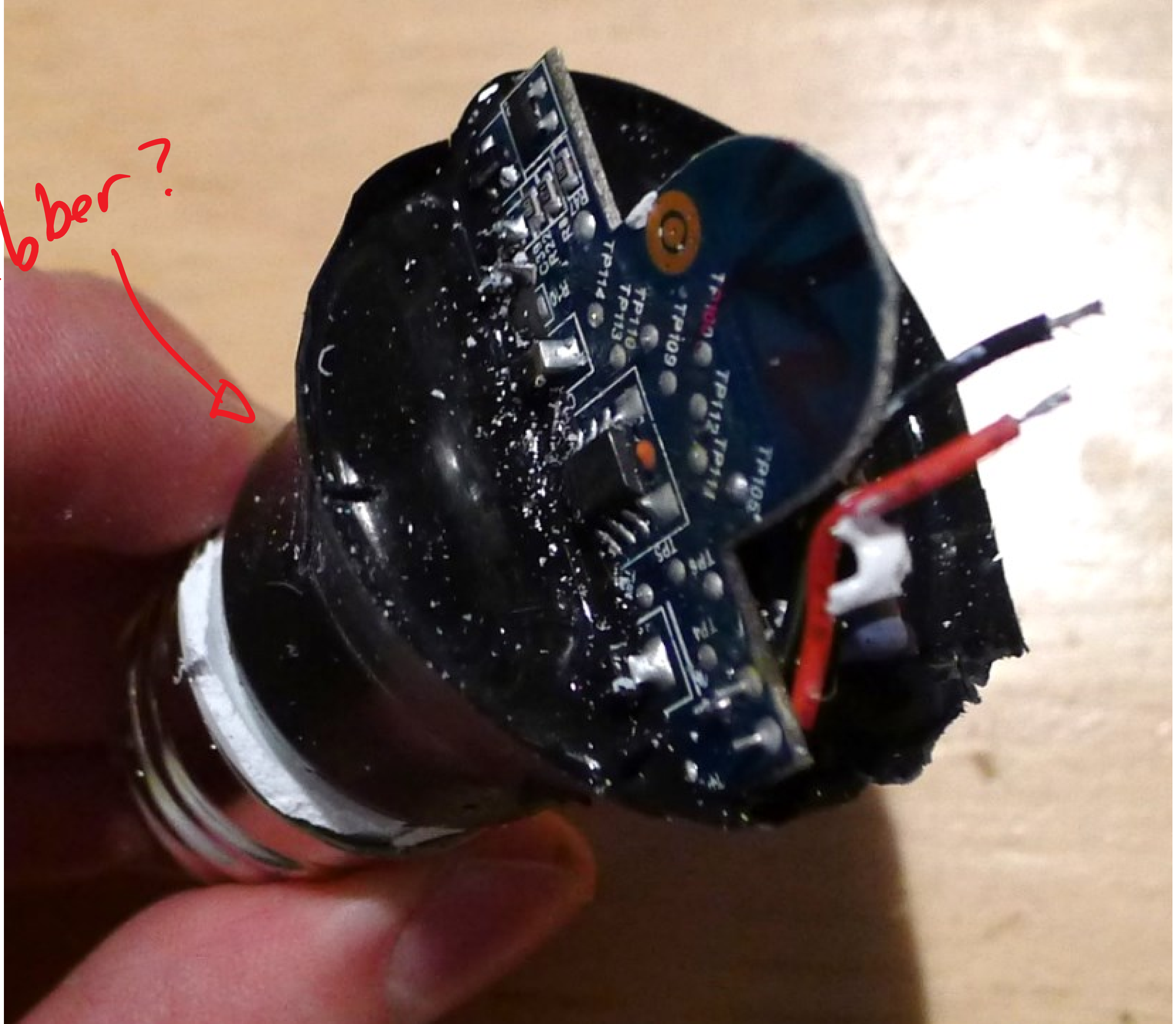
1) ZLL key leaked. We know it's possible to "steal" bulbs.
2) Custom FW on bulbs could turn bulb into "bridge" that searchs for & steals nearby bulbs.
3) If could cause other bulbs to perform OTA FW update → WORM

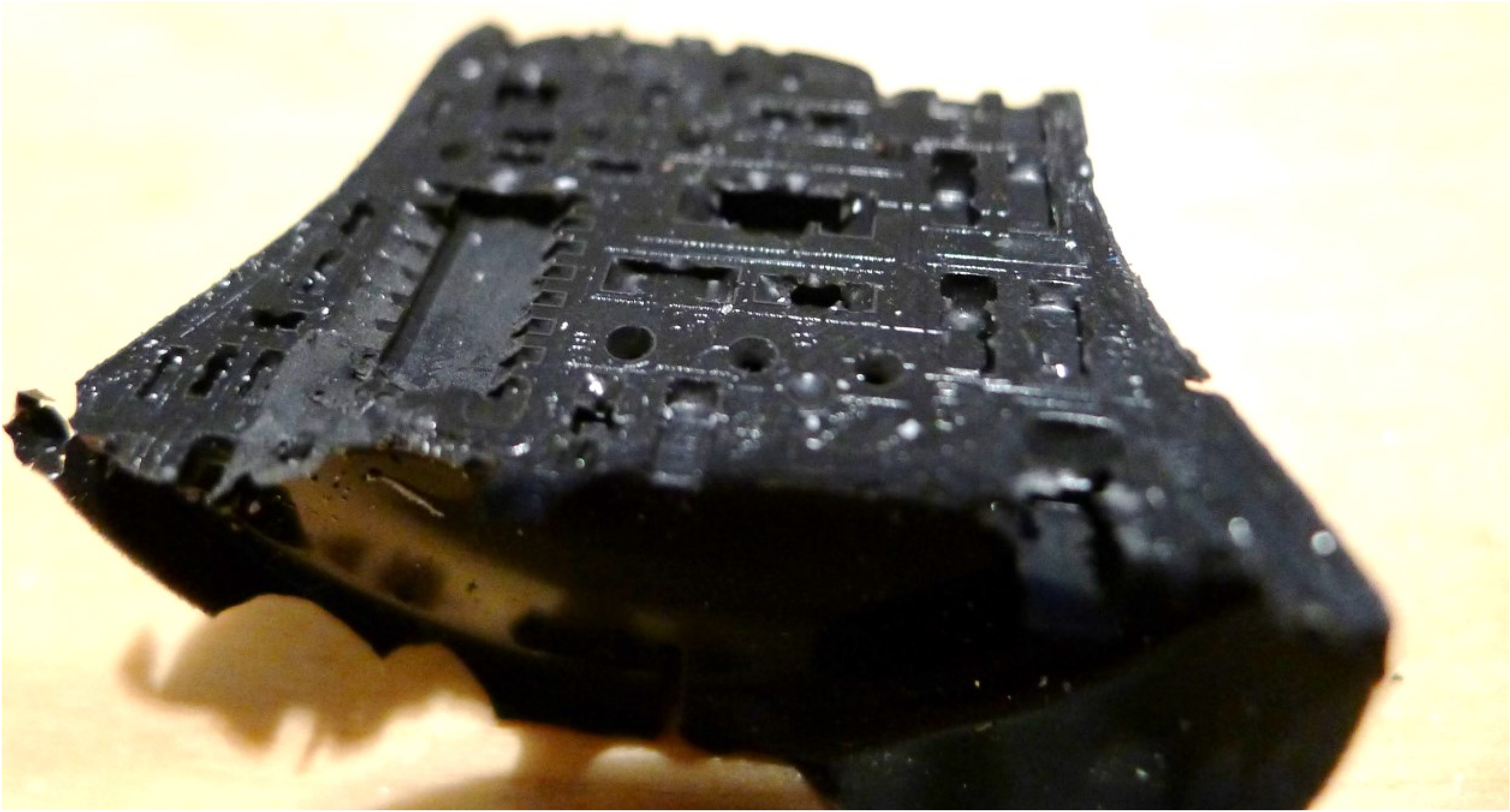

Cheap
Bulbs

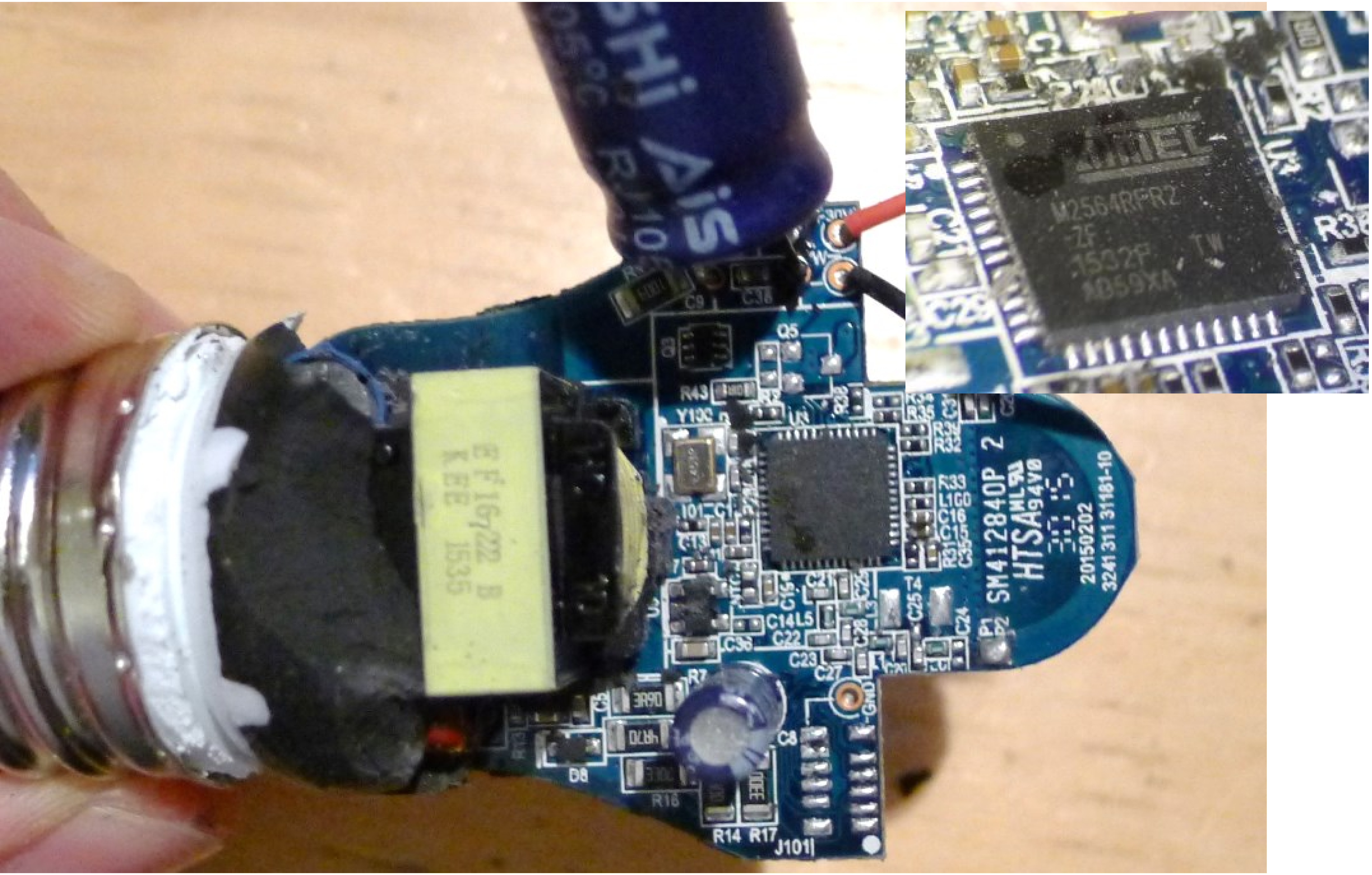

800 Lumen
Hue white A19 9.5W 90mA
110-130Vac 50/60Hz
FCC ID: O3M9290011369X
IC:10469A-1369X
Model: 9290011369

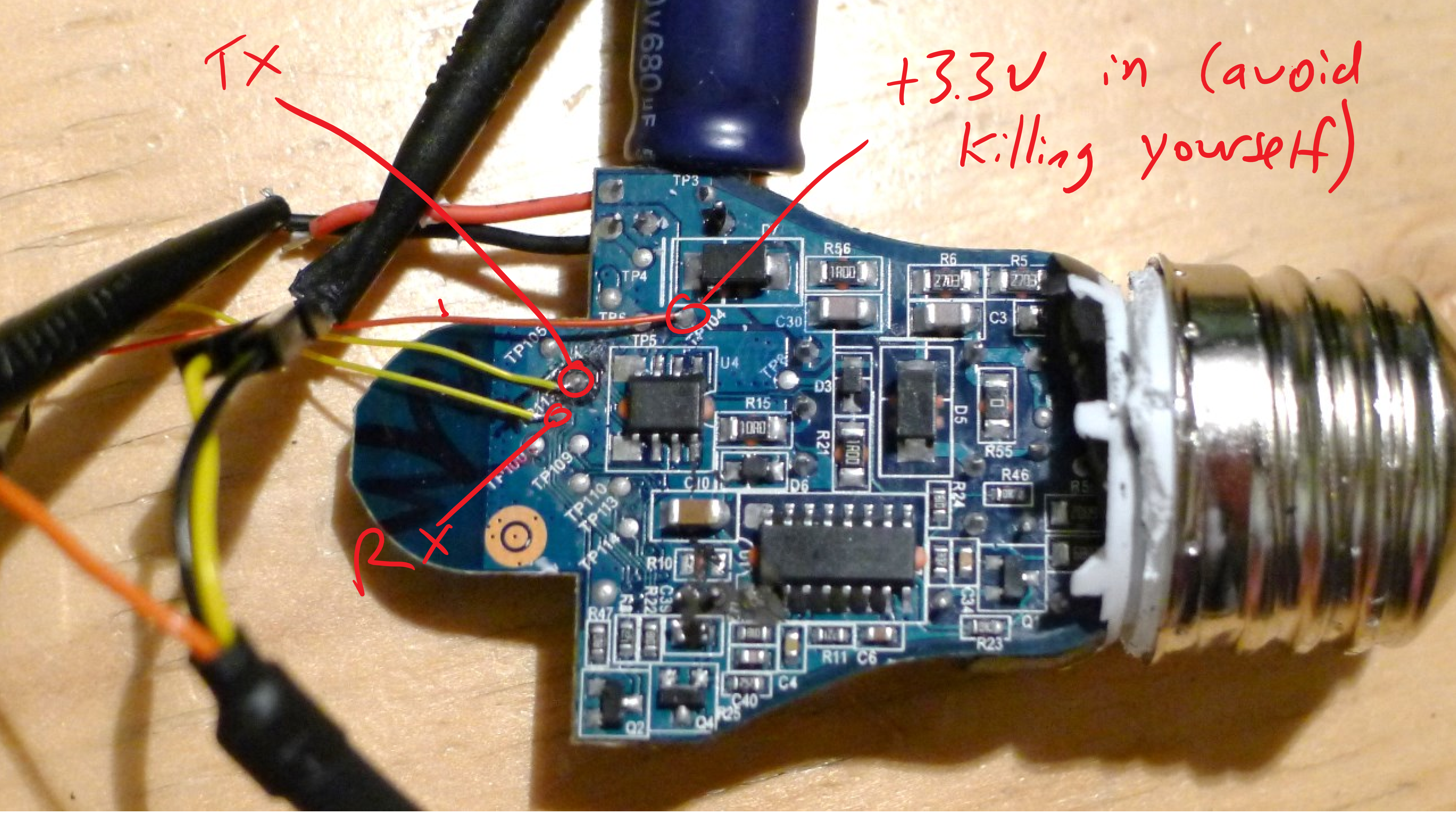

Heatsink
Antenna
800 Lumen

Rubber?






M2564RPR2
SM412840P 2
HTSA
20150202
3241 311 31181-10
R14 R17
J101


Tx
+3.3V in (avoid
killing yourself)
Rx

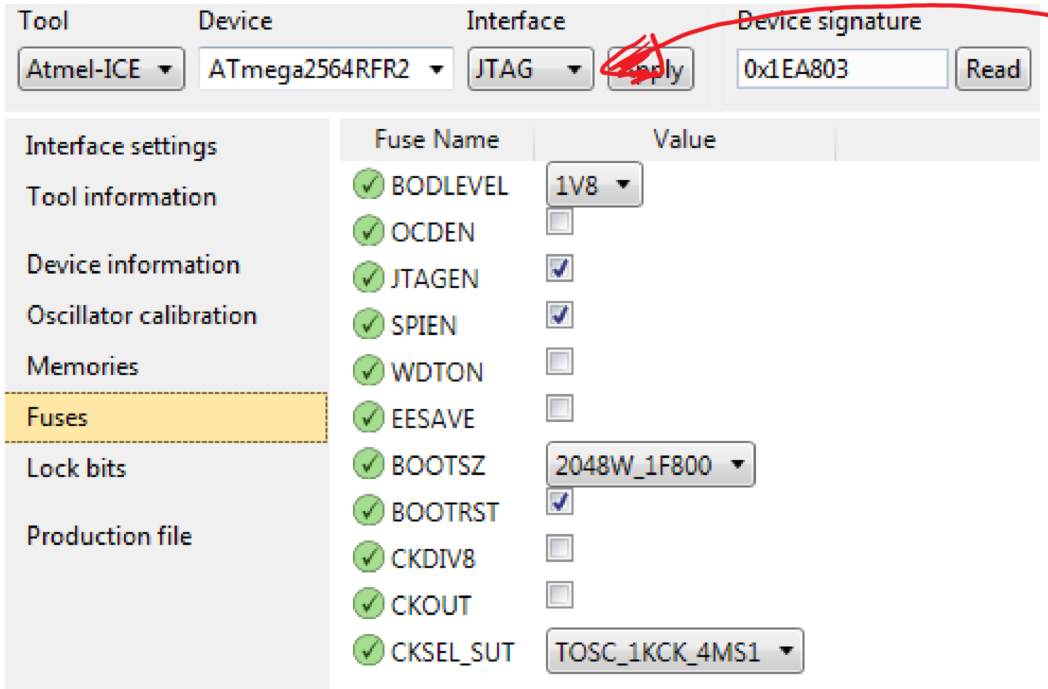
Locked

```
[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]
```

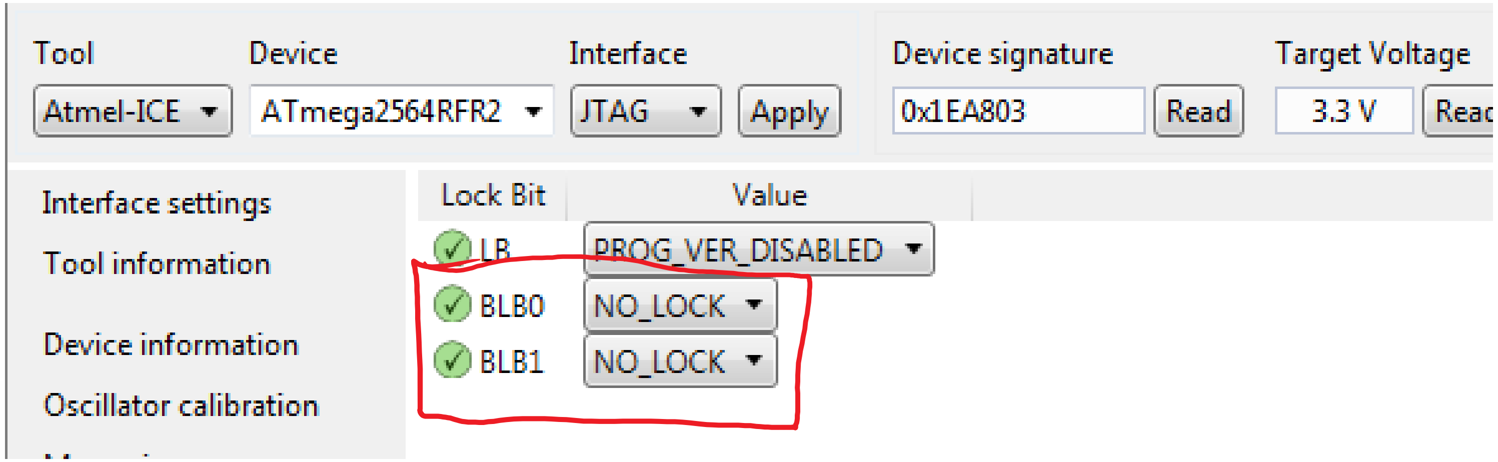
COM32 115200 bps, 8N1, no handshake Settings Clear

```
[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!



JTAG test points (see w.p.)



<u>**See white-paper for JTAG pin-out connections.**</u>

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

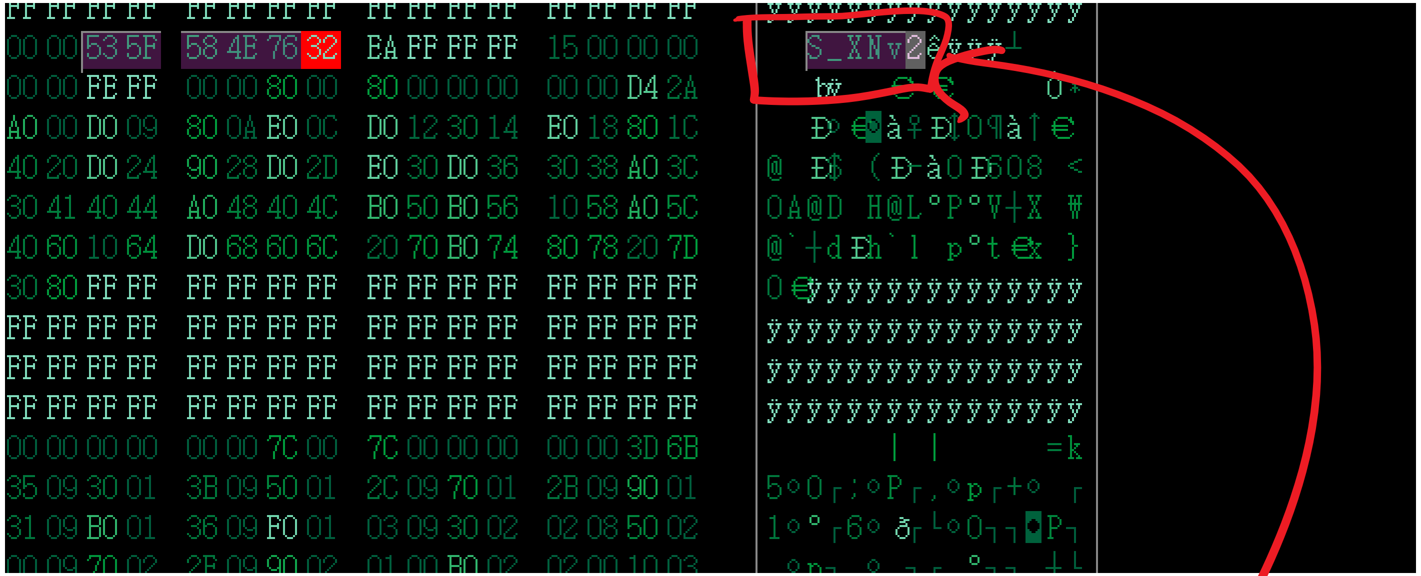
b. Can find same print statements

    [TH,Ready,0]
    [Log,Info,N_Connection,Starting discovery for updated networks]
    [Log,Info,N_Connection,Discovery for updated networks completed]

bridgeClusters.h | N_ConnectionRouter.c | D_Nv.c | S_XNv.c | S_Nv-External.c | S_Nv-ATmega.c

N_ConnectionRouter.c — C:\E\Documents\academic\sidechannel\lux_hue\BitCloud_MEGARF_3_3_0\BitCloud\C

```c
/**************************************************************************//**
  \brief After power up delay timer callback.
******************************************************************************/
static void searchNetworkTimerFired(void)
{
  uint32_t channelMask = N_DeviceInfo_GetPrimaryChannelMask();

  if (N_DeviceInfo_TrustCenterMode_Distributed != N_DeviceInfo_GetTrustCenterMode())
    channelMask |= N_DeviceInfo_GetSecondaryChannelMask();

  CS_WriteParameter(CS_CHANNEL_MASK_ID, &channelMask);
  memset((uint8_t*)networkSearchBeacon.extendedPanId, 0x00, sizeof(ExtPanId_t));

  N_LOG_ALWAYS(("Starting discovery for updated networks"));

  // Begin network discovery
  N_Cmi_NetworkDiscovery(channelMask,
                         N_Beacon_Order_60ms,
                         &networkSearchBeacon,
                         N_Connection_AssociateDiscoveryMode_OnlyThisPan,
                         N_DeviceInfo_GetNetworkExtendedPanId(),
                         BEACON_FILTERING_CRITERION_NETWORK_SEARCH,
                         NULL,
                         searchNetworkDiscoveryDone);
}

/**************************************************************************//**
```




```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;
```

Damn.

December 03, 2014

Lamp software version: 66013452

- Related products are hue A19 and BR30 downlight bulbs and Friends of hue
- hue Tap range is extended if lamp in between Tap and bridge is powered
- Faster start-up when using the wall switch
- Bug fixes and stability improvements

NOT Atmel based.

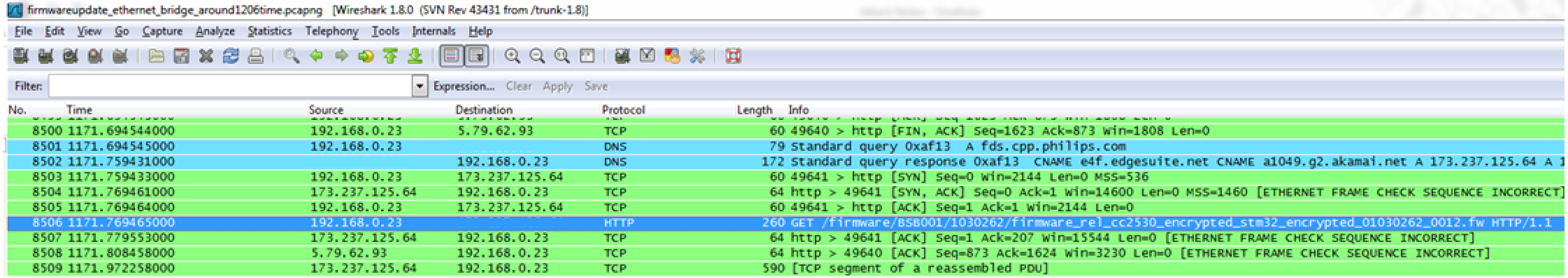

BRIDGE
1.0

BRIDGE 1.0 HACKING

Bridge

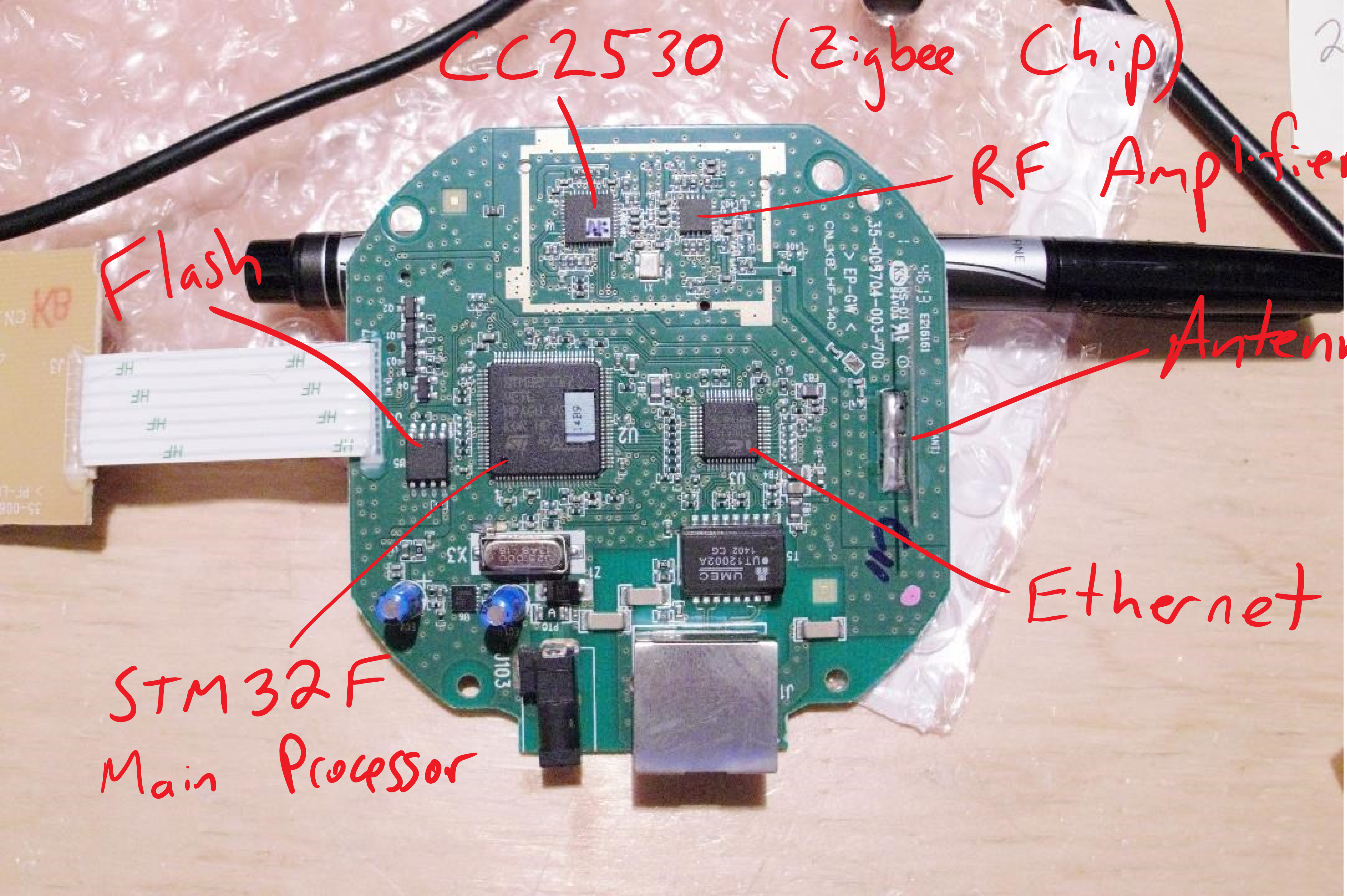
Dumb hub

Run App



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

TP46
TP1
TP6
TP14
TP41
TP24
TP25
TP22
TP23
TP26
TP38
TP40
TP21
TP19
TP27
TP2
TP34
TP37
TP7
TP35
TP5
TP17
TP3
TP20
TP419
TP39
TP4
TP28
TP31
SW1
TP10
TP29
TP15
35-0087C4
> EP-GW <
20706 105C 60V VW-1 -HF-
20706 105C 60V VW-1 -HF-
20706 105C 60V VW-1 -HF-
20706 105C 60V VW-1 -HF-

Output from CC2530

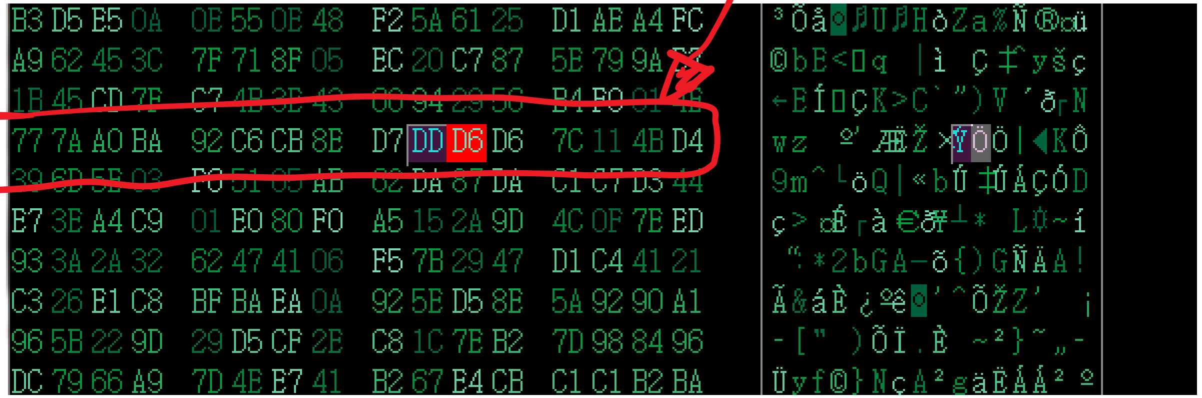
```
[Log,Info,S_DeviceInfo,Booting into normal mode...]
[Log,Info,S_DeviceInfo,DeviceId: IpBridge]
[Log,Info,N_Security,LIB4.4.52]
[Log,Info,N_Security,KeyBitMask,0x0012]
[Log,Info,A_Bridge,Platform version 0.25.0,package_ZigBee 8720,package_Z_Stack
8720,built by LouvreZLL]
[Log,Info,A_Bridge,Product 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...]
```

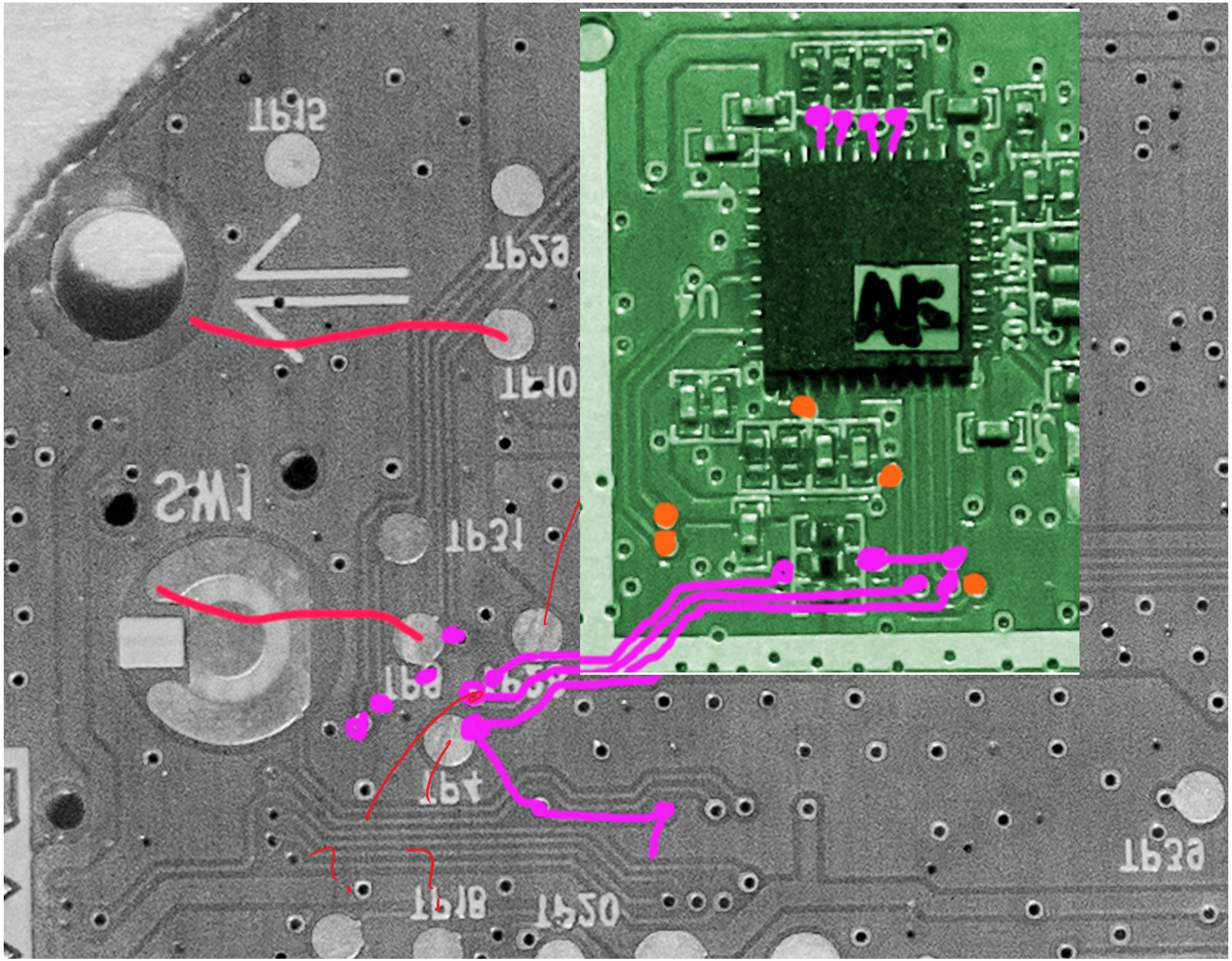
Input to CC2530

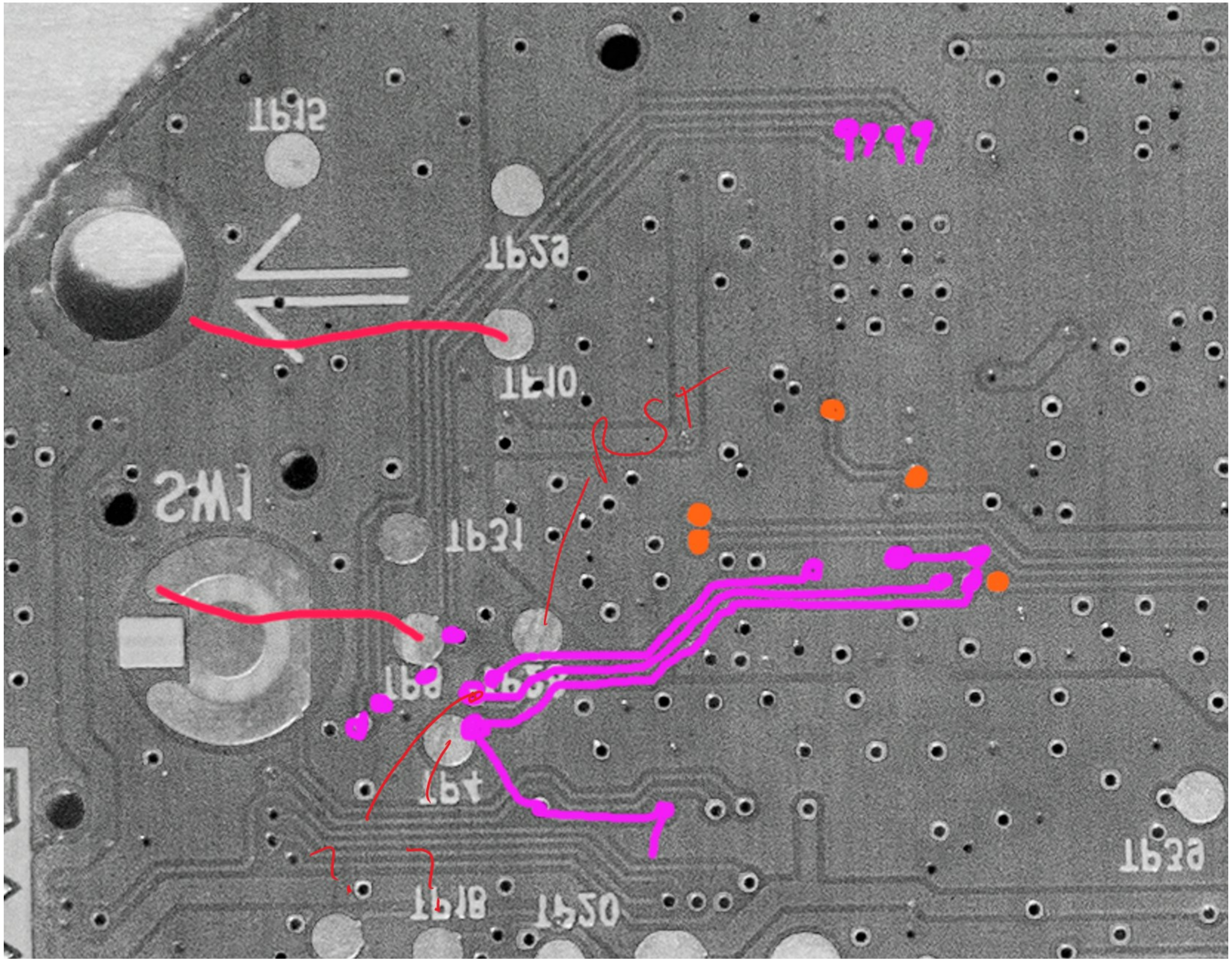
```
[Connection,GetAddress]
[Bridge,StoreGroupRange,0x5357,0x5367]
[Zcl,S,S=0x0002.11,6,0000000000]
[Routing,ClearEntry,1]
[Routing,SendMtoRR,True]
[Zcl,S,S=0x0003.11,6,0001000000]
[Routing,ClearEntry,2]
[Routing,SendMtoRR,True]
[Zcl,S,S=0x0002.11,6,0002000000]
[Zcl,S,S=0x0003.11,6,0003000000]
[Zcl,S,S=0x0002.11,6,0004000000]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

BYPASS ZLL KEY?

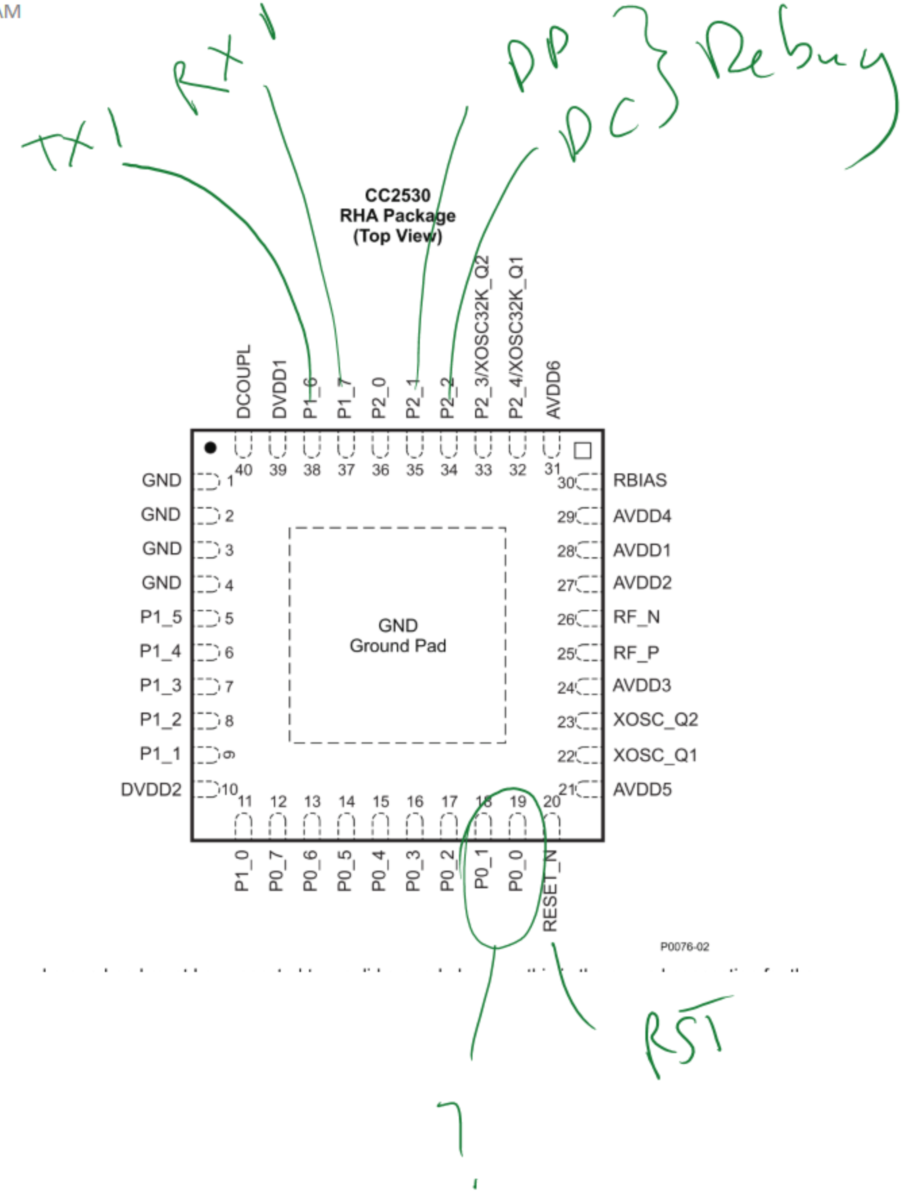
[zcl,s,s=0x0004.11,6,001a000000,64]
[zcl,s,s=0x0004.11,8,001b000000,64]
[zcl,s,s=0x0004.11,768,001c0001000040024003000040007000800,64]
[zcl,s,s=0x0004.11,25,190d05000b1000018c340042ae0100002b97098fac0320f2f31e3c8fd035a34097da5018feb50a2e8b40d3678aa57c866a47122020a3a86220a25c93,64]
[zcl,s,s=0x0004.11,25,190e05000b1000018c340042d90100002b2d3d4e5d25c0622af60856c62900d59f71b104541e744b3657ebc32286f3e635474145d3189f7deca60cd9,64]
[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=0x0004.11,25,191105000b1000018c3400425a0200002b2956b4f0014e77aa0ba92c6cb8ed7ddd6d67c114bd4396d5e03f65105ab62da87dac1c7d344e73ea4c901,64]
[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=0x0004.11,25,191505000b1000018c34004206030000 2be67a19318a005a40204ccbc01269519827 09f080f5806e33d478efd8dcda9e79303ad662ddcfa822316b03,64]



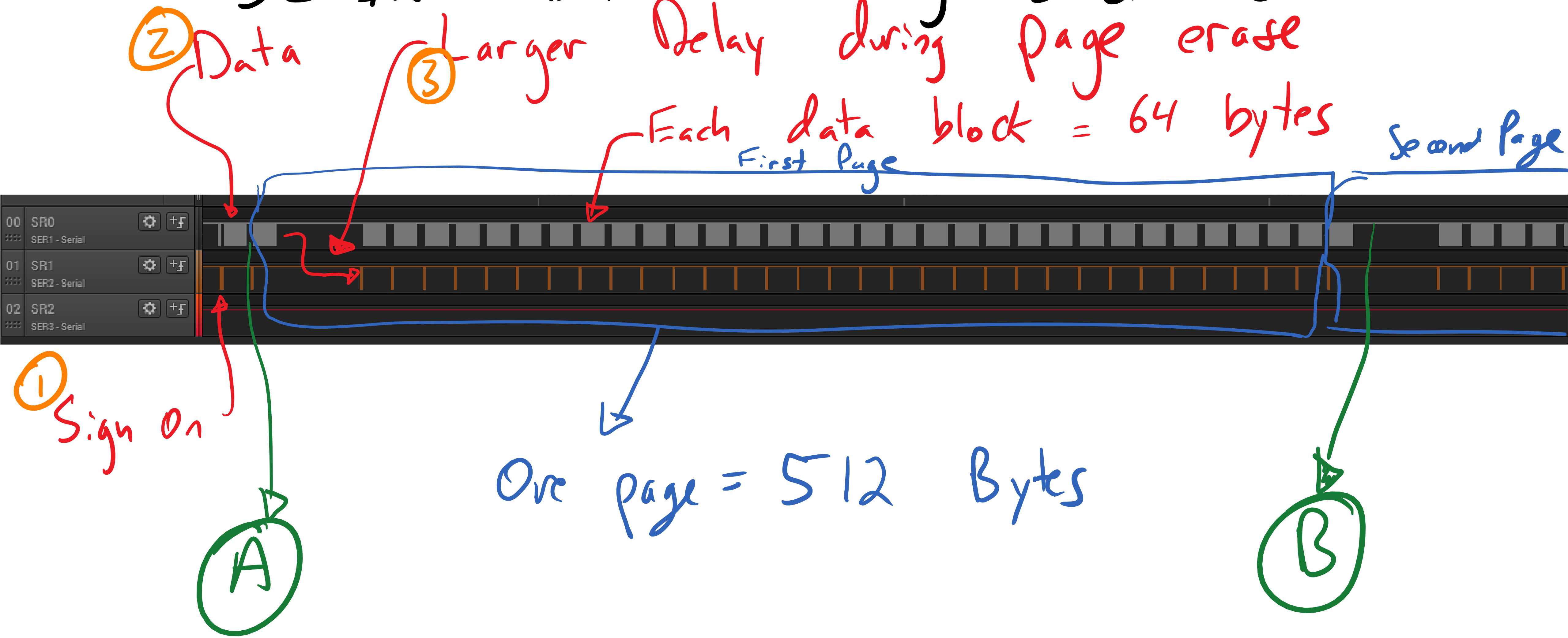



TP15
TP29
TP10
SW1
TP31
TP4
TP18
TP20
TP39
RST

:41 AM


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
RBIAS
AVDD4
AVDD1
AVDD2
RF_N
RF_P
AVDD3
XOSC_Q2
XOSC_Q1
AVDD5
GND
Ground Pad
P1_0
P0_7
P0_6
P0_5
P0_4
P0_3
P0_2
P0_1
P0_0
RESET_N
P0076-02
TX1
RX1
DD
DC
Debug
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

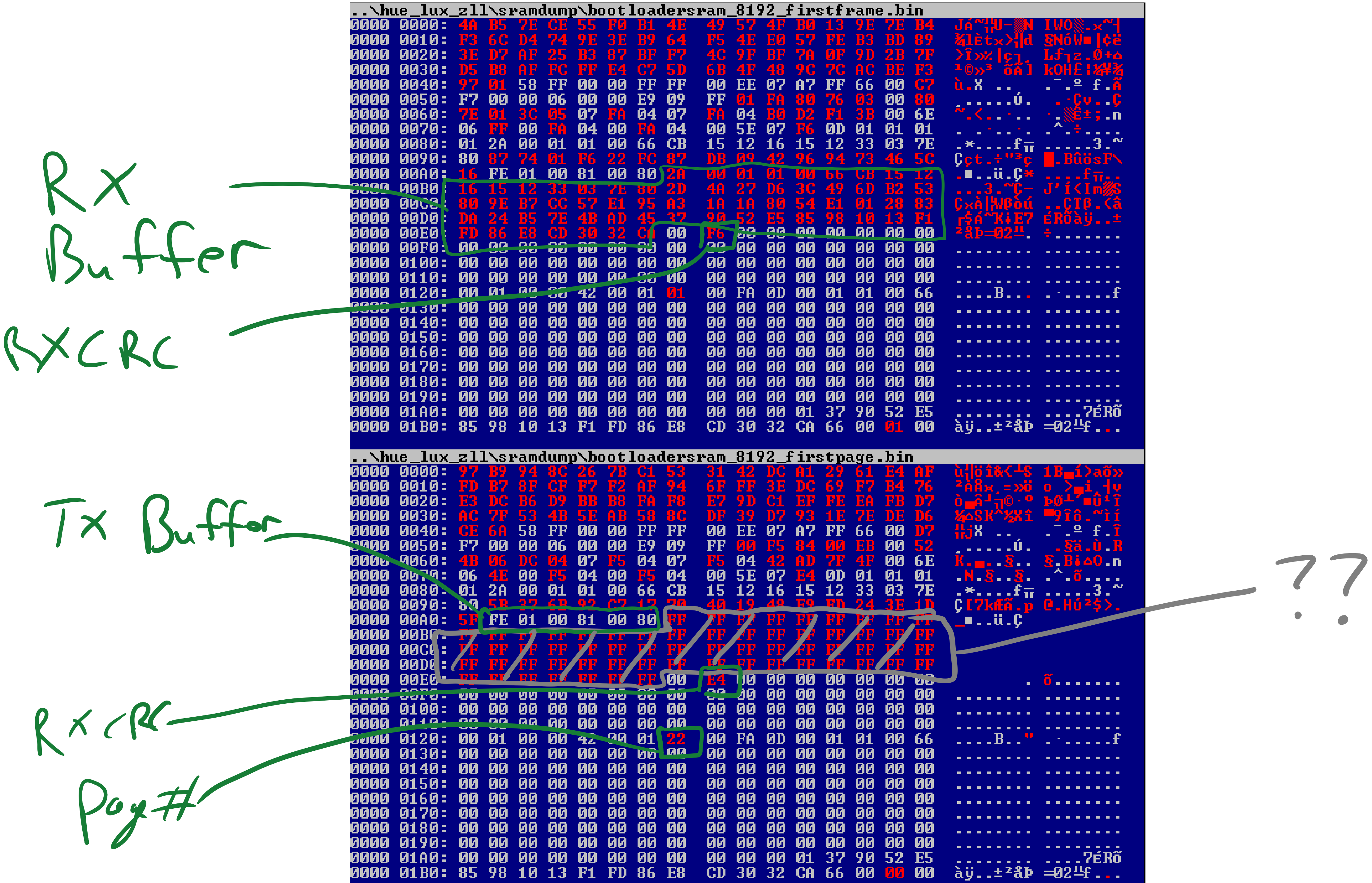
The Ember EM250 contains a 16-bit XAP2b microprocessor from Cambridge Consultants Ltd.[3] Debugging support is provided by that firm's proprietary SIF protocol, with hardware and software available only through Ember. SIF itself is a variant of JTAG.

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

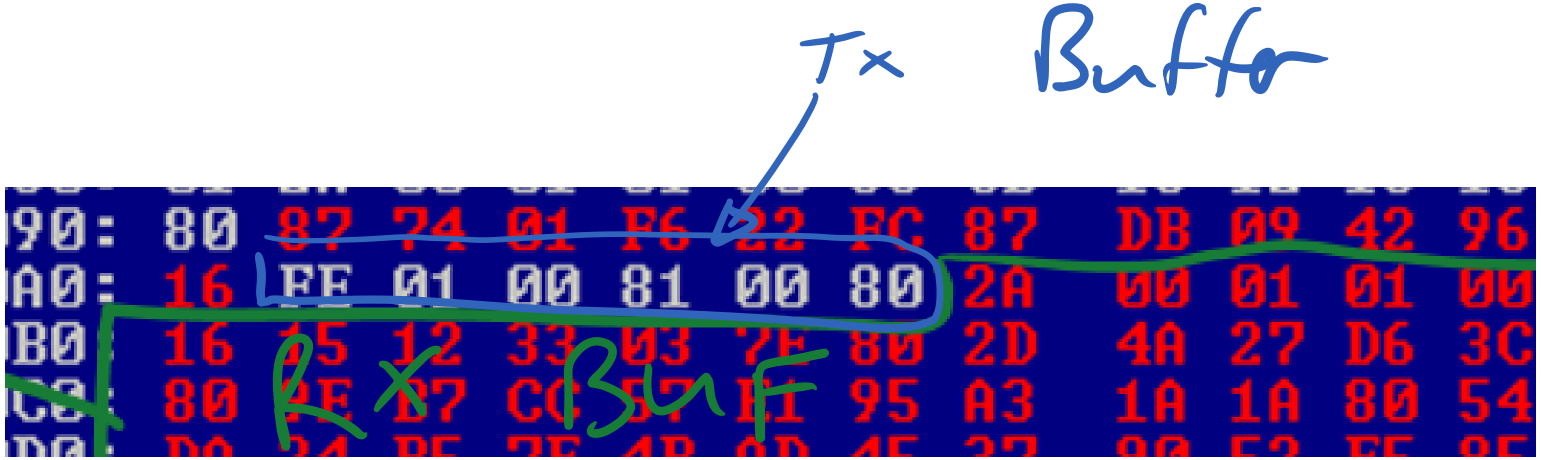
- ZLL master key <u>not</u> in regular SRAM
- Tried AES-128 CBC to decrypt bootloader image, where key = {every possible 16-byte block}

↳ No sucess, key not in SRAM?


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

TX Buffer Attack

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

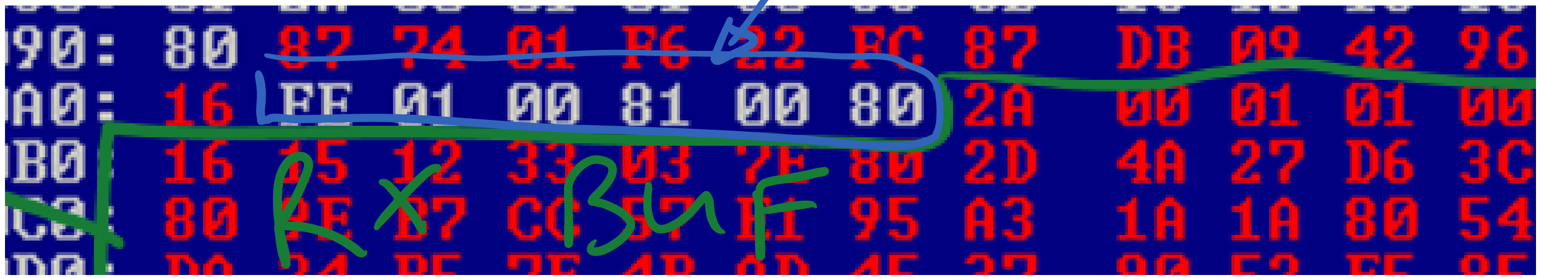


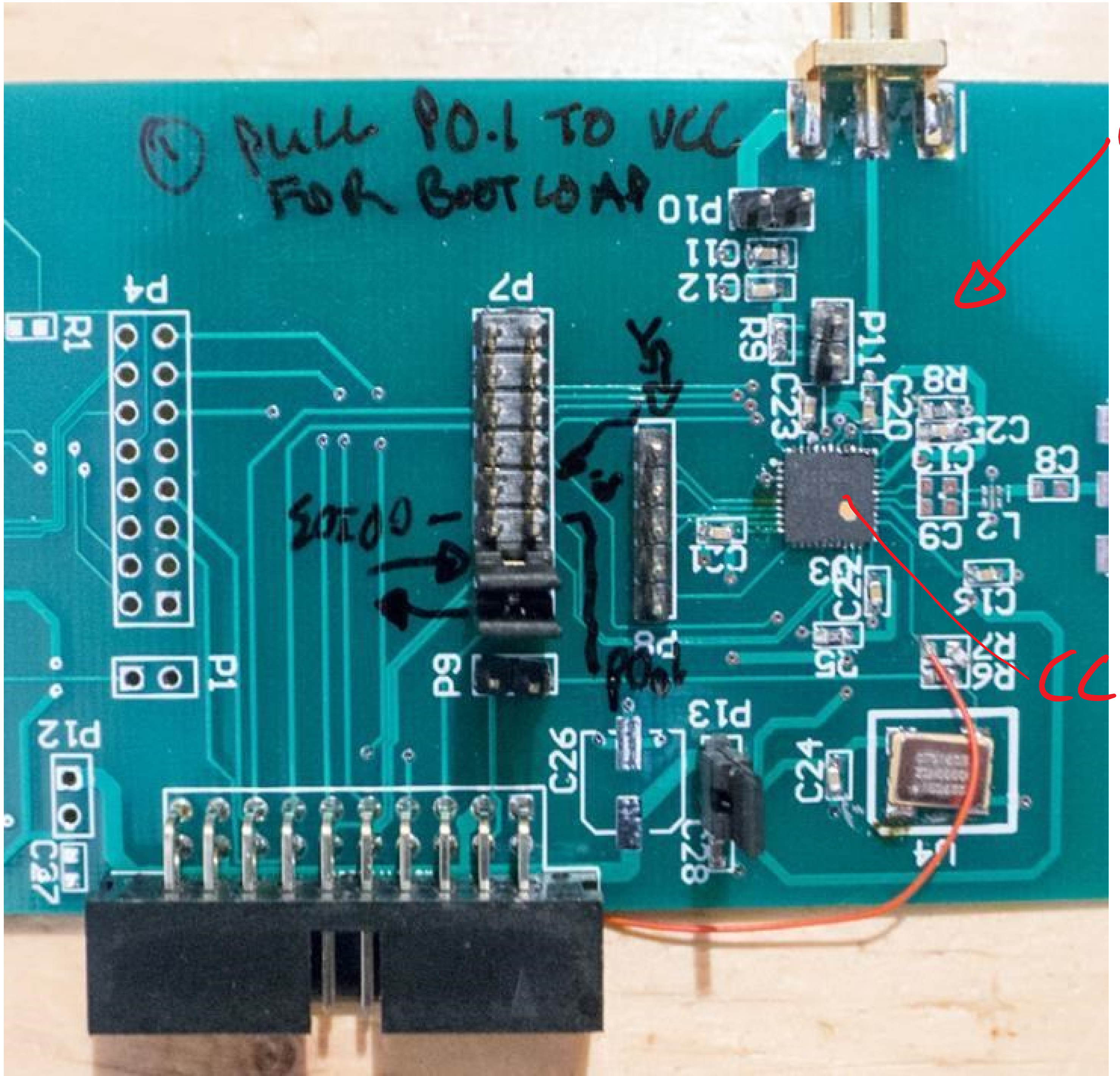
TX BUFFER ATTACK

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

Glitch Attack!

Tx Buffer

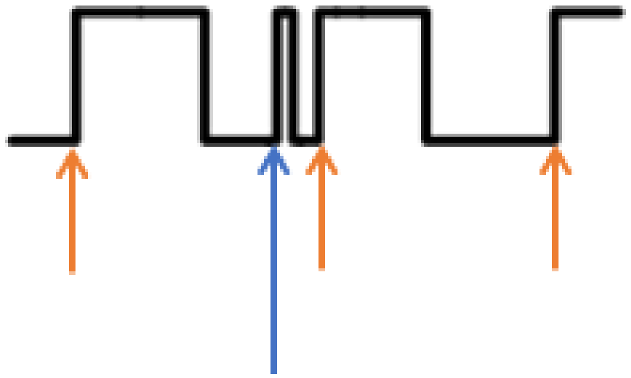


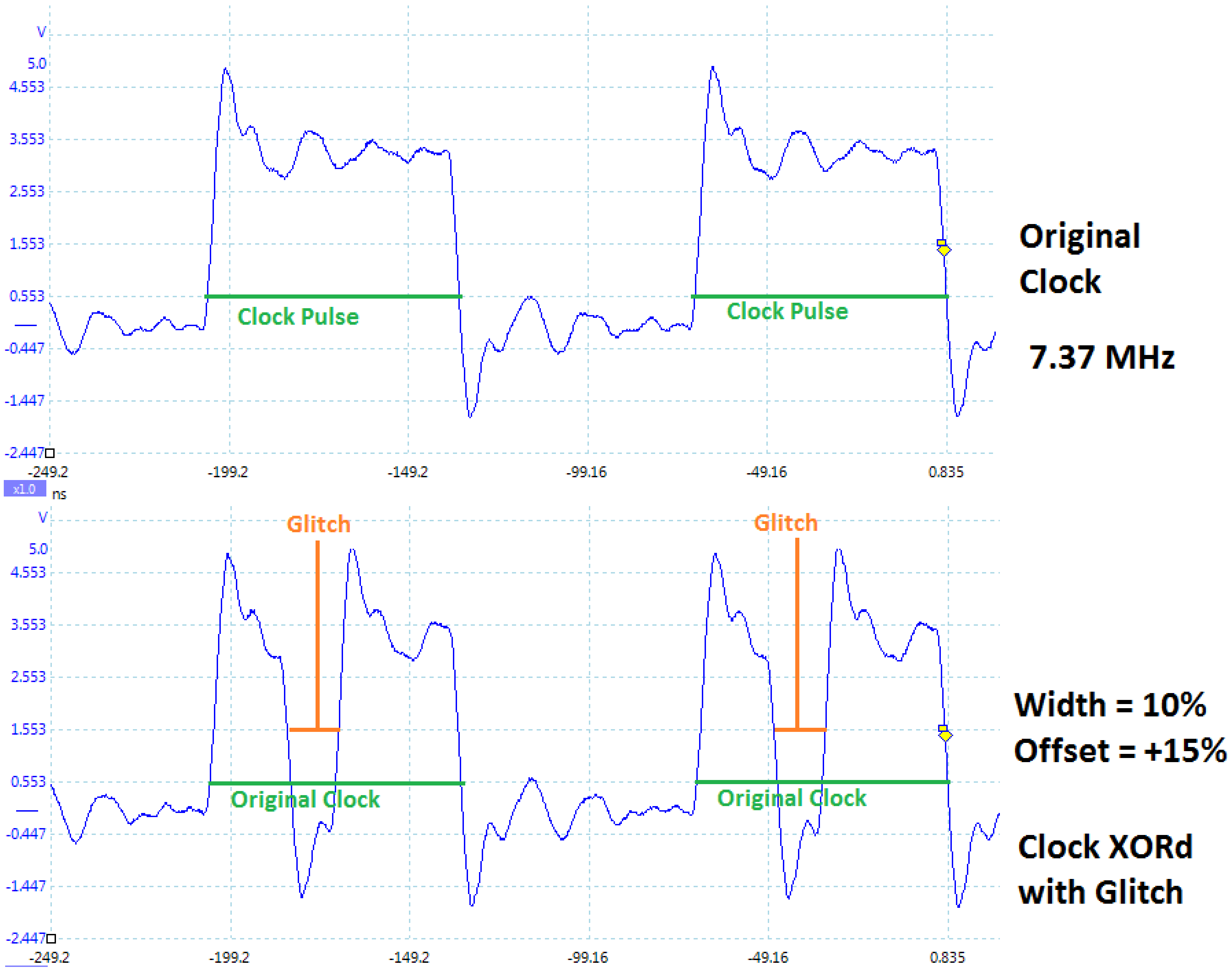


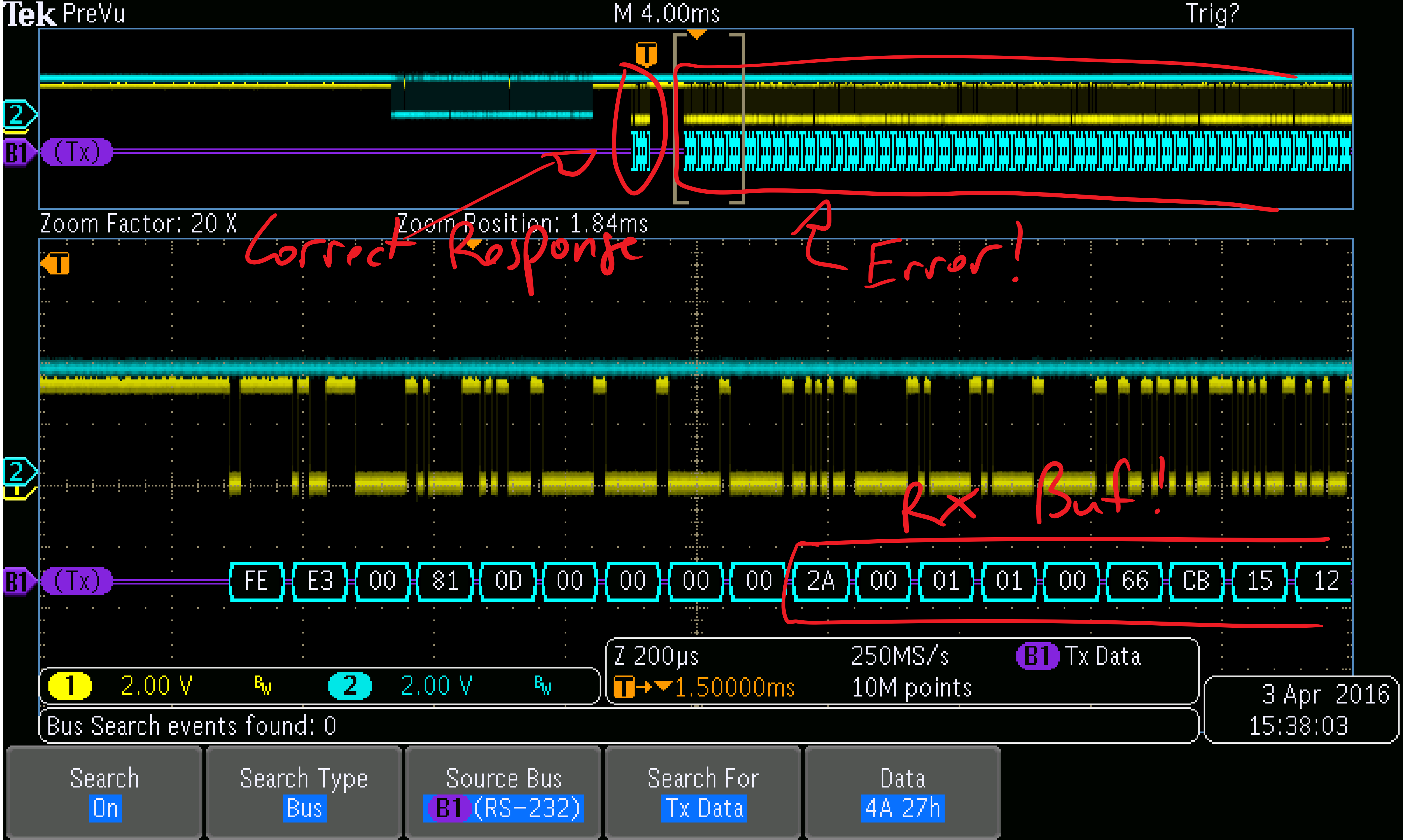
Custom PCB

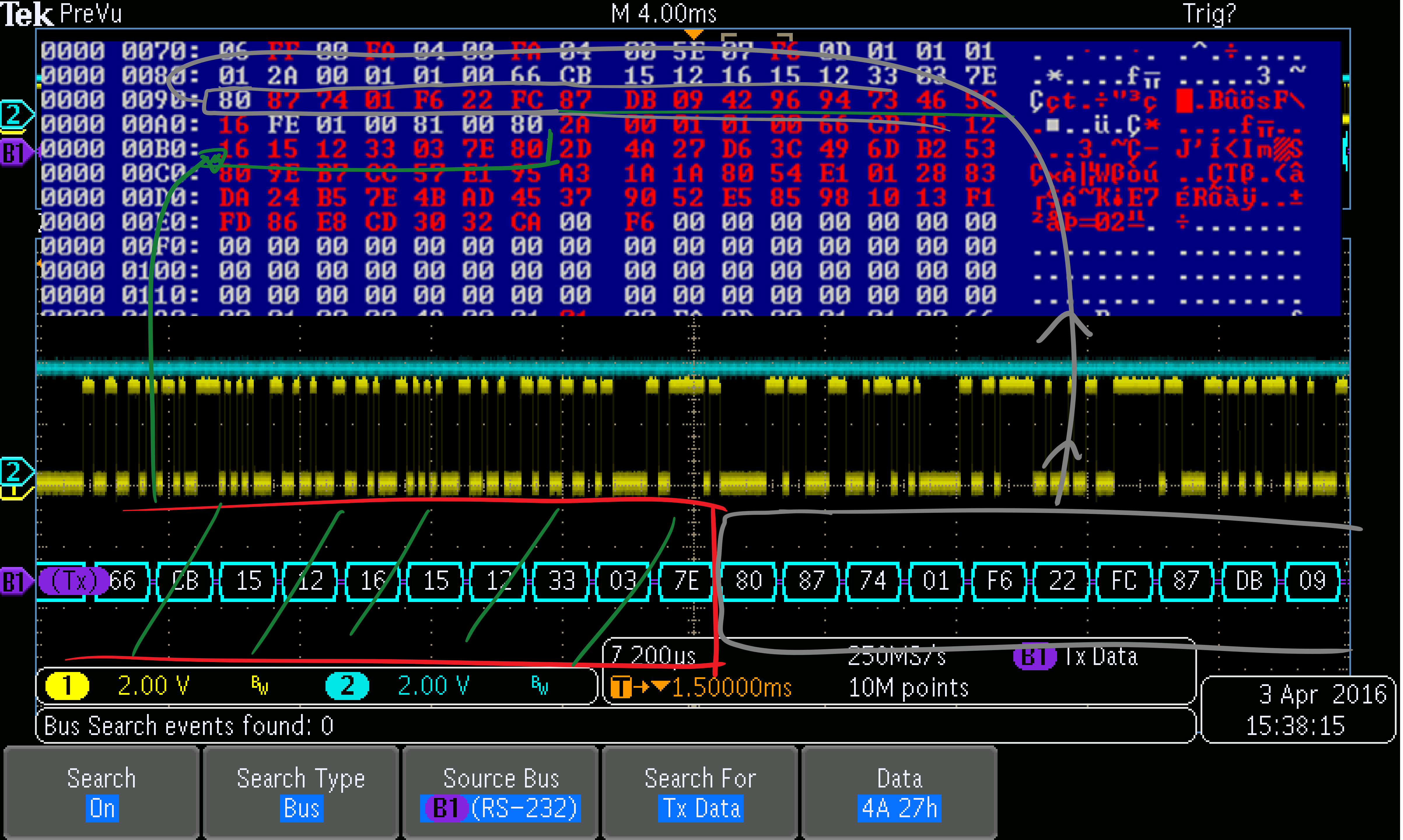
CC2530 from Bridge

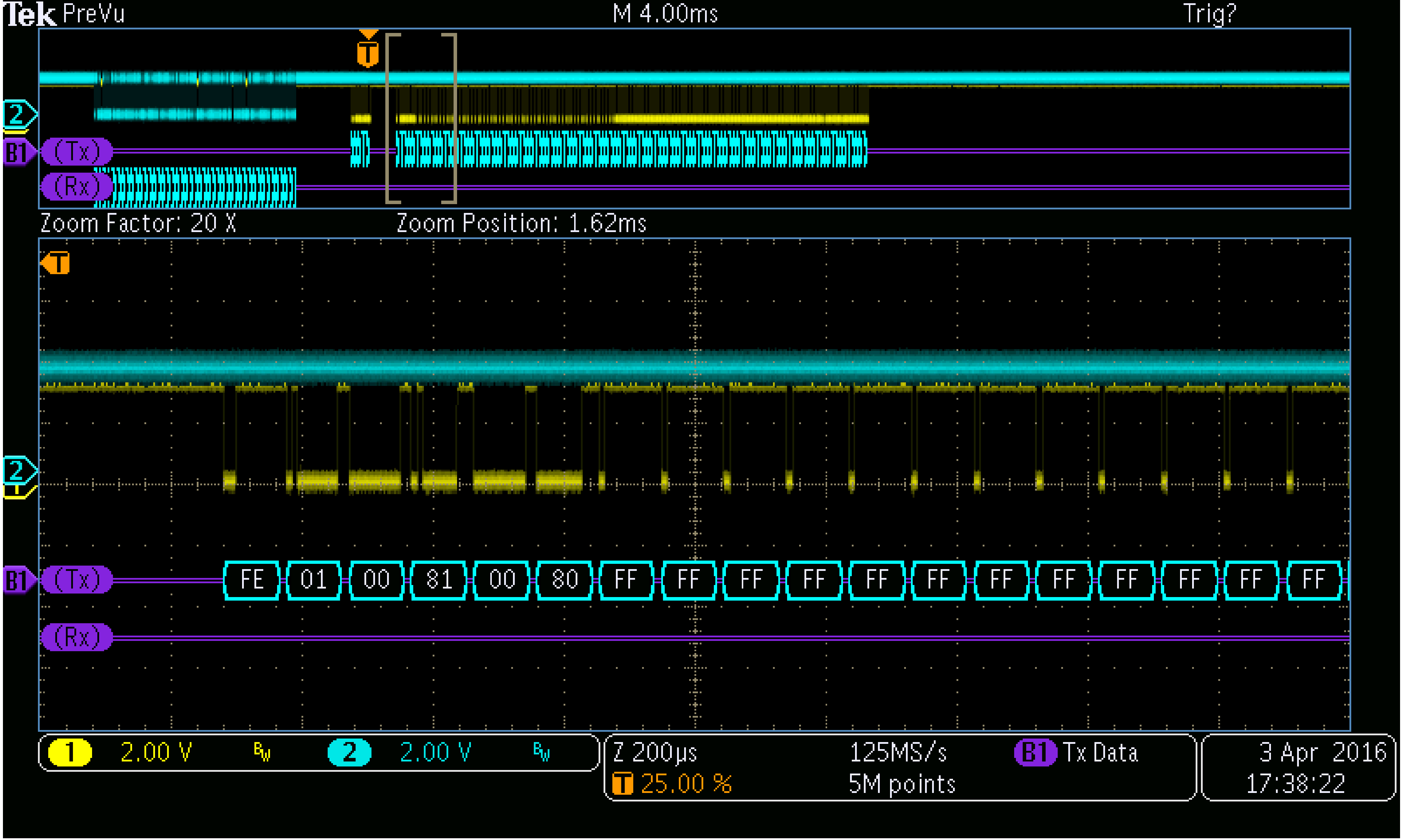
Clock Glitching

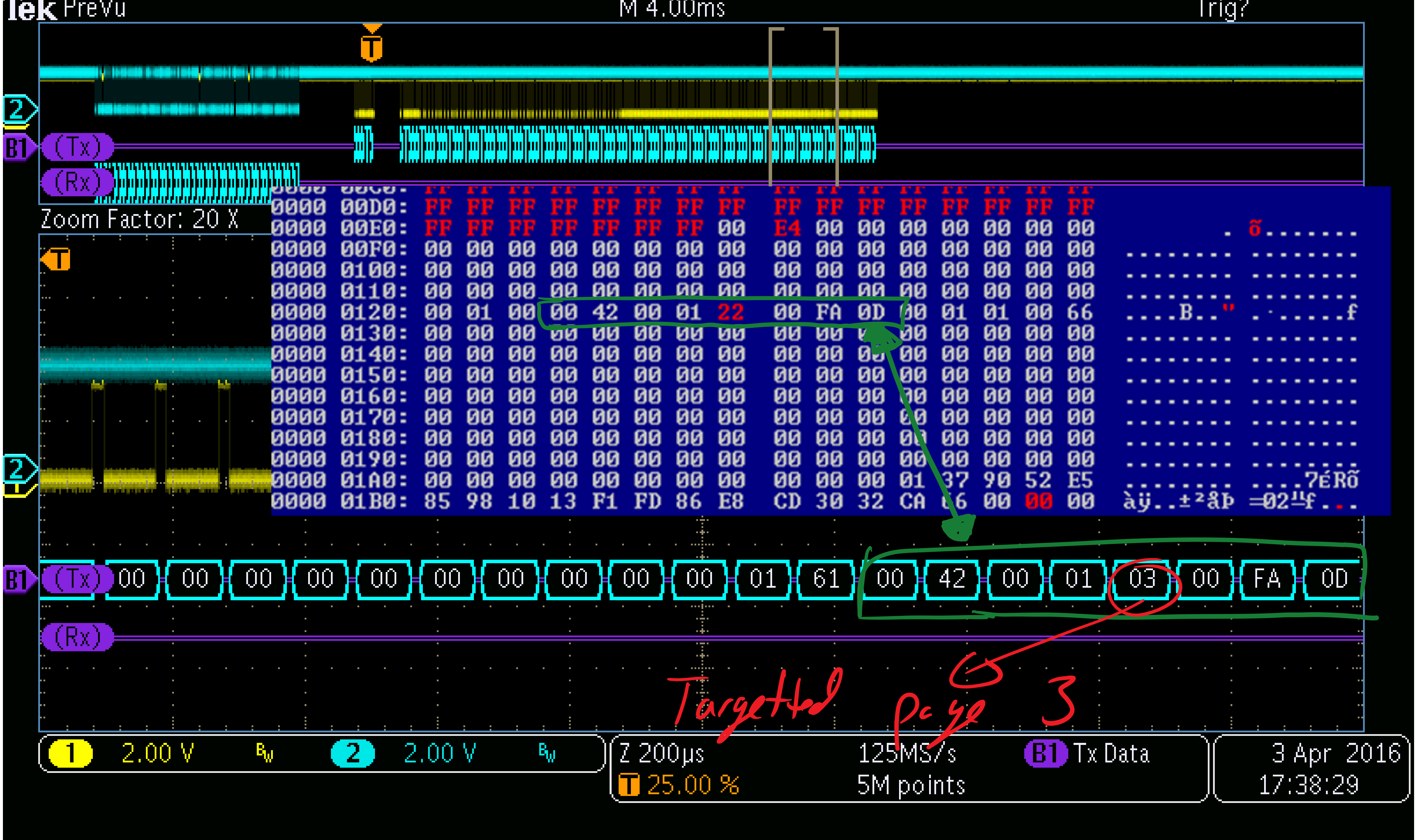



Original Clock
7.37 MHz
Clock Pulse
Clock Pulse
Glitch
Glitch
Original Clock
Original Clock
Width = 10%
Offset = +15%
Clock XORd with Glitch
V
ns
x1.0
5.0
4.553
3.553
2.553
1.553
0.553
-0.447
-1.447
-2.447
-249.2
-199.2
-149.2
-99.16
-49.16
0.835


Tek PreVu
M 4.00ms
Trig?
2
B1 (Tx)
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
Z 200µs
250MS/s
B1 Tx Data
1.50000ms
10M points
1 2.00 V
2 2.00 V
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
B1 (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
2 2.00 V
T→1.50000ms
10M points
3 Apr 2016
15:38:15
Bus Search events found: 0
Search On
Search Type Bus
Source Bus B1 (RS-232)
Search For Tx Data
Data 4A 27h


Tek PreVu
M 4.00ms
Trig?
(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 FF
1 2.00 V
2 2.00 V
Z 200µs
25.00 %
125MS/s
5M points
B1 Tx Data
3 Apr 2016
17:38:22


Tek PreVu
M 4.00ms
Trig?
(Tx)
(Rx)
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
(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
2.00 V
2.00 V
Z 200µs
25.00 %
125MS/s
5M points
B1 Tx Data
3 Apr 2016
17:38:29

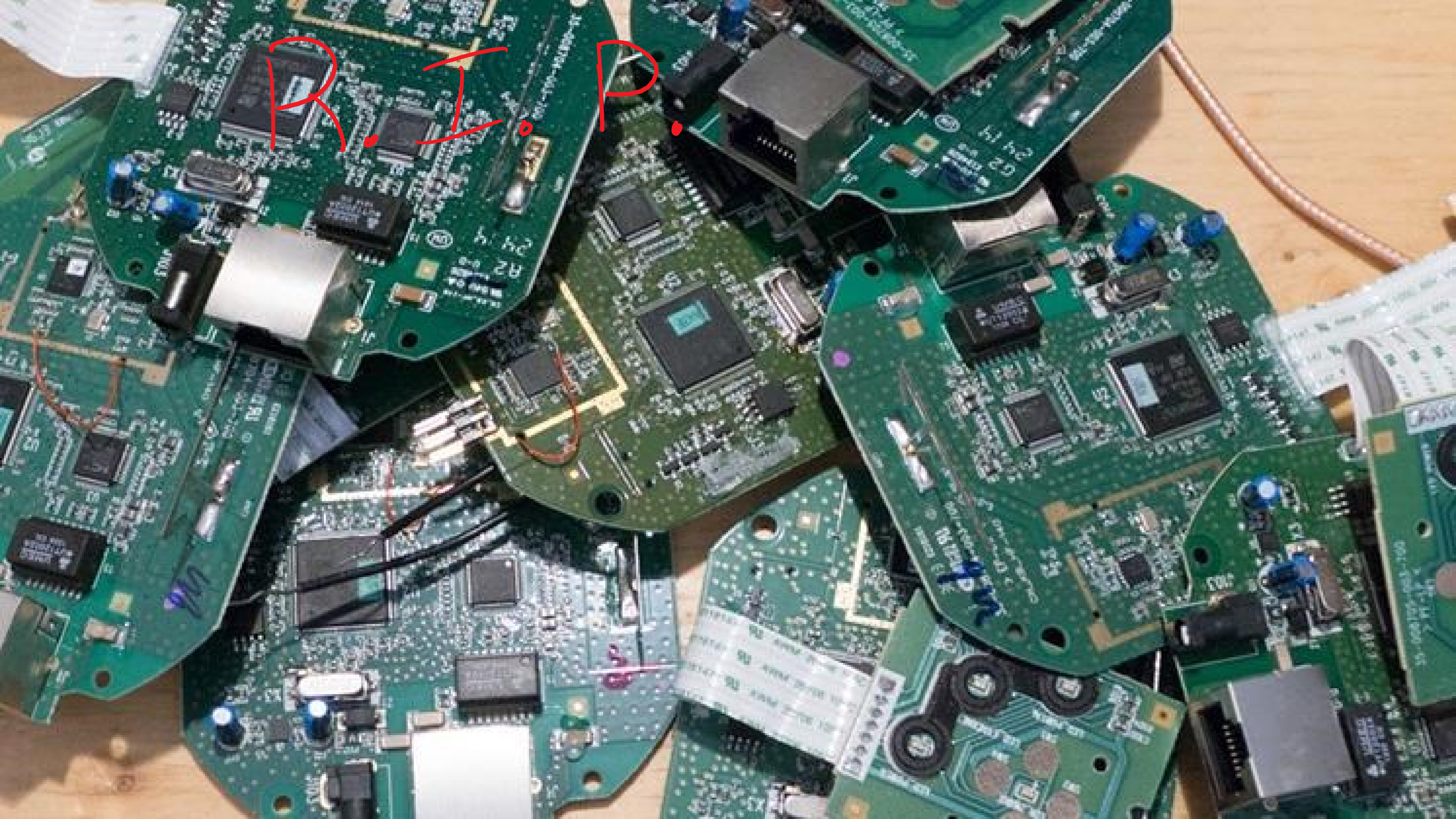
Appears section of SRAM is erased after use.

↳ This is $\boxed{\text{good}}$ practice!

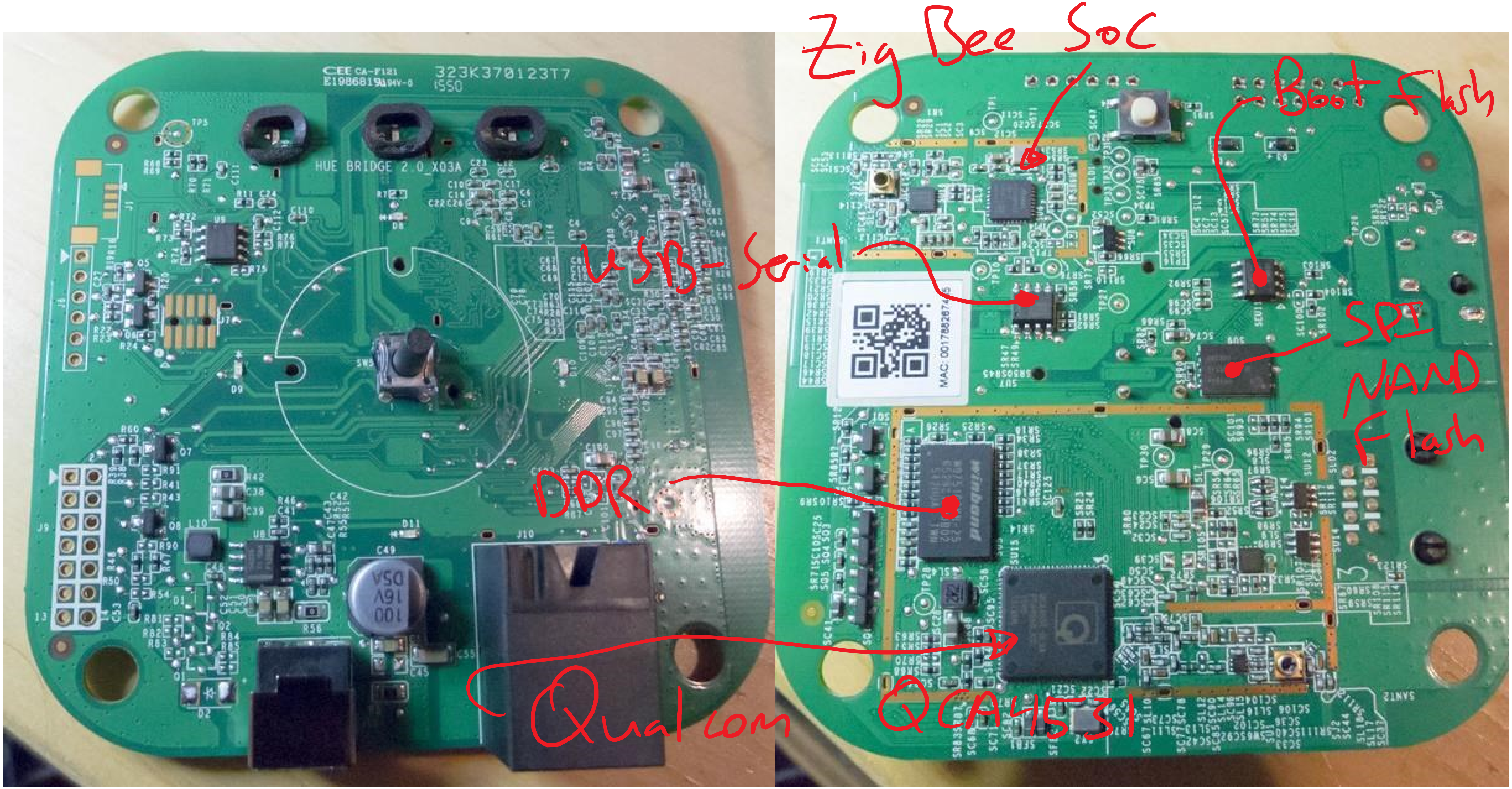
↳ May be possible with more glitches.

Glitch Attacks To Firmware

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

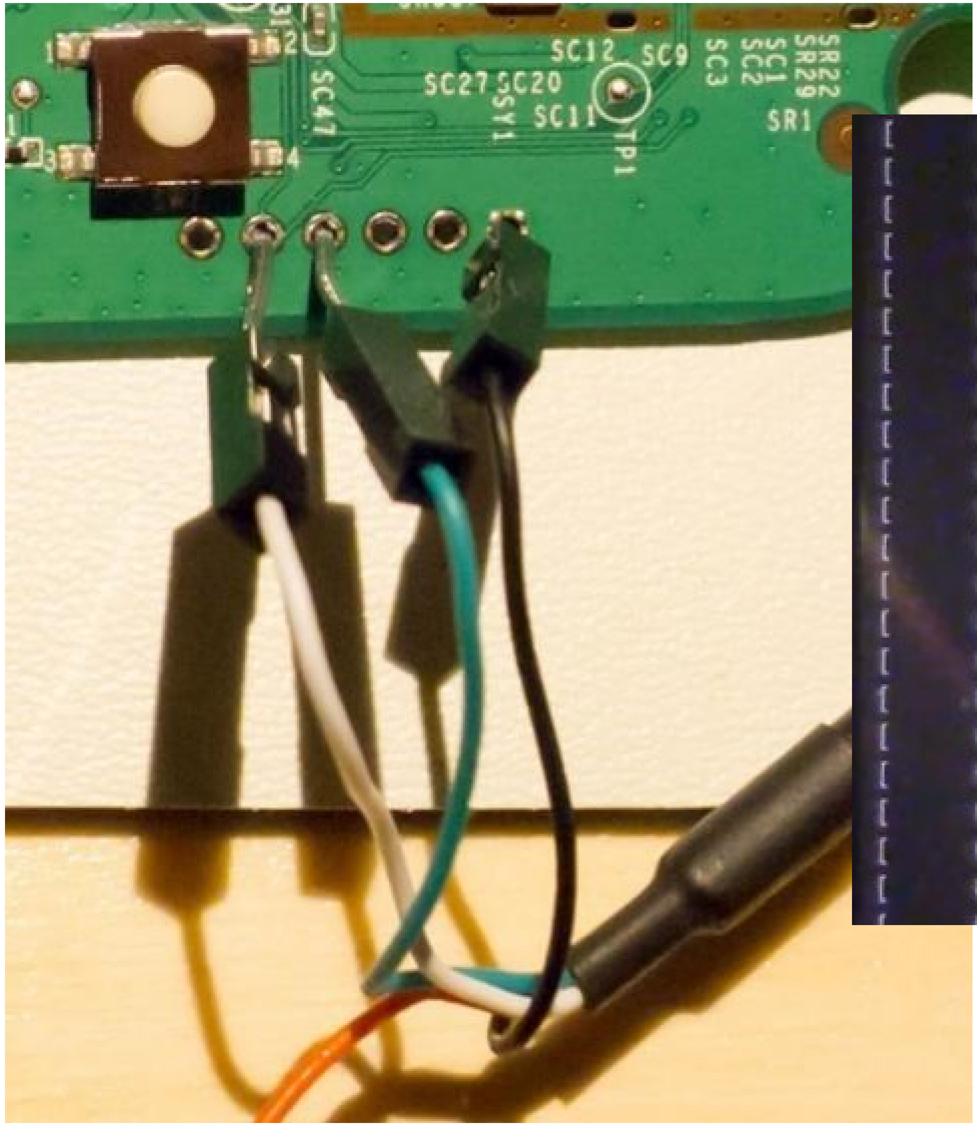

R.I.P.

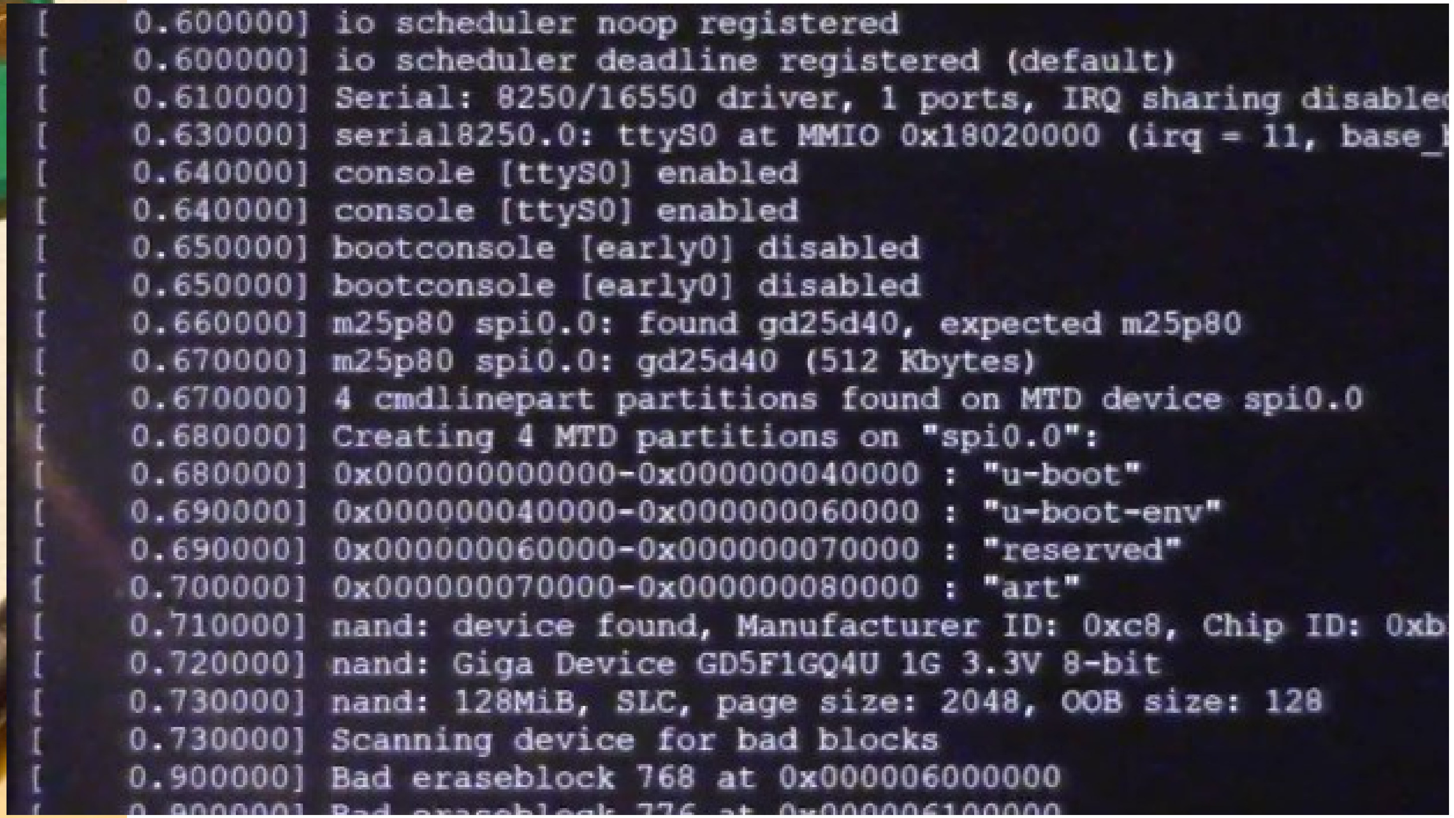

BRIDGE
2.0



Zig Bee SoC
Boot Flsh
USB-Serial
SPI NAND Flash
DDR
Qualcom
QCA4531
HUE BRIDGE 2.0 X03A
323K370123T7

HACKING TOOLS

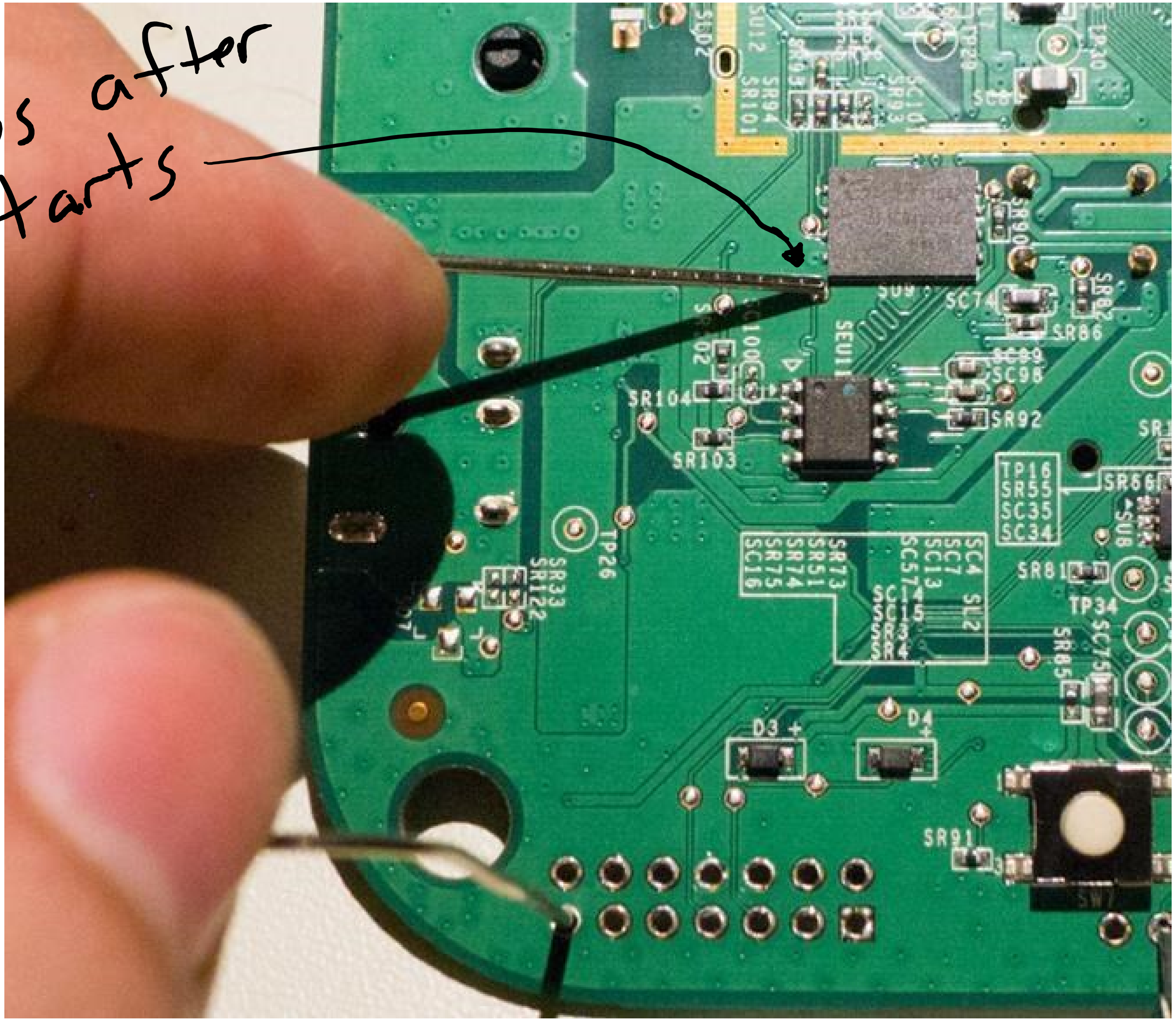




https://www.youtube.com/watch?v=hi2D2MnwiGM

Or: http://www.oflynn.com

Short ~3s after boot 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
```

<u>*****COPY THE DEFAULT VALUE THAT WAS PRINTED & SAVE THIS SOMEWHERE****</u>

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

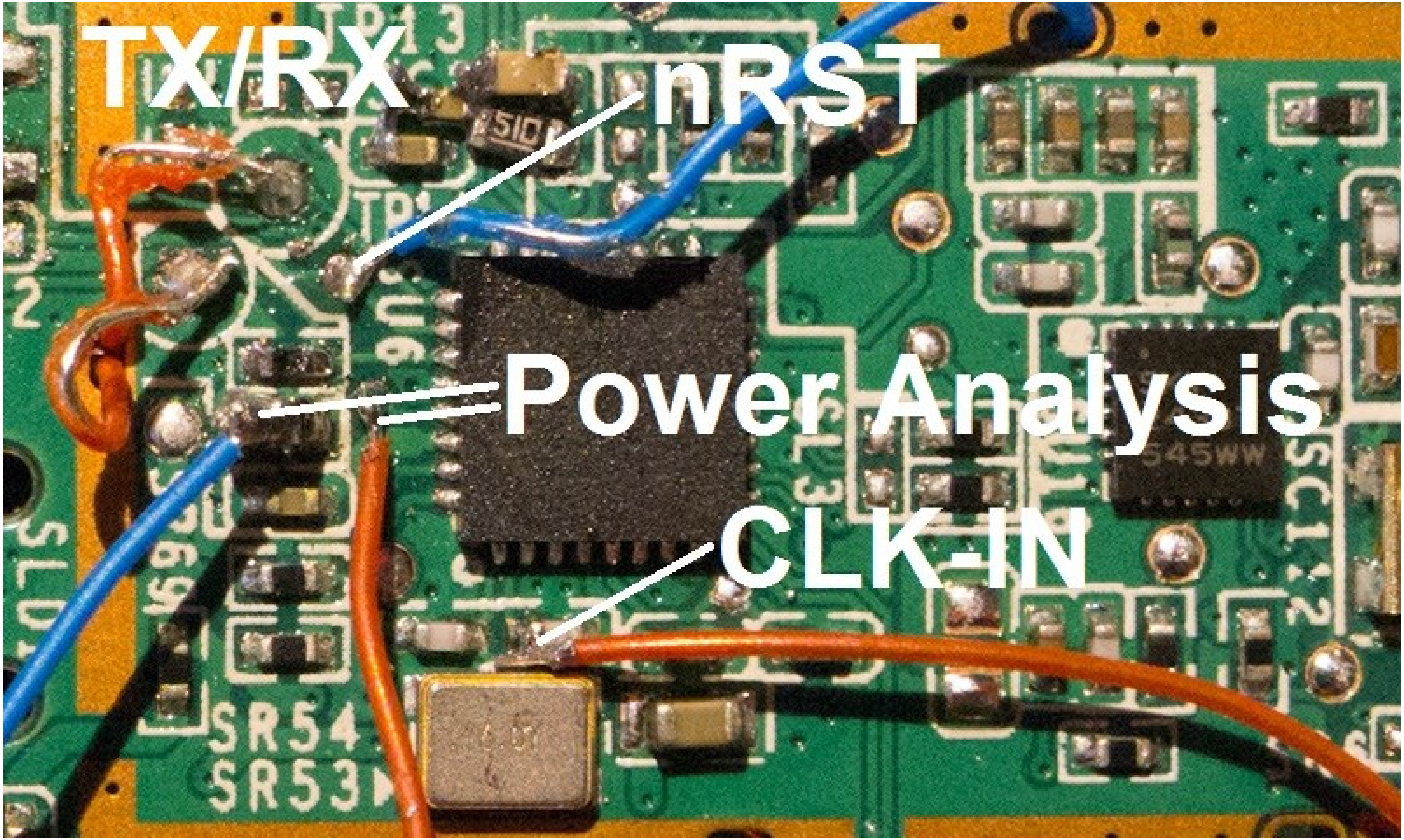
https://www.youtube.com/watch?v=hi2D2MnwiGM

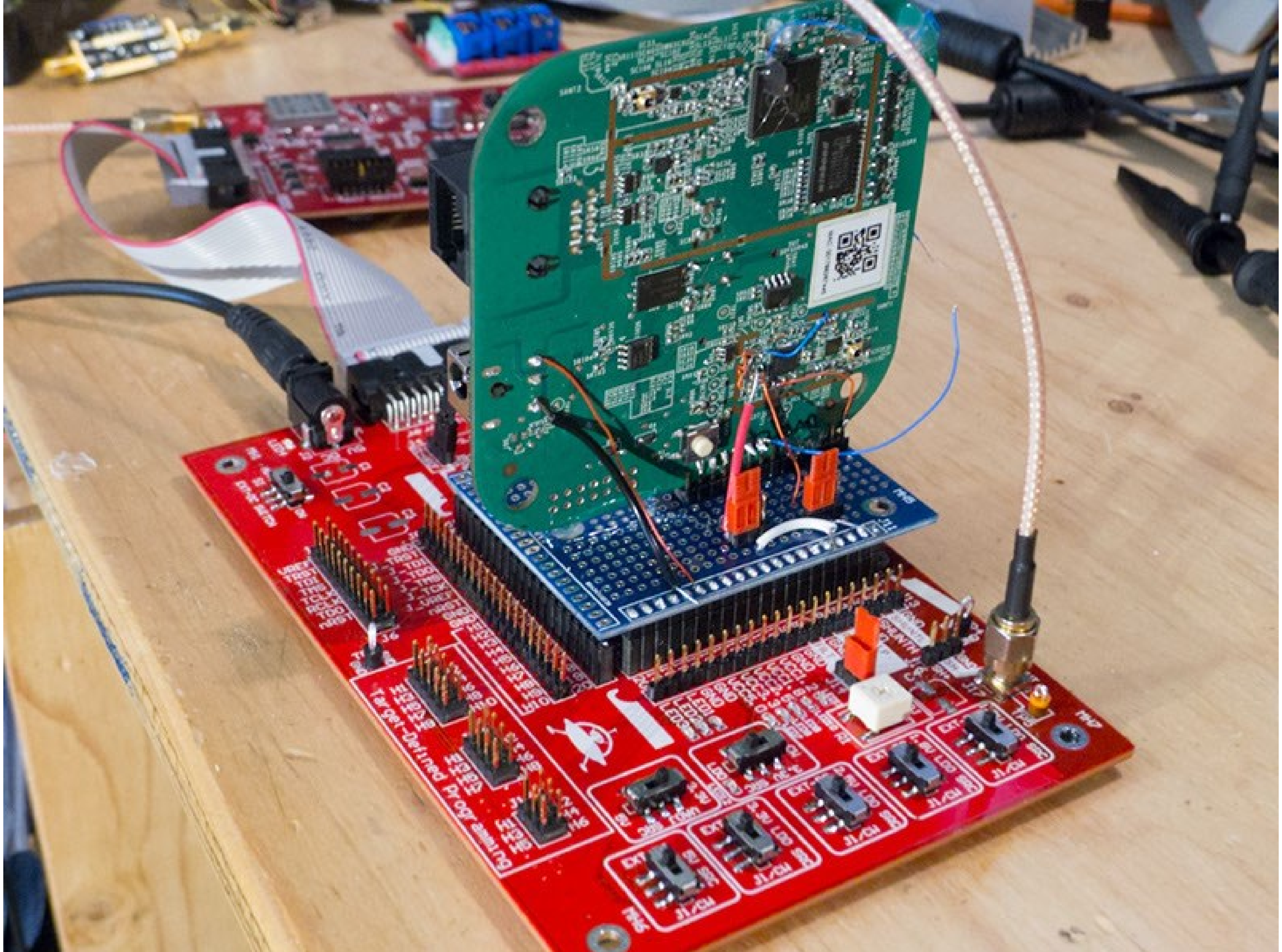
http://colinoflynn.com/?p=706

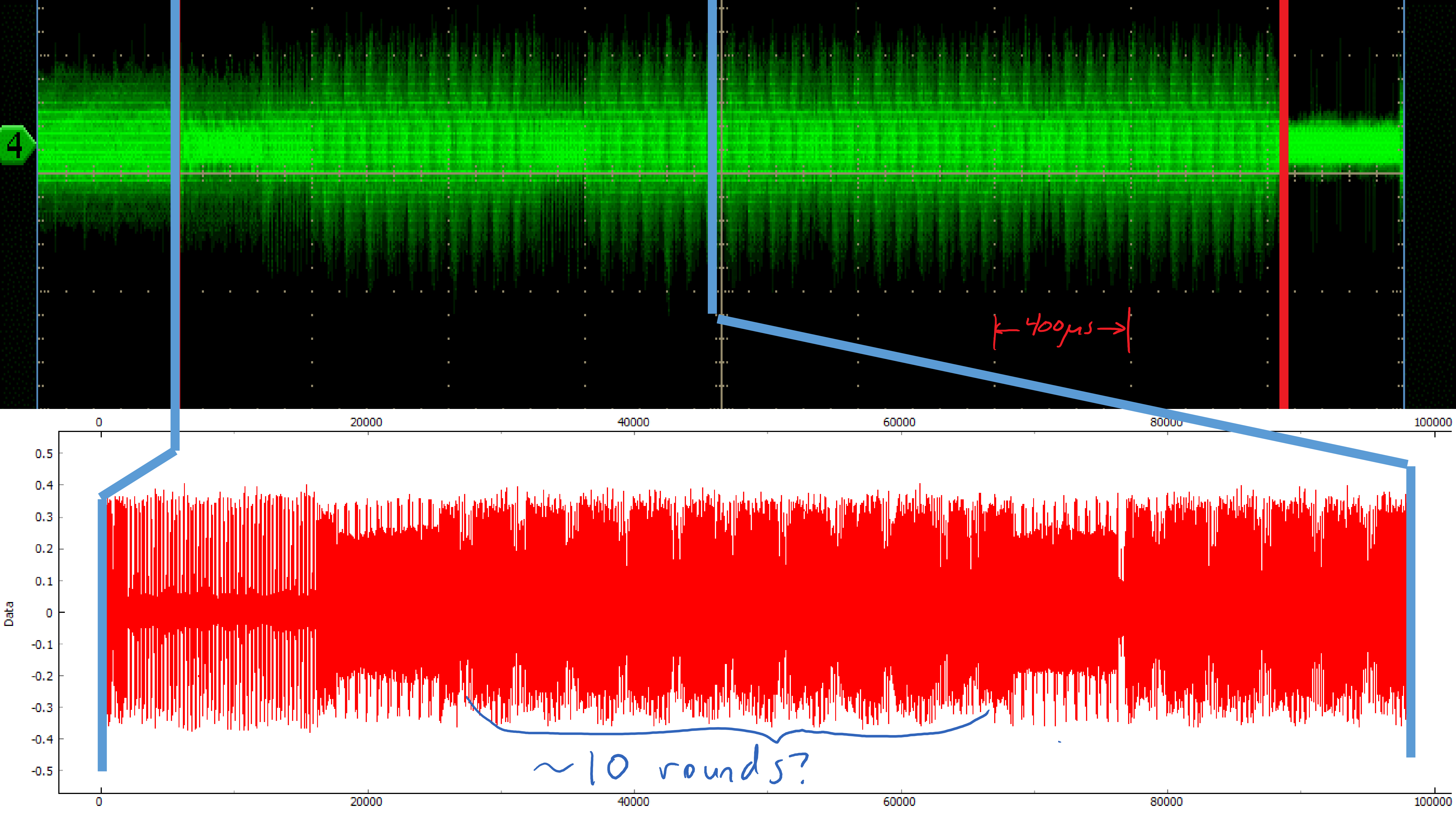
- Master binary seems to “do it all” (webserver, parsing requests, etc.) at `/usr/sbin/ipbridge`

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

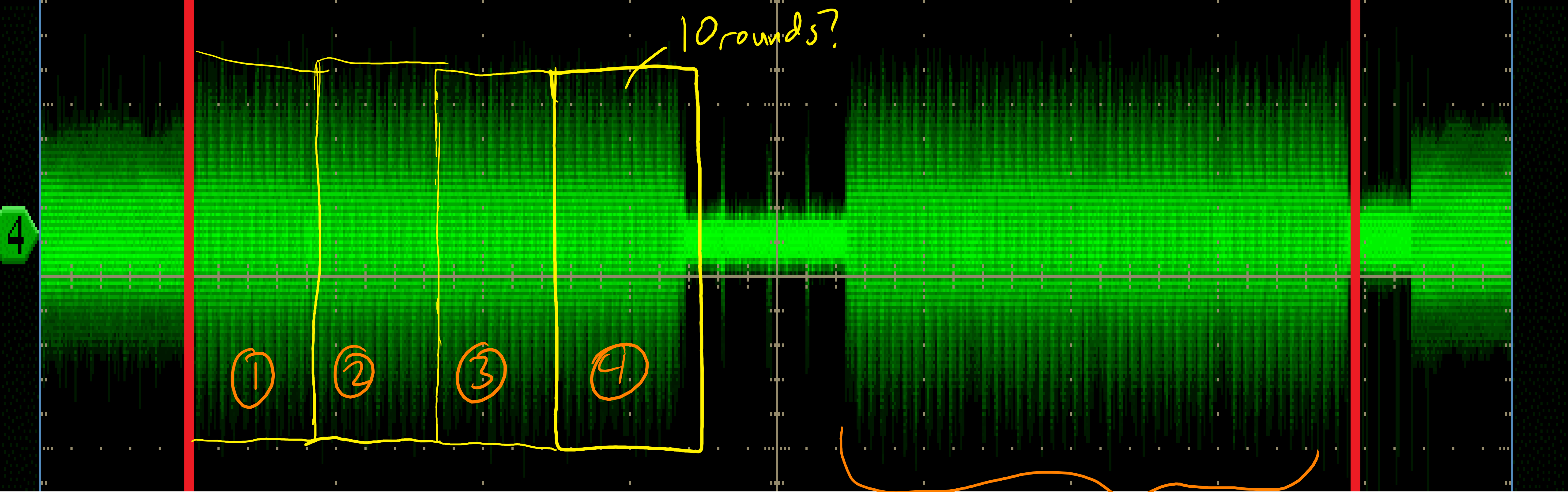
  - Two different bridges used same AES key (not really a big deal, as we already have unencrypted binaries since we have root).


TX/RX
nRST
Power Analysis
CLK-IN





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

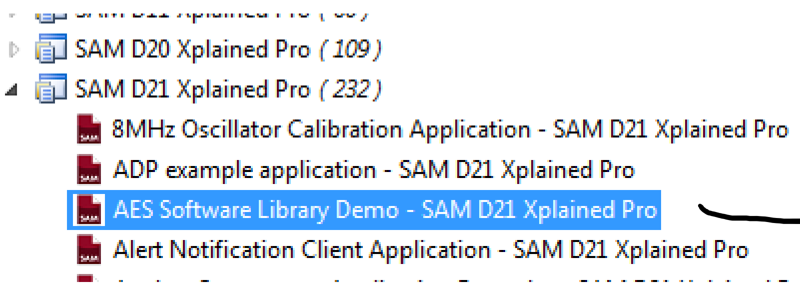


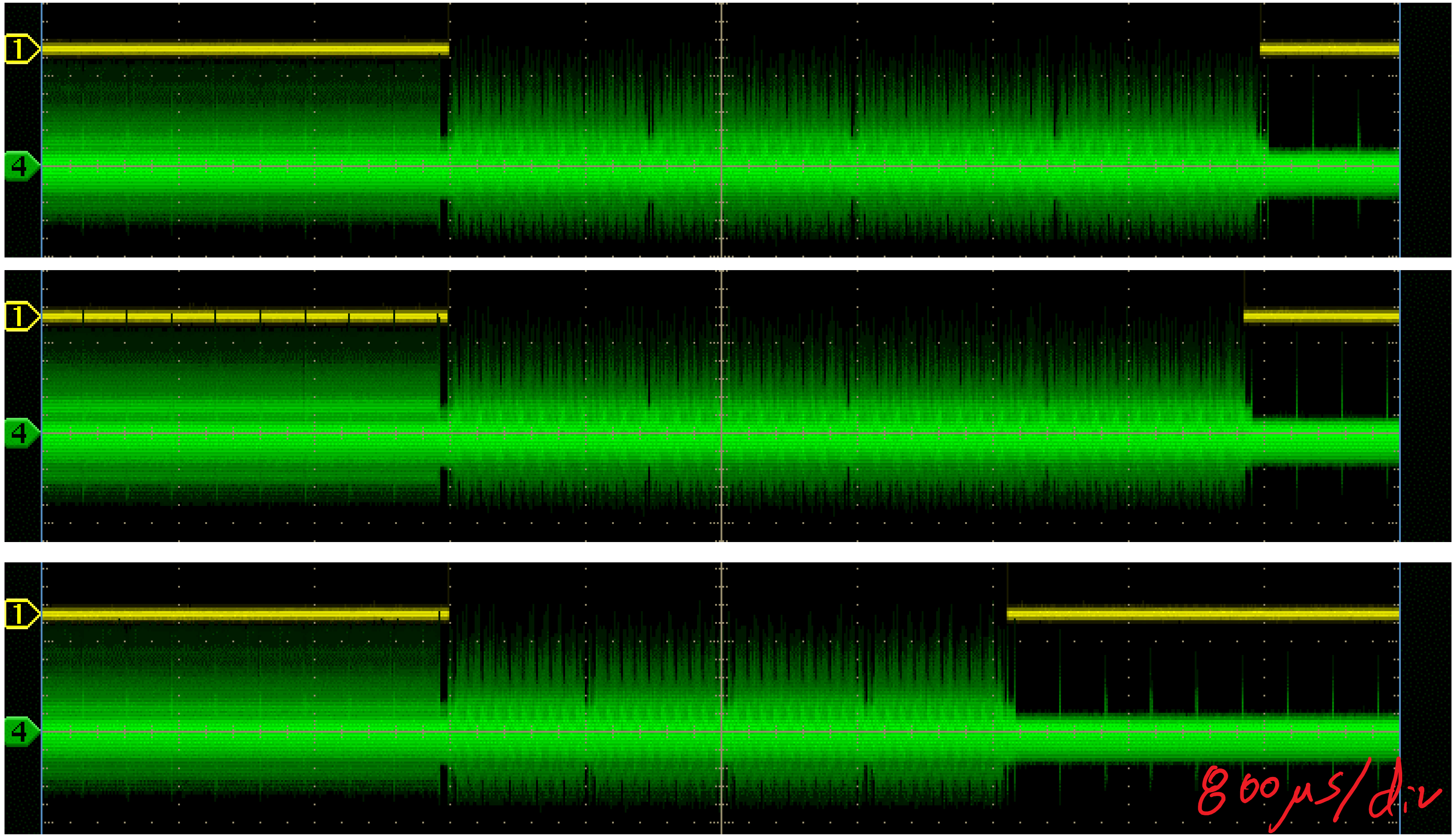
1 – 4 = AES-128?
Would explain 64 bytes.

Again!?
↳ Maybe signature?

800 μS/div

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





ECB

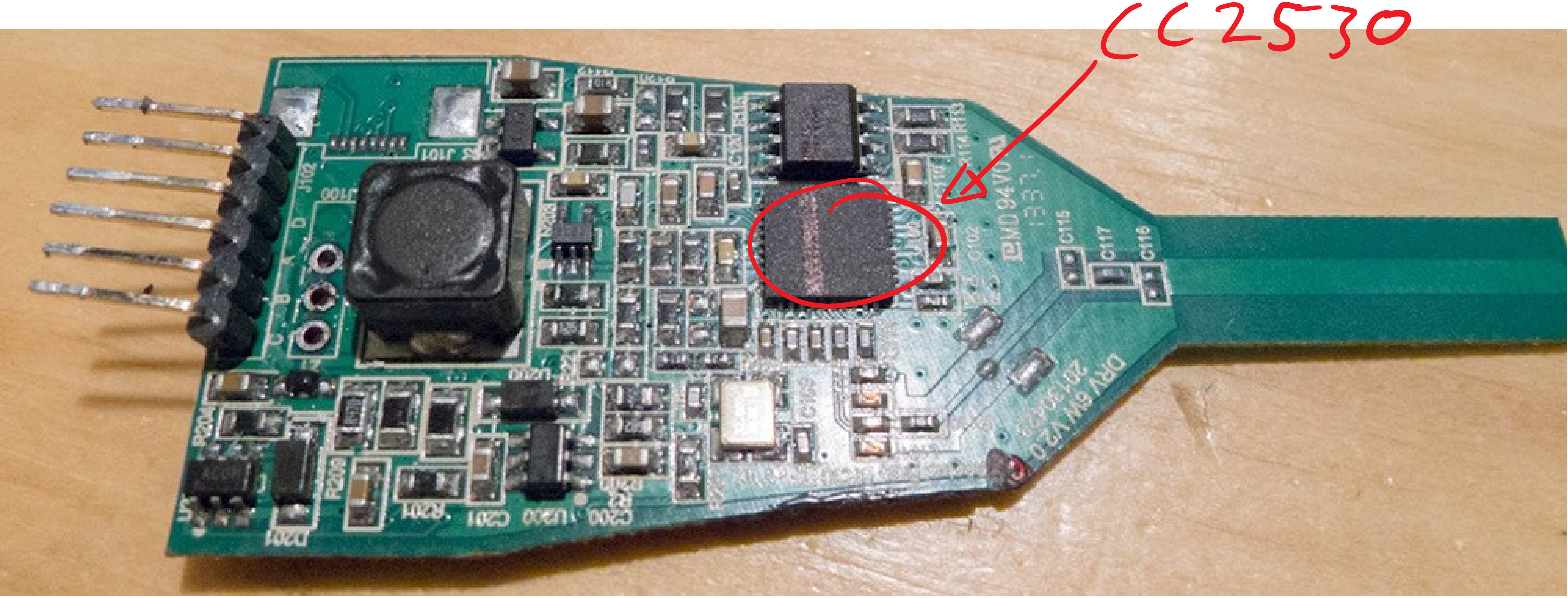
CBC

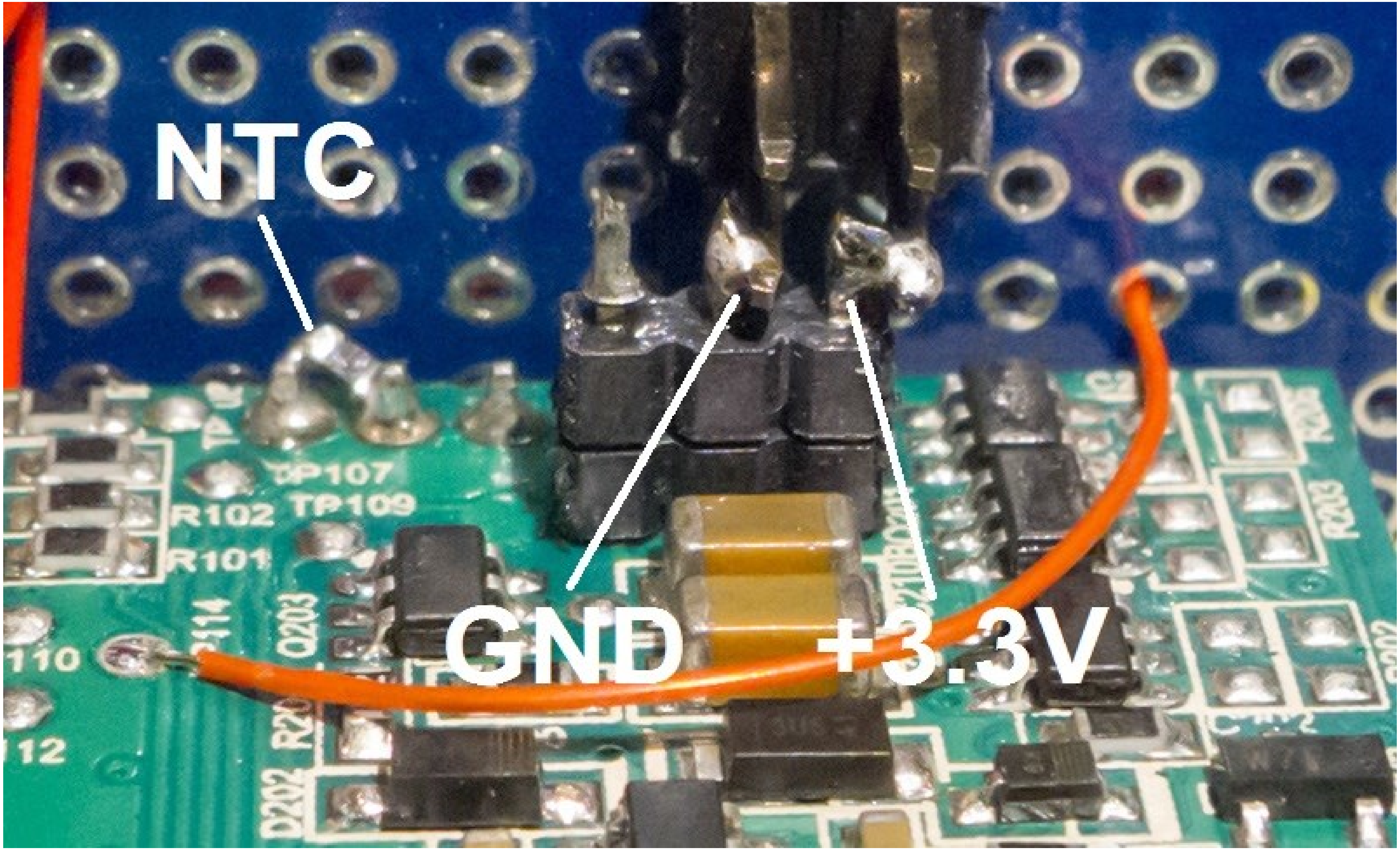
CTR

64 BYTE DECRYPTION

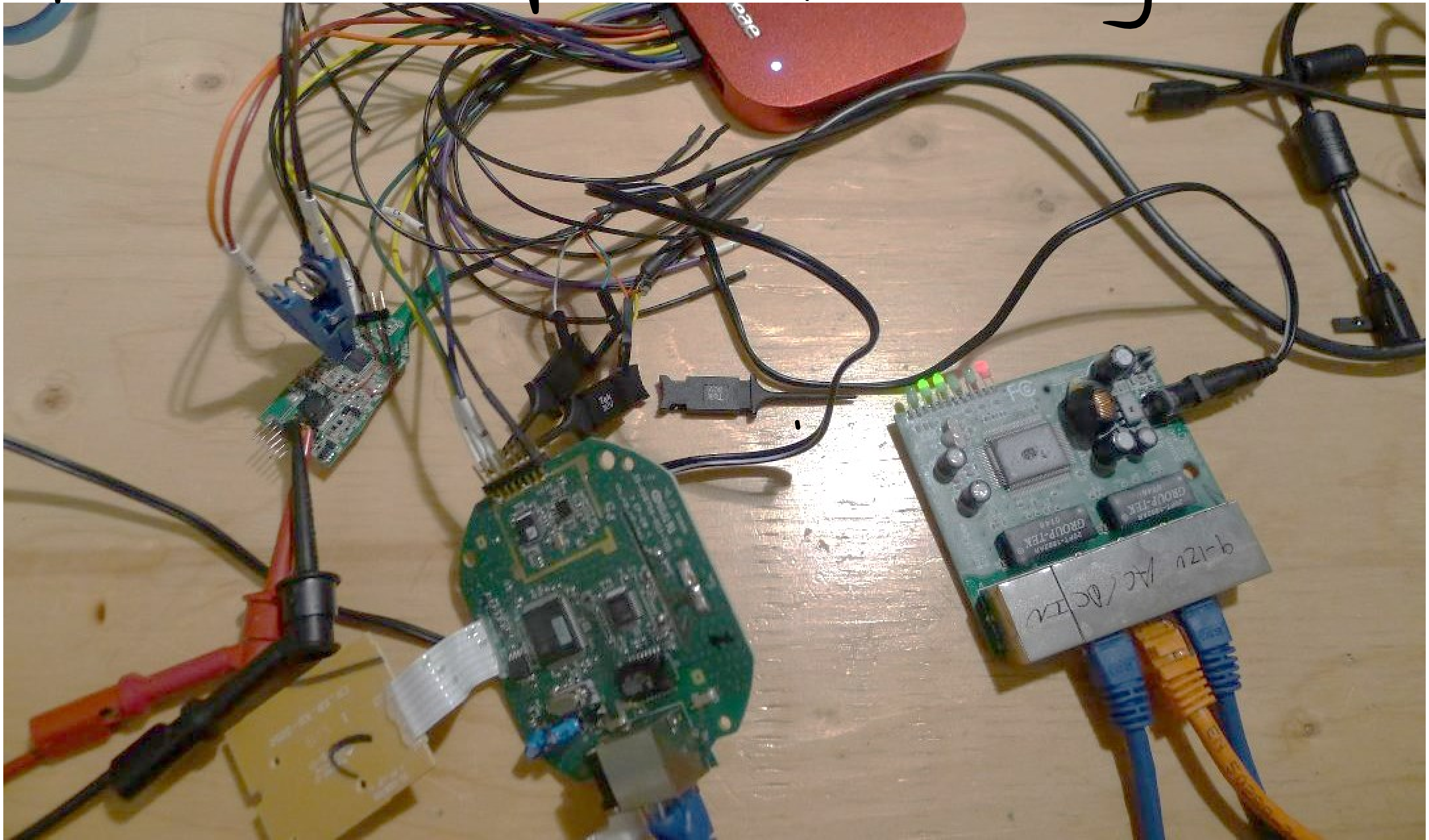

BR30
DOWN LIGHT


CC2530 Based


CC2530

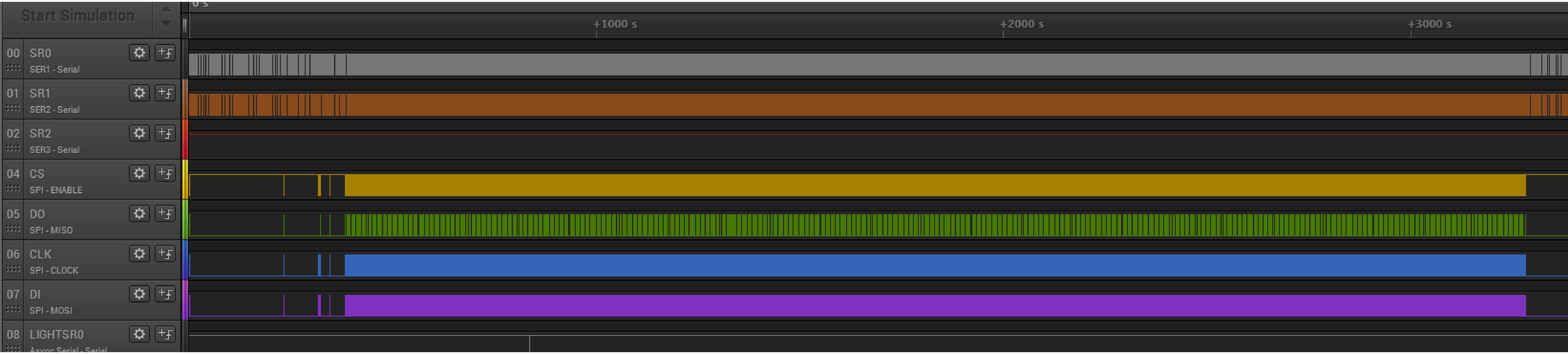

NTC
GND
+3.3V

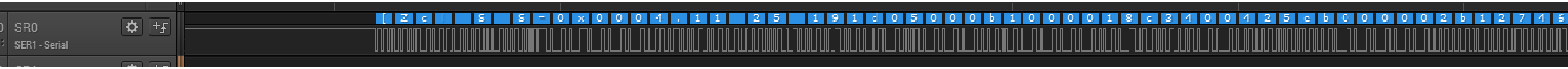
Firmware Update Hacking

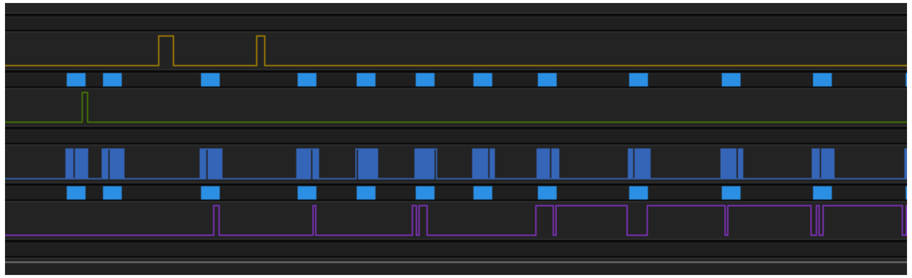


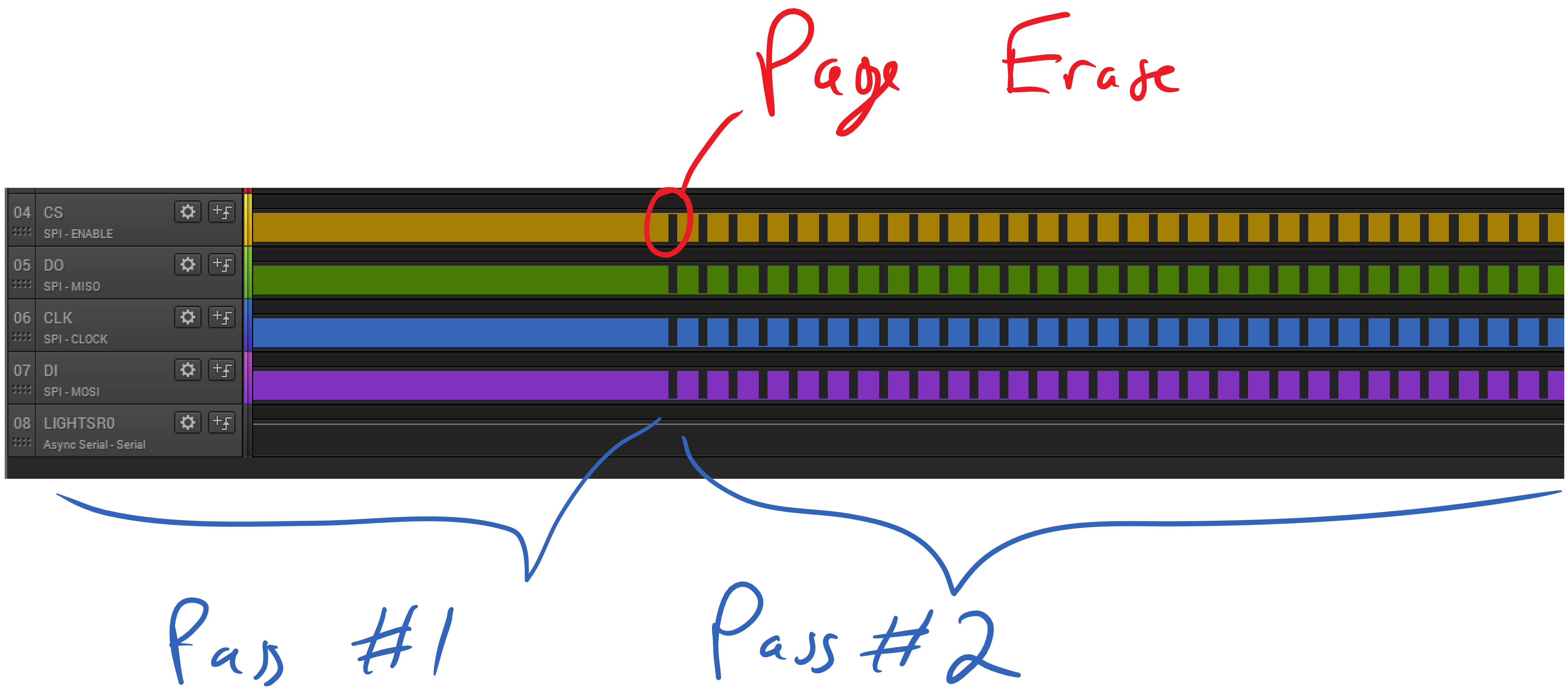
Using Salae Pro-16

↳ Capture 8MHz SPI Traffic

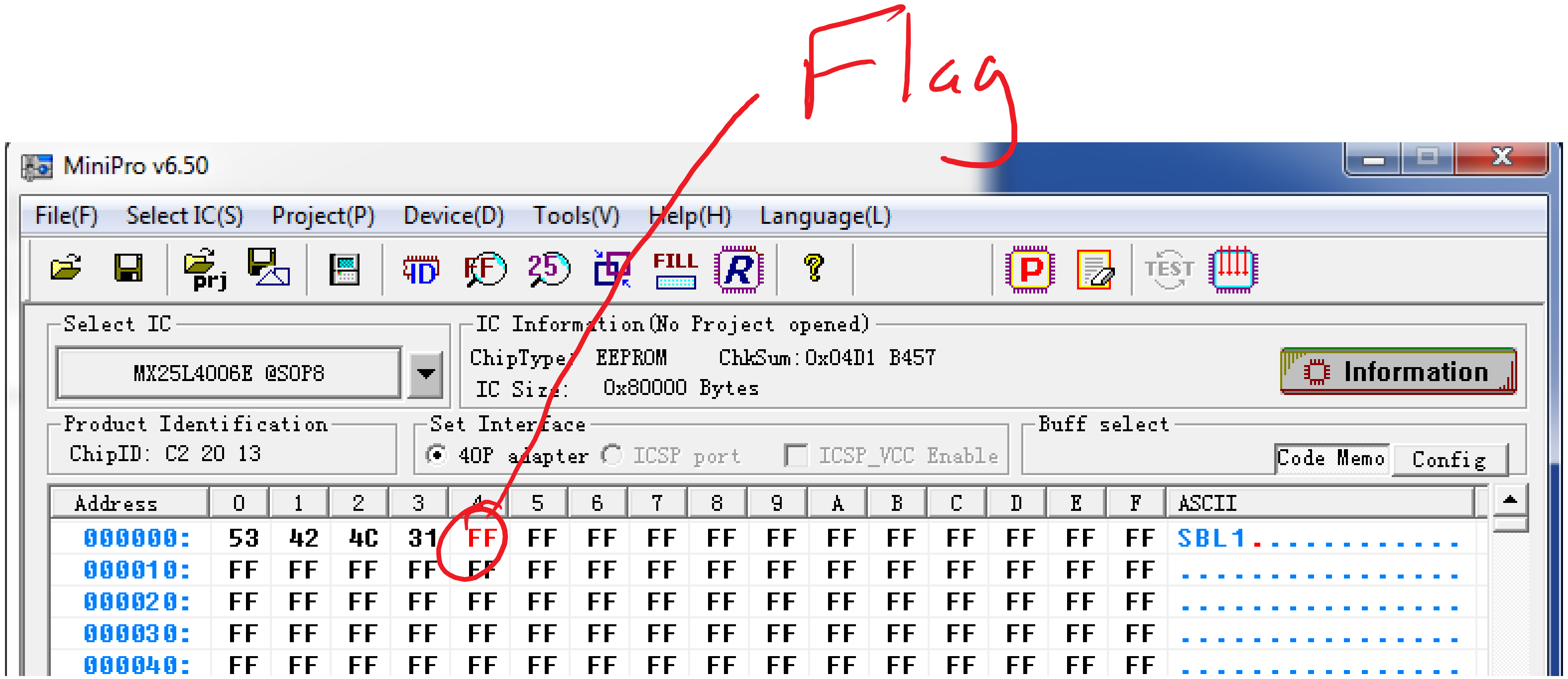




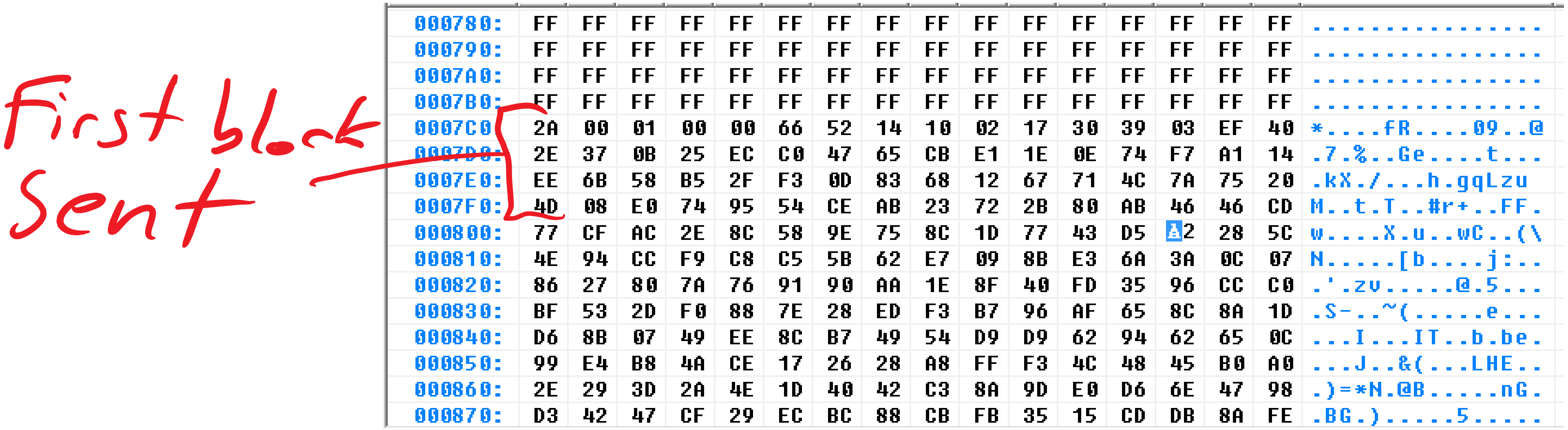




Page Erase
04 CS
SPI - ENABLE
05 DO
SPI - MISO
06 CLK
SPI - CLOCK
07 DI
SPI - MOSI
08 LIGHTSR0
Async Serial - Serial
Pass #1
Pass #2



Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
000000:	53	42	4C	31	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	SBL1............
000010:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
000020:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
000030:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
000040:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....

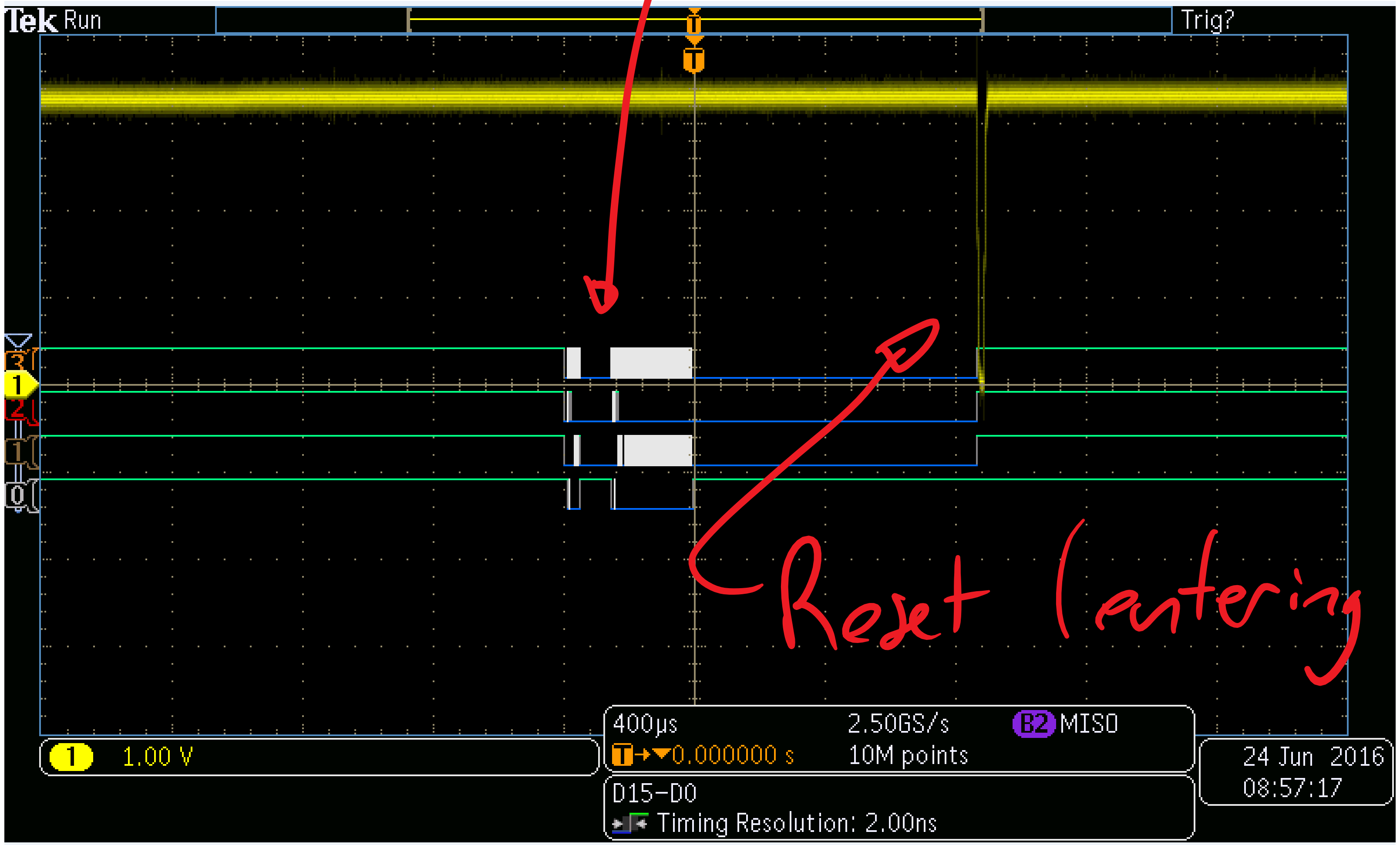


000780:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
000790:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
0007A0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
0007B0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
0007C0:	2A	00	01	00	00	66	52	14	10	02	17	30	39	03	EF	40	*....fR.....09..@
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)


00000000 A7 7B 8B 33 11 A4 C9 33 84 A2 DE 32 5C DA E4 B0
00000010 EA 67 DE CF DF 6B 06 5E EF 41 2F 9E BE 7F 66 AE
00000020 A7 FA CB BB F6 FA B0 3C 17 FB 34 F9 9B F4 90 FB
00000030 EE FF FD 77 8F F3 7B 76 DF 9E 79 63 84 EB FA B3
00000040 F9 6D C5 F6 F0 5B EF 00 00 FF FF 00 7D 01 B9 EF
00000050 66 00 D9 E7 00 00 06 0E 00 78 03 FF FA E6 00 30
00000060 F0 0E C4 1C 00 4C F8 07 E6 00 30 F0 0E C4 1C 46
00000070 F8 DE FA 9B BE 00 8C FA 00 E6 EF C2 EF A5 00 E0
00000080 07 2A 00 01 00 00 66 52 14 10 02 17 30 39 03 EF
00000090 40 4C 49 A6 E7 2D 6E 2D B8 D4 48 DD 37 B3 10 1F
000000A0 9C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000B0 00 00 00 00 00 00 00 00 00 2A 00 01 00 00 66 52
000000C0 14 10 02 17 30 39 03 EF 40 2E 37 0B 25 EC C0 47
000000D0 65 CB E1 1E 0E 74 F7 A1 14 EE 6B 58 B5 2F F3 0D
000000E0 3 68 12 67 71 4C 7A 75 20 4D 08 E0 74 95 54 CE
000000F0 AB 23 72 2B 80 AB 46 46 CD F4 3E 66 00 53 42 4C
00000100 81 FF 00 00 00 00 00 E8 FB 01 02 03 01 09 5E B5
00000110 A3 00 FE F8 BF 6B FB 79 F5 1F 89 8F B5 F9 E2 7E
00000120 F3 CD 44 5D 3B B9 FB 1B FA 60 DD FF DF 1F 72 FE
00000130 17 99 3A 97 DE FF 4E 33 78 7D 06 DE C7 71 AD DF
00000140 CE BE B2 EB 4C D4 CE 05 0E BD BB 6C BD 23 D4 BF
00000150 1D 75 BE F5 E6 FF 95 BB DD C3 BD 11 DF F0 DF 88
00000160 8E 89 7F FD 51 11 E7 DA 7E 7C AC 4B D4 AF 7D 5C
00000170 74 CD A9 EE DF 29 C7 BB 8D B8 EF BB FE D1 BD F3
00000180 A5 F9 5C 57 A6 B4 96 FF B0 7F BD 7E D1 7F AA 4F
00000190 F6 FF FE 3D 6F FB A9 F7 F4 0F 6E 7D 66 E2 FB C5
000001A0 EB 9E FF 7F F6 B1 3D 0F BA 3C 5E 6F CA 65 4D CB
000001B0 B1 8F FD F8 FB 4F F7 A3 F9 0F ED 38 FD 55 D1 ED
000001C0 53 1B 87 9D 92 AE DF 95 F2 BB 54 1E DB 28 7B 5D
000001D0 FB EC EB AF D6 2E E1 63 BD B3 37 D3 AE 83 FB B2
000001E0 76 3D 3D CB 31 BF DA 15 67 E7 6E DF EB 7F 4D 5C
000001F0 4C 7F 28 F7 4F DE 25 91 FF 2A E5 03 7E ED D7 77
00000200 81 C7 B8 DC DA B7 AF 7C 2F 6A B5 F9 31 62 14 FC
Block
First 16 bytes of block
what is this 16 bytes?
Address of SPI
0 ??
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/

Colin O'Flynn

@colinoflynn

oflynn.com

newae.com

coflynn@newae.com

Also read the W.P. for details.