black hat® USA 2023

AUGUST 9-10, 2023

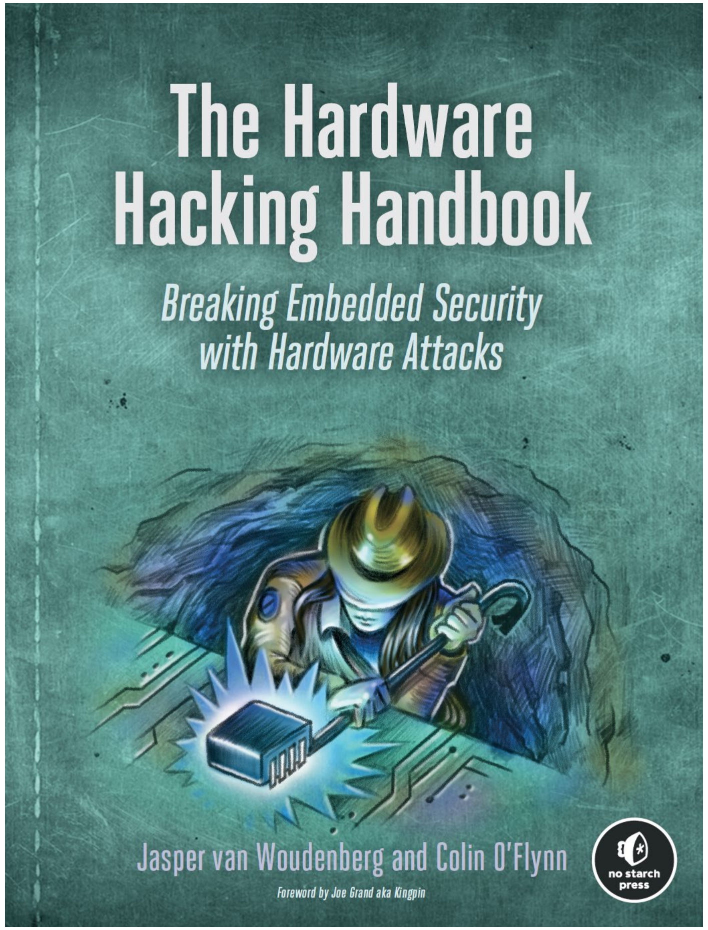
BRIEFINGS

Oven Repair The Hardware Hacking Way

Speaker: Colin O’Flynn

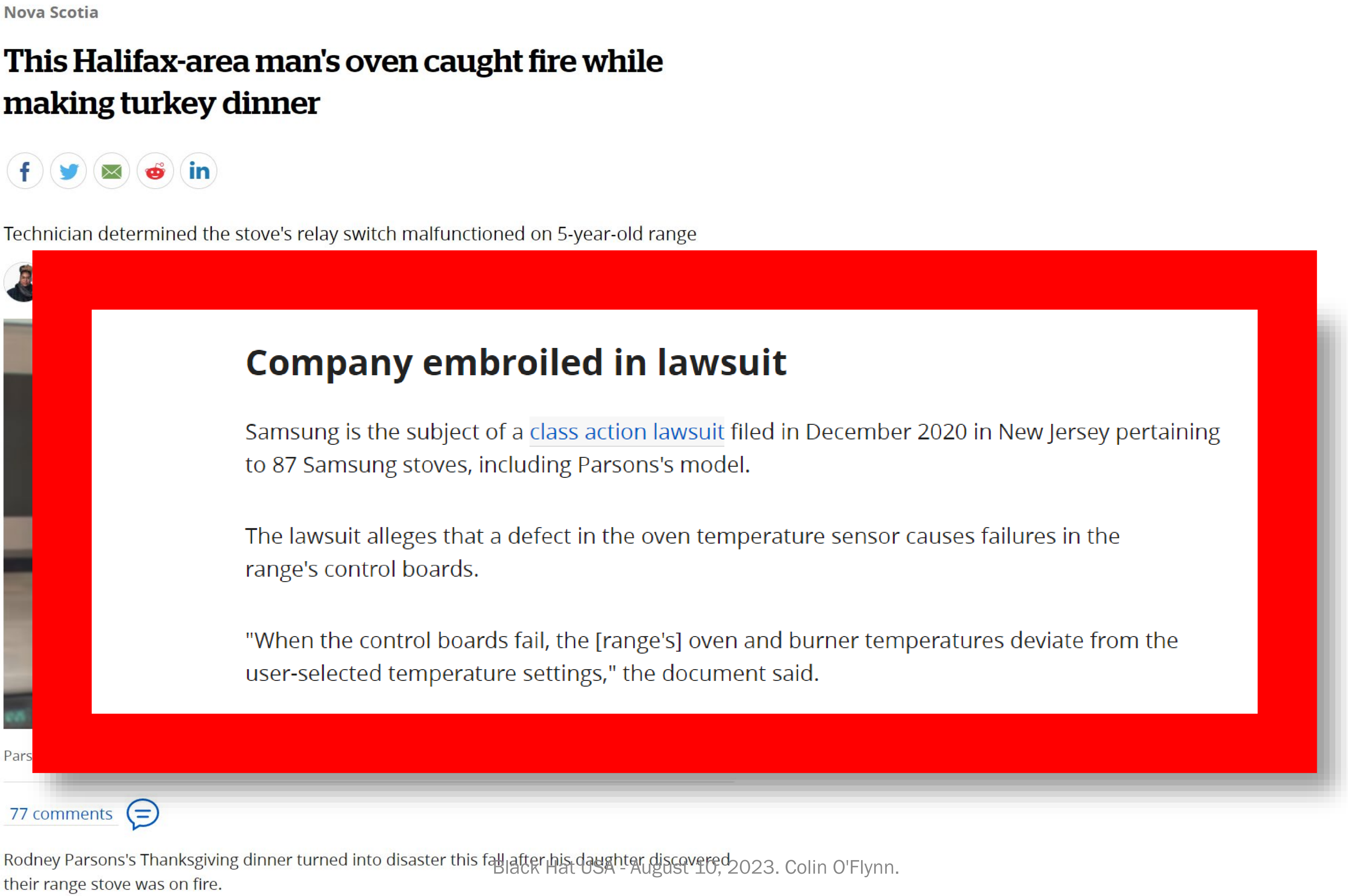
#BHUSA @BlackHatEvents

About Me


The Hardware Hacking Handbook
Breaking Embedded Security with Hardware Attacks
Jasper van Woudenberg and Colin O'Flynn
Foreword by Joe Grand aka Kingpin
no starch press

- Co-author of *Hardware Hacking Handbook*
- Started *ChipWhisperer* project & related company (NewAE Technology), now part of lowRISC CIC
- Adjunct professor at Dalhousie University

- Lives in Halifax, NS, Canada



Nova Scotia

This Halifax-area man's oven caught fire while making turkey dinner

Technician determined the stove's relay switch malfunctioned on 5-year-old range

Pars

Company embroiled in lawsuit

Samsung is the subject of a class action lawsuit filed in December 2020 in New Jersey pertaining to 87 Samsung stoves, including Parsons's model.

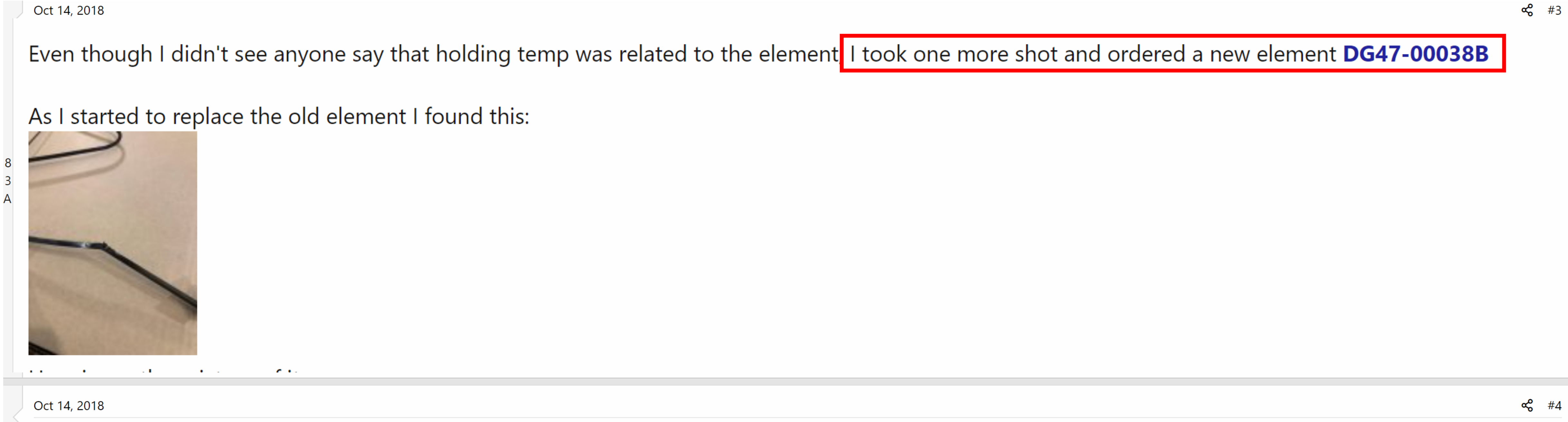
The lawsuit alleges that a defect in the oven temperature sensor causes failures in the range's control boards.

"When the control boards fail, the [range's] oven and burner temperatures deviate from the user-selected temperature settings," the document said.

77 comments

Rodney Parsons's Thanksgiving dinner turned into disaster this fall after his daughter discovered their range stove was on fire.

Wasted $$, Wasted Resources



Oct 14, 2018

#3

Even though I didn't see anyone say that holding temp was related to the element I took one more shot and ordered a new element **DG47-00038B**

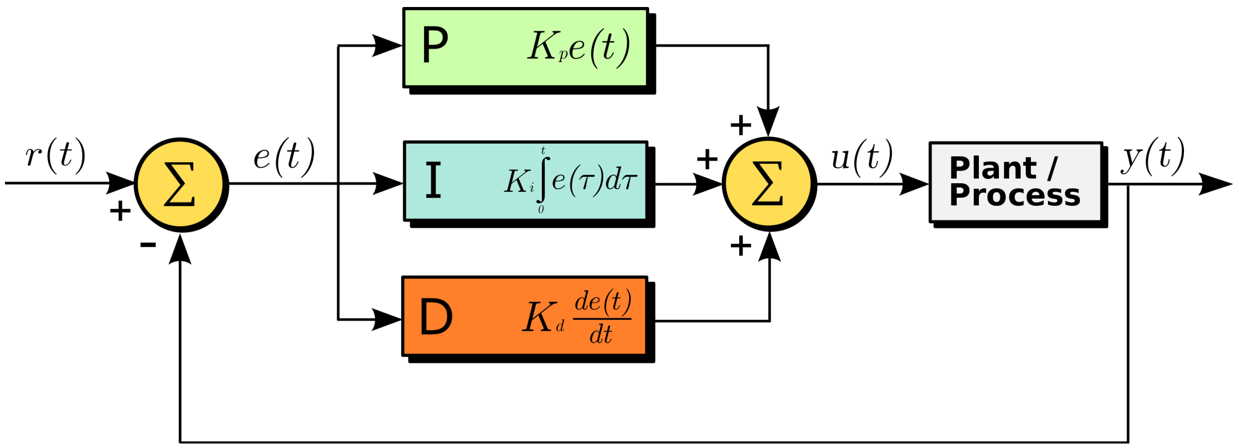
As I started to replace the old element I found this:

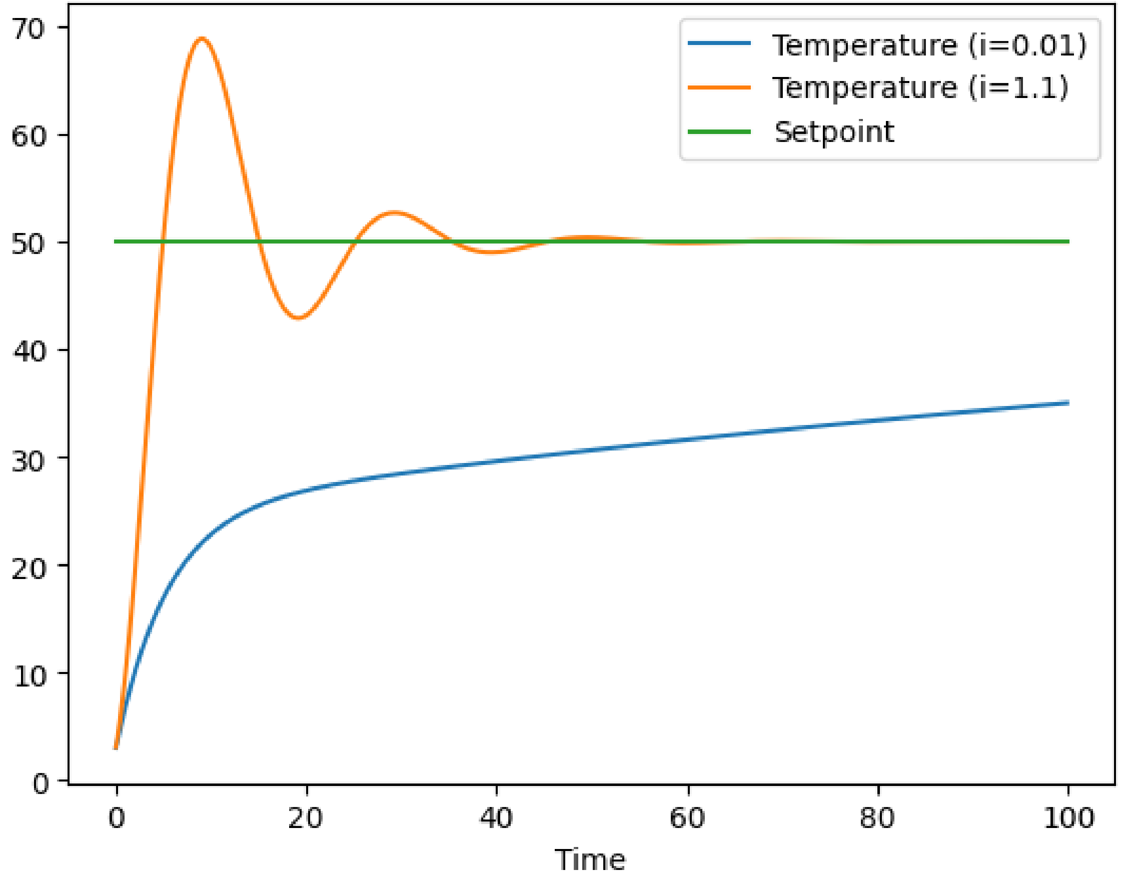
Oct 14, 2018

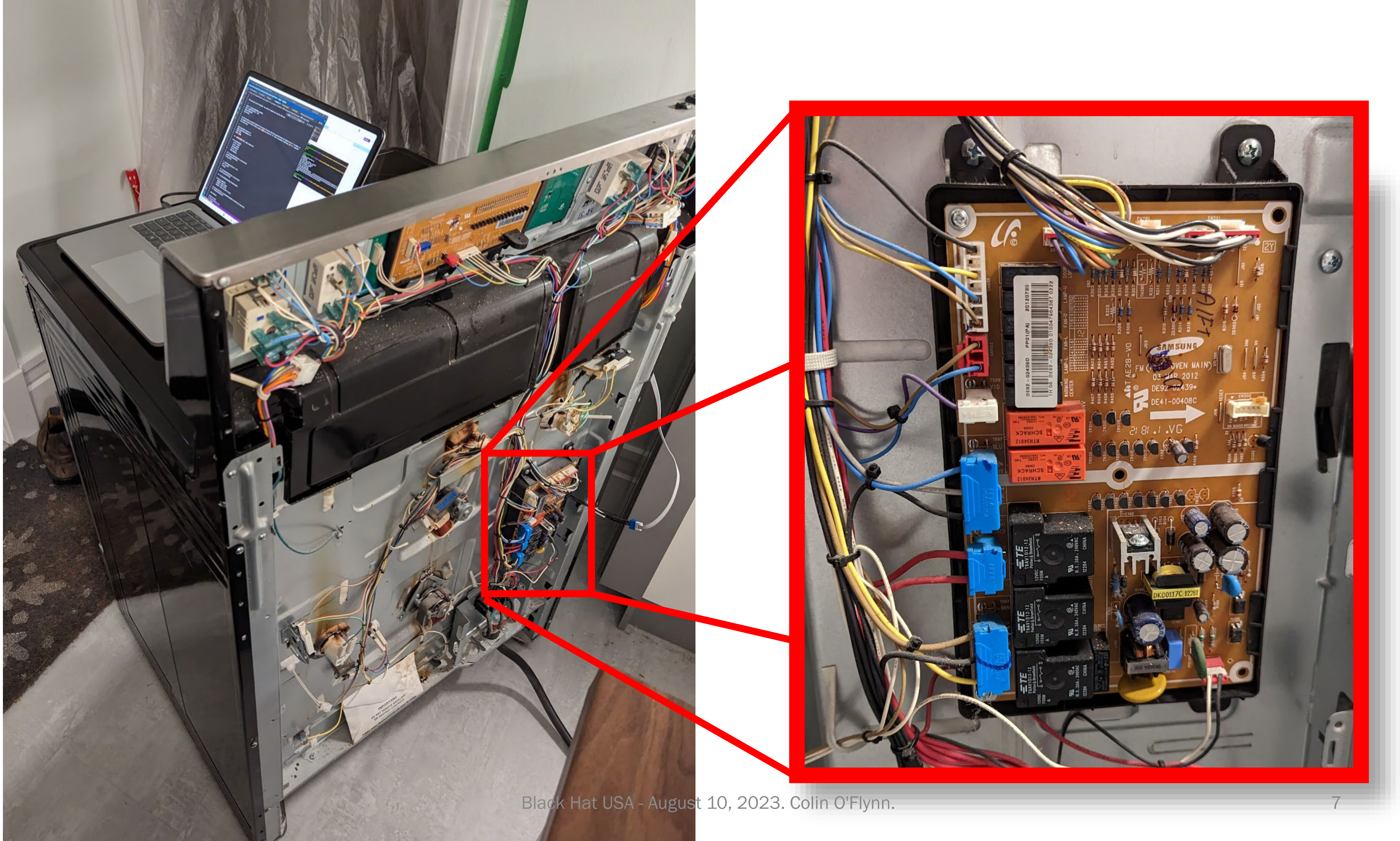
#4

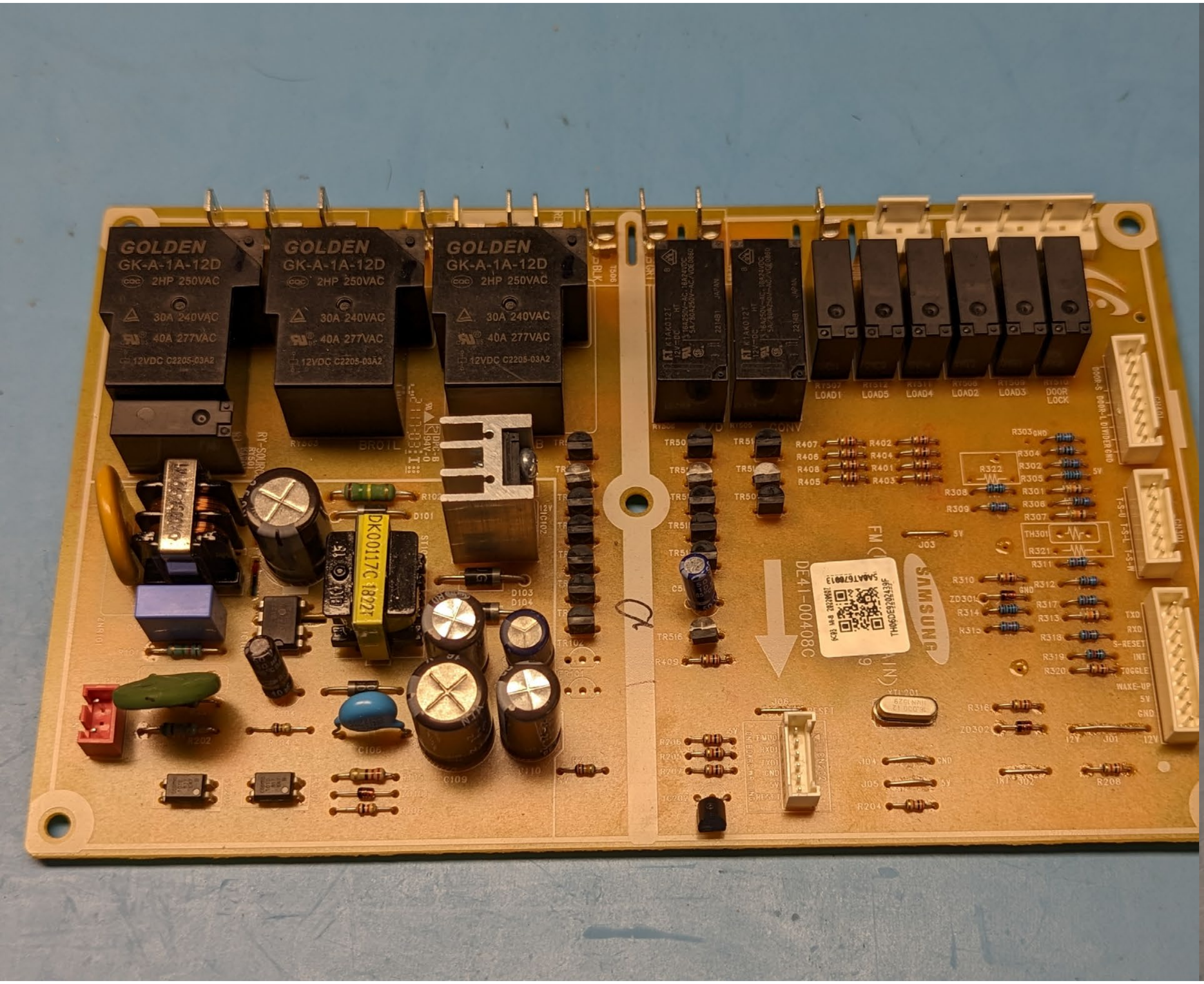
https://www.applianceblog.com/mainforums/threads/samsung-fer300sx-will-not-maintain-temperature.68145/

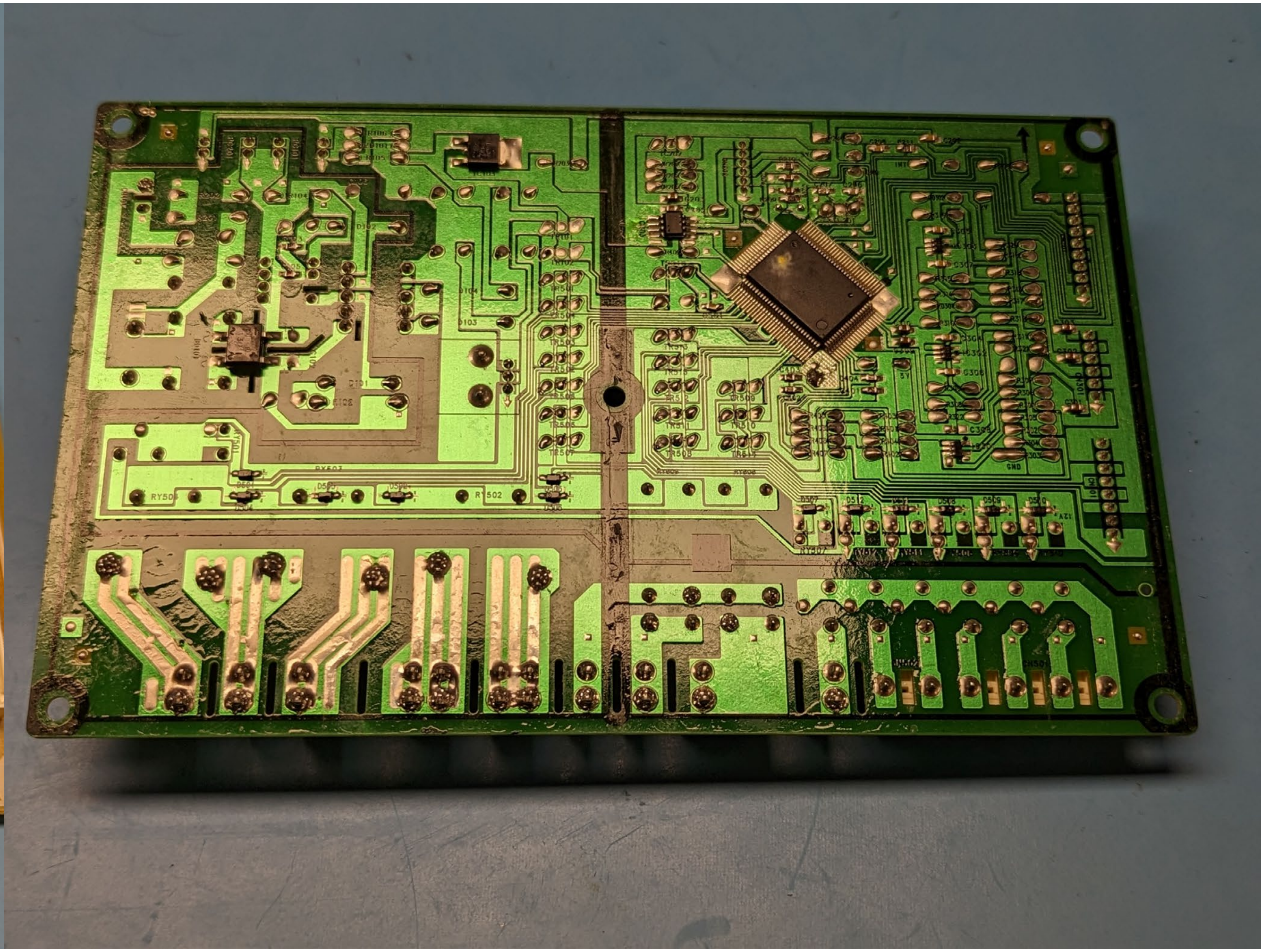
PID Controller?





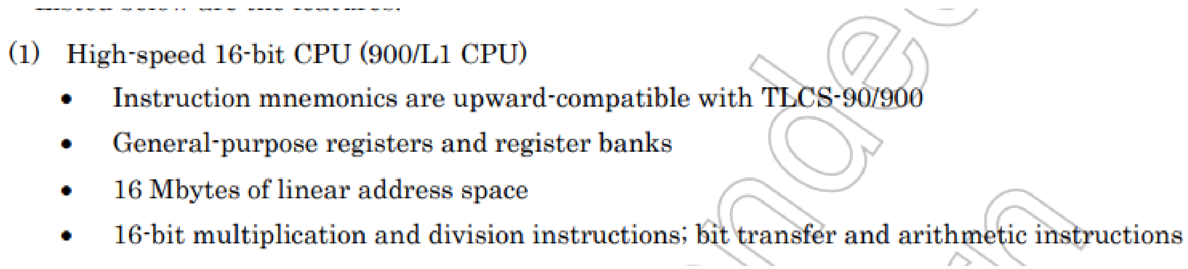

SAMSUNG
OVEN MAIN
03 MAR 2012
DE92-02439*
DE41-00408C
SCHRACK
TE
DK00117C 1228I


GOLDEN
GK-A-1A-12D
2HP 250VAC
30A 240VAC
40A 277VAC
12VDC C2205-03A2
LOAD1
LOAD5
LOAD4
LOAD2
LOAD3
DOOR LOCK
SAMSUNG
DE41-00408C
DK00117C 1822



TMP91FW60

- TLCS 900/L1 CPU
- 8K RAM / 128 K flash
- Bootloader in ROM
- External xtal (no PLL)
- Obsolete…



(1) High-speed 16-bit CPU (900/L1 CPU)

- Instruction mnemonics are upward-compatible with TLCS-90/900
- General-purpose registers and register banks
- 16 Mbytes of linear address space
- 16-bit multiplication and division instructions; bit transfer and arithmetic instructions

1.3 Block Diagram

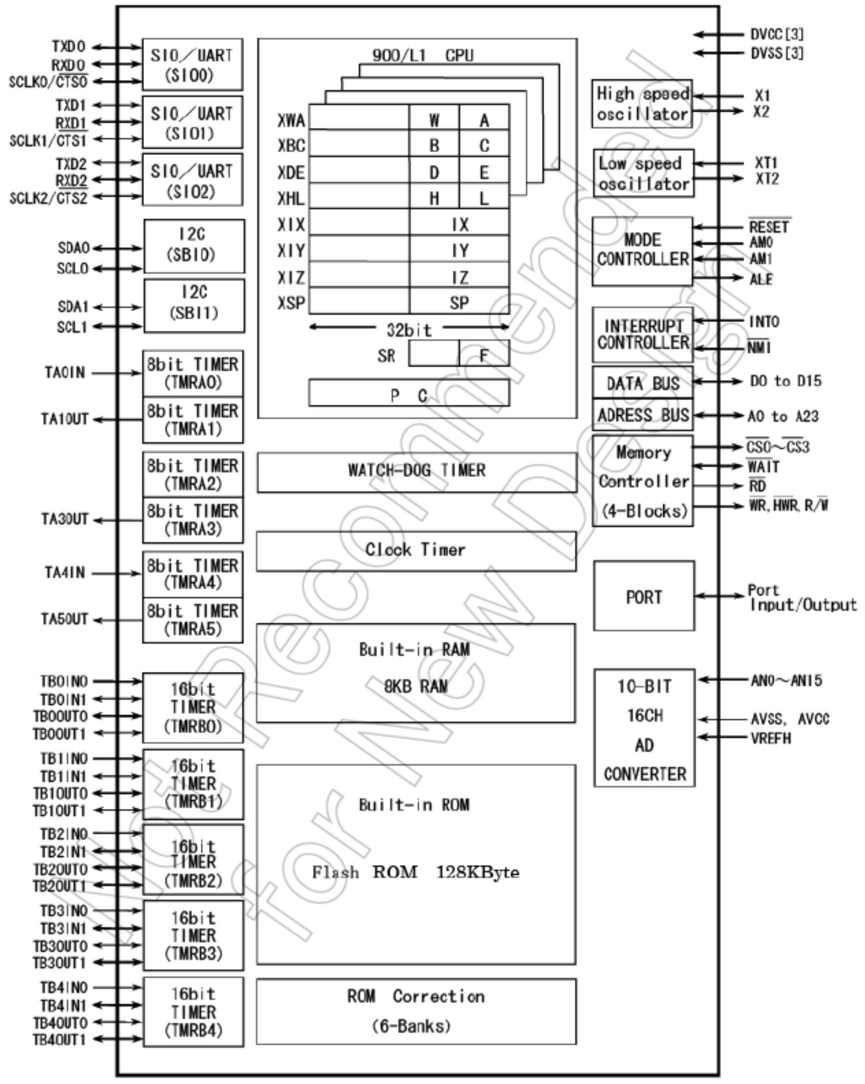
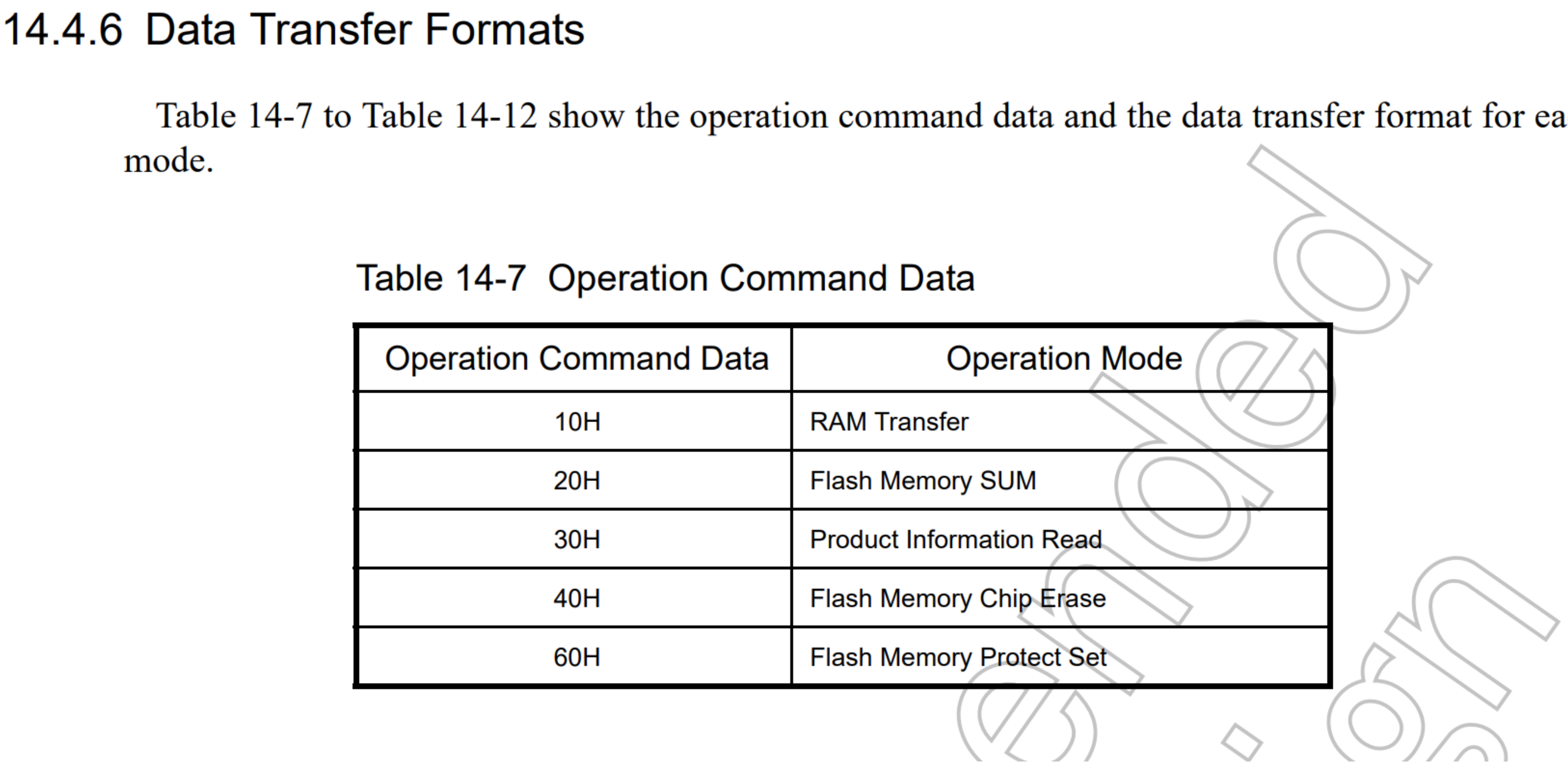


Figure 1-3 Block Diagram

Bootloader



14.4.6 Data Transfer Formats

Table 14-7 to Table 14-12 show the operation command data and the data transfer format for each operation mode.

Table 14-7 Operation Command Data

Operation Command Data	Operation Mode
10H	RAM Transfer
20H	Flash Memory SUM
30H	Product Information Read
40H	Flash Memory Chip Erase
60H	Flash Memory Protect Set

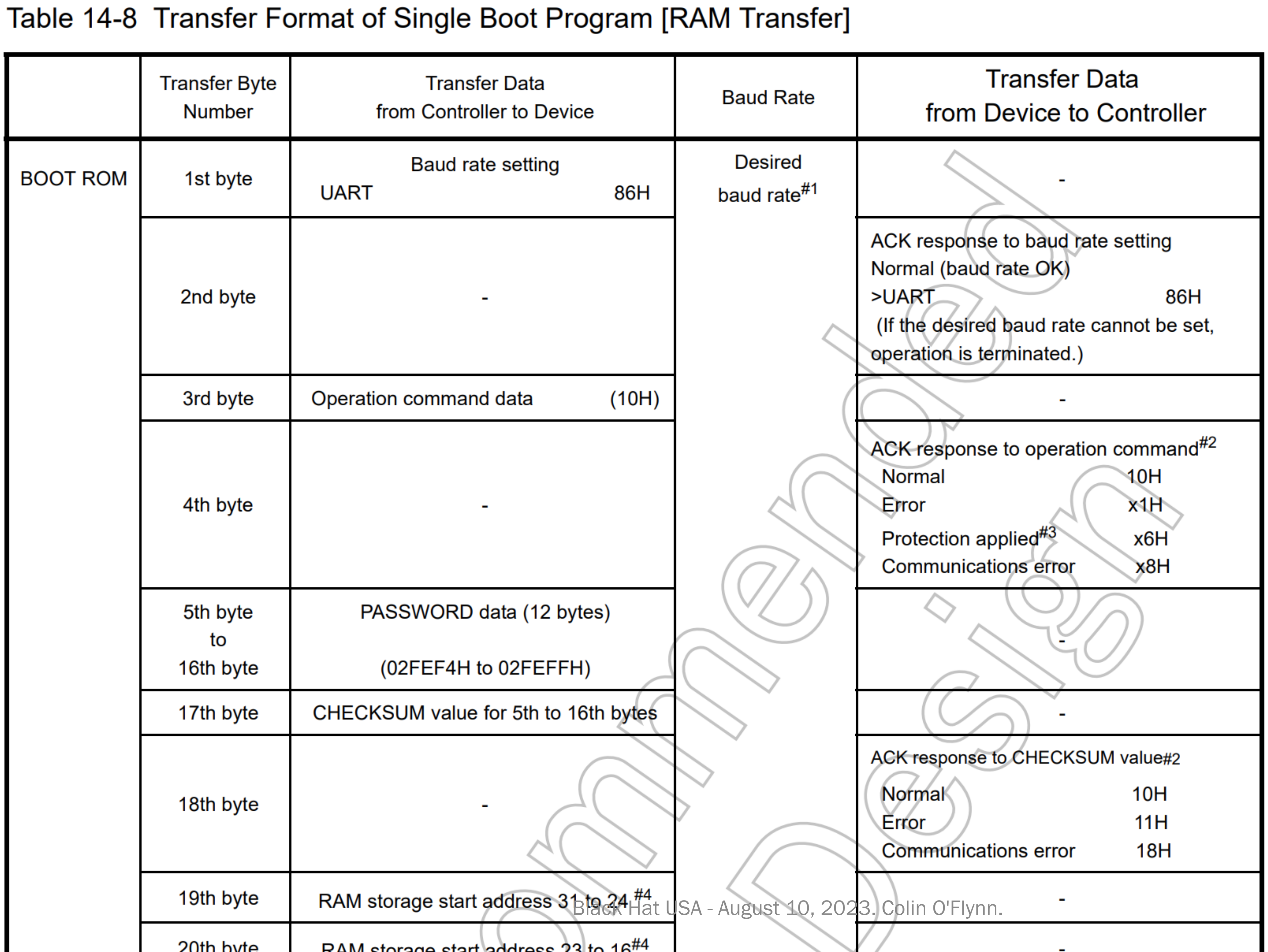


Table 14-8 Transfer Format of Single Boot Program [RAM Transfer]

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
BOOT ROM	1st byte	Baud rate setting UART 86H	Desired baud rate[#1]	-
	2nd byte	-		ACK response to baud rate setting Normal (baud rate OK) >UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (10H)		-
	4th byte	-		ACK response to operation command[#2] Normal 10H Error x1H Protection applied[#3] x6H Communications error x8H
	5th byte to 16th byte	PASSWORD data (12 bytes) (02FEF4H to 02FEFFH)		-
	17th byte	CHECKSUM value for 5th to 16th bytes		-
	18th byte	-		ACK response to CHECKSUM value#2 Normal 10H Error 11H Communications error 18H
	19th byte	RAM storage start address 31 to 24[#4]		-
	20th byte	RAM storage start address 23 to 16[#4]		-

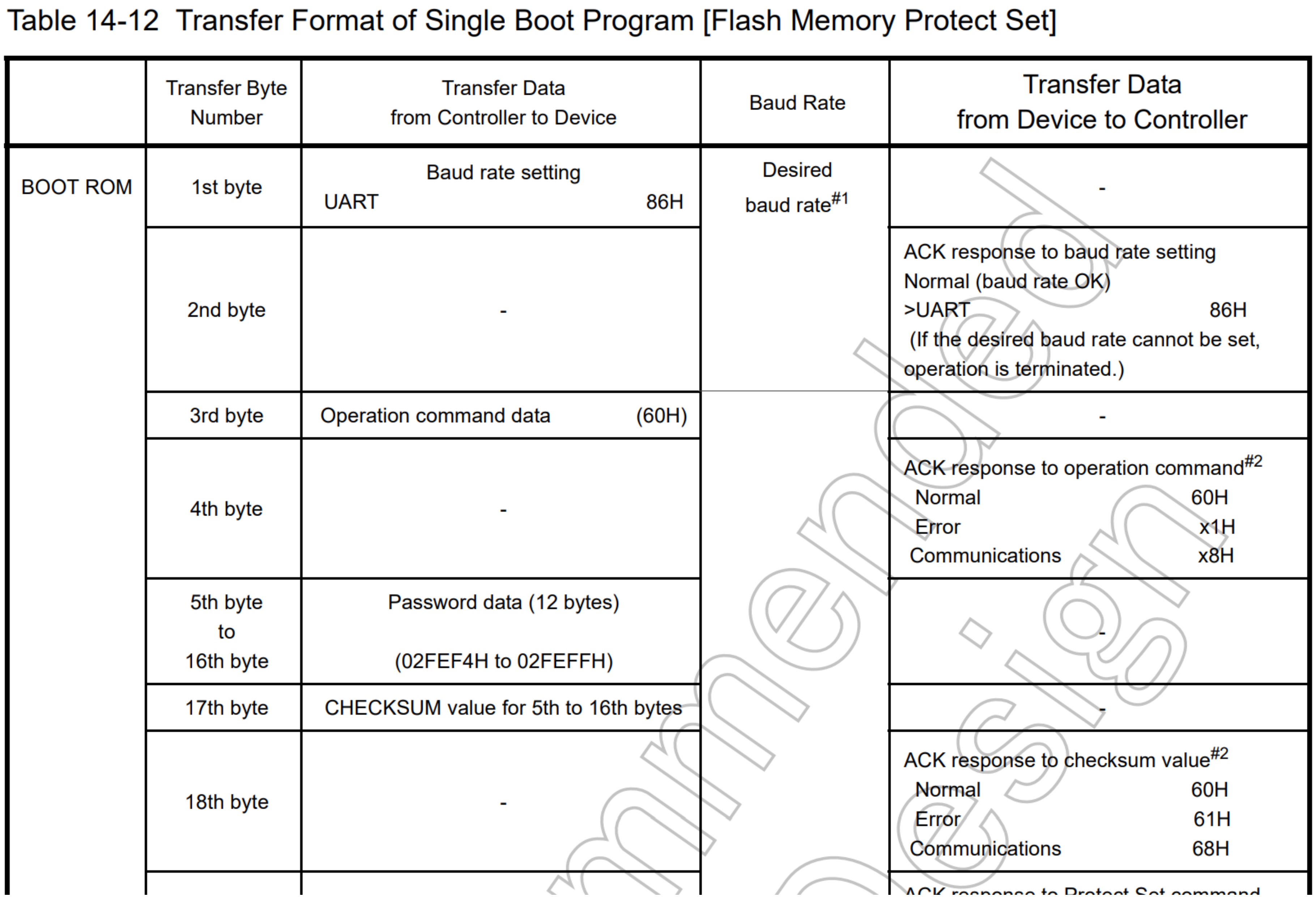


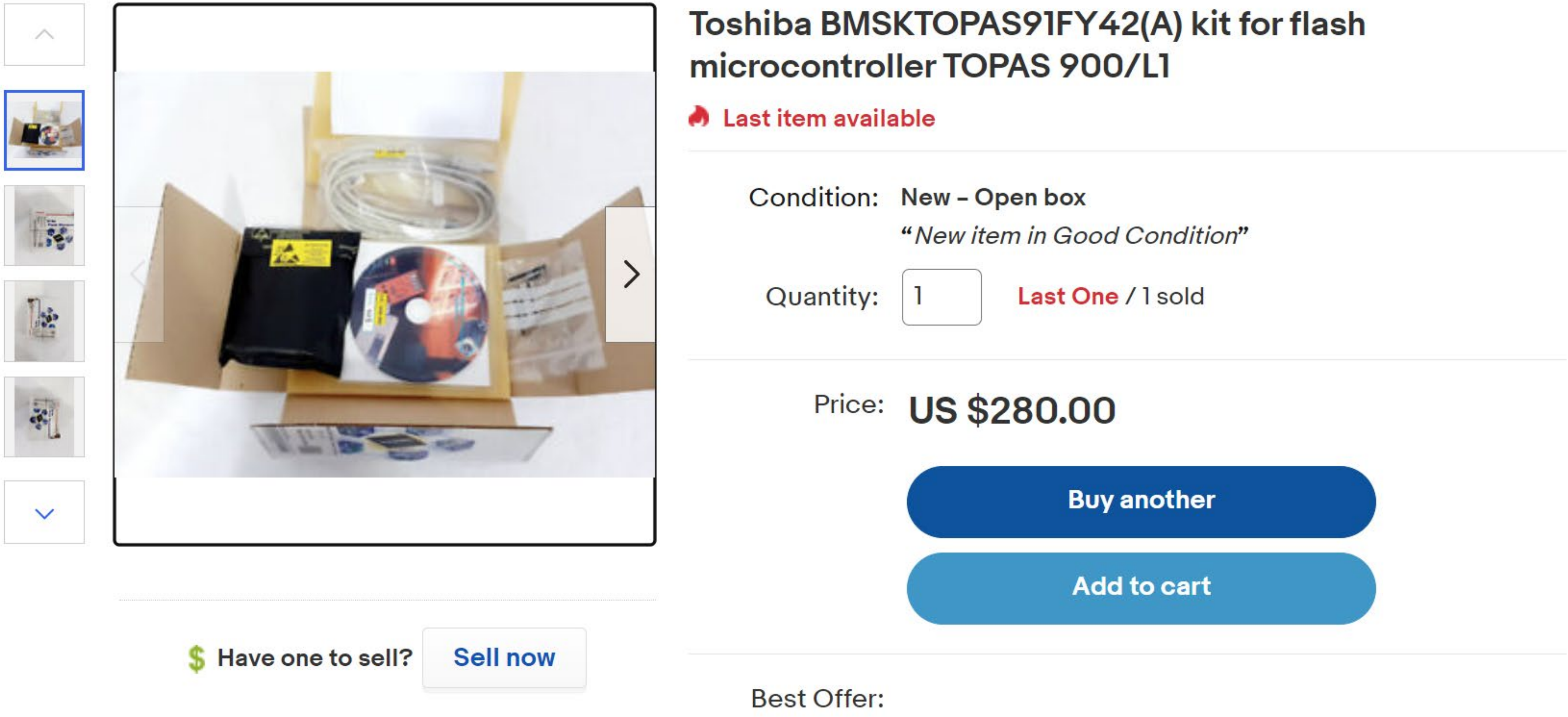
Table 14-12 Transfer Format of Single Boot Program [Flash Memory Protect Set]

	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
BOOT ROM	1st byte	Baud rate setting UART 86H	Desired baud rate#1	-
	2nd byte	-		ACK response to baud rate setting Normal (baud rate OK) >UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (60H)		-
	4th byte	-		ACK response to operation command#2 Normal 60H Error x1H Communications x8H
	5th byte to 16th byte	Password data (12 bytes) (02FEF4H to 02FEFFH)		-
	17th byte	CHECKSUM value for 5th to 16th bytes		-
	18th byte	-		ACK response to checksum value#2 Normal 60H Error 61H Communications 68H
				ACK response to Protect Set command

Important Take-Aways (for next part)

1. Bootloader has no read-back command, only RAM program. Need to build/find 2nd stage bootloader.
2. Bootloader has TWO security protections that can be enabled:
    1. “Protection Flag” → Disables second-stage capability (leaves “erase” enabled). Disables RAM functionality, so no chance to read-back flash.
    2. 12-byte Password that can be set in Flash. Password locks RAM functionality but does not disable it.
3. Bootloader has a function that only needs password (even if protection is set).

Programmer / Disassembler / Simulator?



Toshiba BMSKTOPAS91FY42(A) kit for flash microcontroller TOPAS 900/L1

Last item available

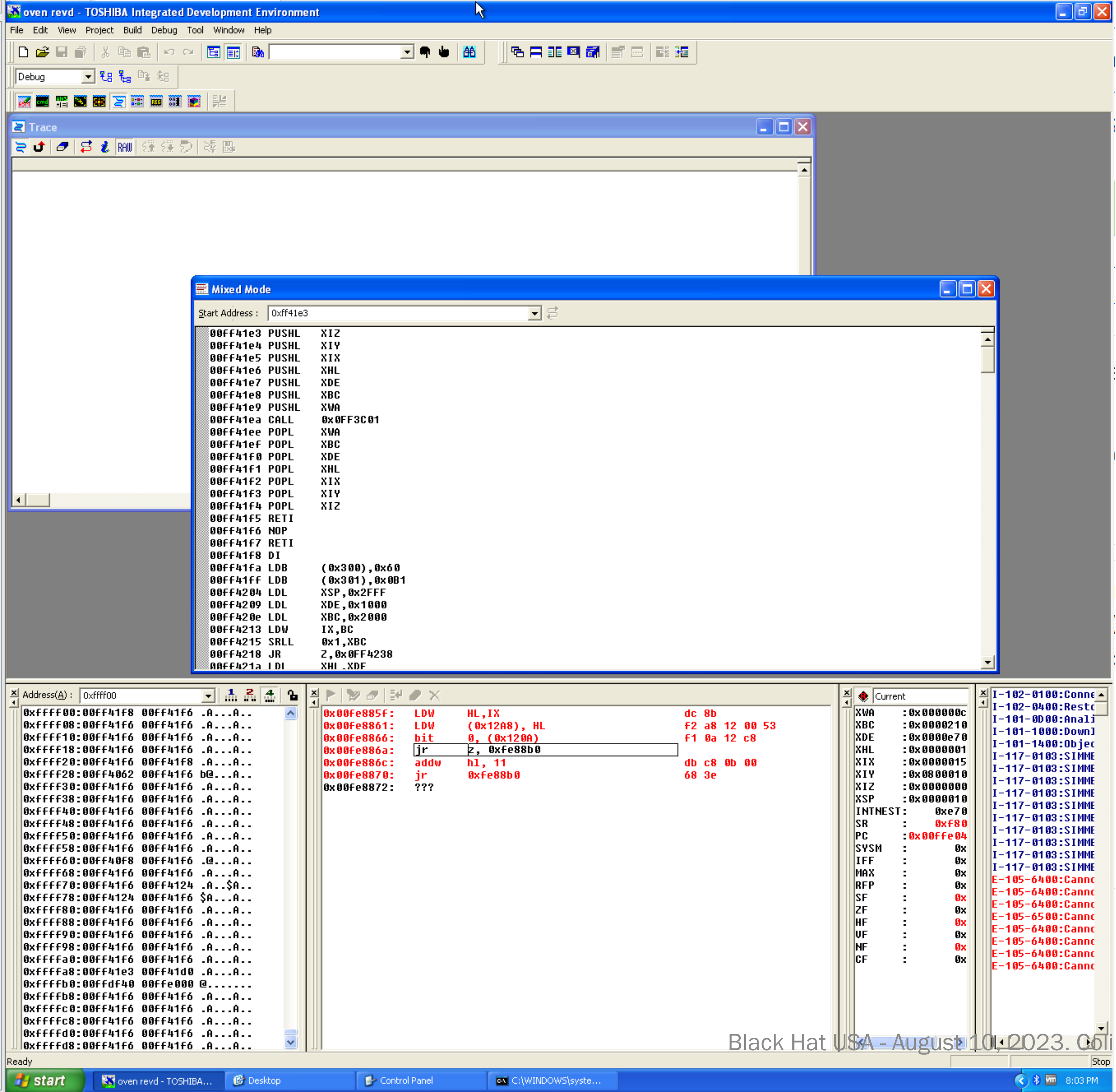
Condition: New – Open box

"*New item in Good Condition*"

Quantity: 1 Last One / 1 sold

Price: US $280.00

Best Offer:



```
00ff41e3 PUSHL   XIZ
00ff41e4 PUSHL   XIY
00ff41e5 PUSHL   XIX
00ff41e6 PUSHL   XHL
00ff41e7 PUSHL   XDE
00ff41e8 PUSHL   XBC
00ff41e9 PUSHL   XWA
00ff41ea CALL    0x0FF3C01
00ff41ee POPL    XWA
00ff41ef POPL    XBC
00ff41f0 POPL    XDE
00ff41f1 POPL    XHL
00ff41f2 POPL    XIX
00ff41f3 POPL    XIY
00ff41f4 POPL    XIZ
00ff41f5 RETI
00ff41f6 NOP
00ff41f7 RETI
00ff41f8 DI
00ff41fa LDB     (0x300),0x60
00ff41ff LDB     (0x301),0x0B1
00ff4204 LDL     XSP,0x2FFF
00ff4209 LDL     XDE,0x1000
00ff420e LDL     XBC,0x2000
00ff4213 LDW     IX,BC
00ff4215 SRLL    0x1,XBC
00ff4218 JR      Z,0x0FF4238
00ff421a LDL     XHL,XDE
```

```
0xffff00:00ff41f8 00ff41f6 .A...A..
0xffff08:00ff41f6 00ff41f6 .A...A..
0xffff10:00ff41f6 00ff41f6 .A...A..
0xffff18:00ff41f6 00ff41f6 .A...A..
0xffff20:00ff41f6 00ff41f8 .A...A..
0xffff28:00ff4062 00ff41f6 b@...A..
0xffff30:00ff41f6 00ff41f6 .A...A..
0xffff38:00ff41f6 00ff41f6 .A...A..
0xffff40:00ff41f6 00ff41f6 .A...A..
0xffff48:00ff41f6 00ff41f6 .A...A..
0xffff50:00ff41f6 00ff41f6 .A...A..
0xffff58:00ff41f6 00ff41f6 .A...A..
0xffff60:00ff40f8 00ff41f6 .@...A..
0xffff68:00ff41f6 00ff41f6 .A...A..
0xffff70:00ff41f6 00ff4124 .A..$A..
0xffff78:00ff4124 00ff41f6 $A...A..
0xffff80:00ff41f6 00ff41f6 .A...A..
0xffff88:00ff41f6 00ff41f6 .A...A..
0xffff90:00ff41f6 00ff41f6 .A...A..
0xffff98:00ff41f6 00ff41f6 .A...A..
0xffffa0:00ff41f6 00ff41f6 .A...A..
0xffffa8:00ff41e3 00ff41d0 .A...A..
0xffffb0:00ffdf40 00ffe000 @.......
0xffffb8:00ff41f6 00ff41f6 .A...A..
0xffffc0:00ff41f6 00ff41f6 .A...A..
0xffffc8:00ff41f6 00ff41f6 .A...A..
0xffffd0:00ff41f6 00ff41f6 .A...A..
0xffffd8:00ff41f6 00ff41f6 .A...A..
```

```
0x00fe885f:   LDW    HL,IX                dc 8b
0x00fe8861:   LDW    (0x12A8), HL         f2 a8 12 00 53
0x00fe8866:   bit    0, (0x120A)          f1 0a 12 c8
0x00fe886a:   jr     z, 0xfe88b0
0x00fe886c:   addw   hl, 11               db c8 0b 00
0x00fe8870:   jr     0xfe88b0             68 3e
0x00fe8872:   ???
```

```
XWA      :0x000000c
XBC      :0x0000210
XDE      :0x0000e70
XHL      :0x0000001
XIX      :0x0000015
XIY      :0x0800010
XIZ      :0x0000000
XSP      :0x0000010
INTNEST:     0xe70
SR       :     0xf80
PC       :0x00ffe04
SYSM     :        0x
IFF      :        0x
MAX      :        0x
RFP      :        0x
SF       :        0x
ZF       :        0x
HF       :        0x
VF       :        0x
NF       :        0x
CF       :        0x
```

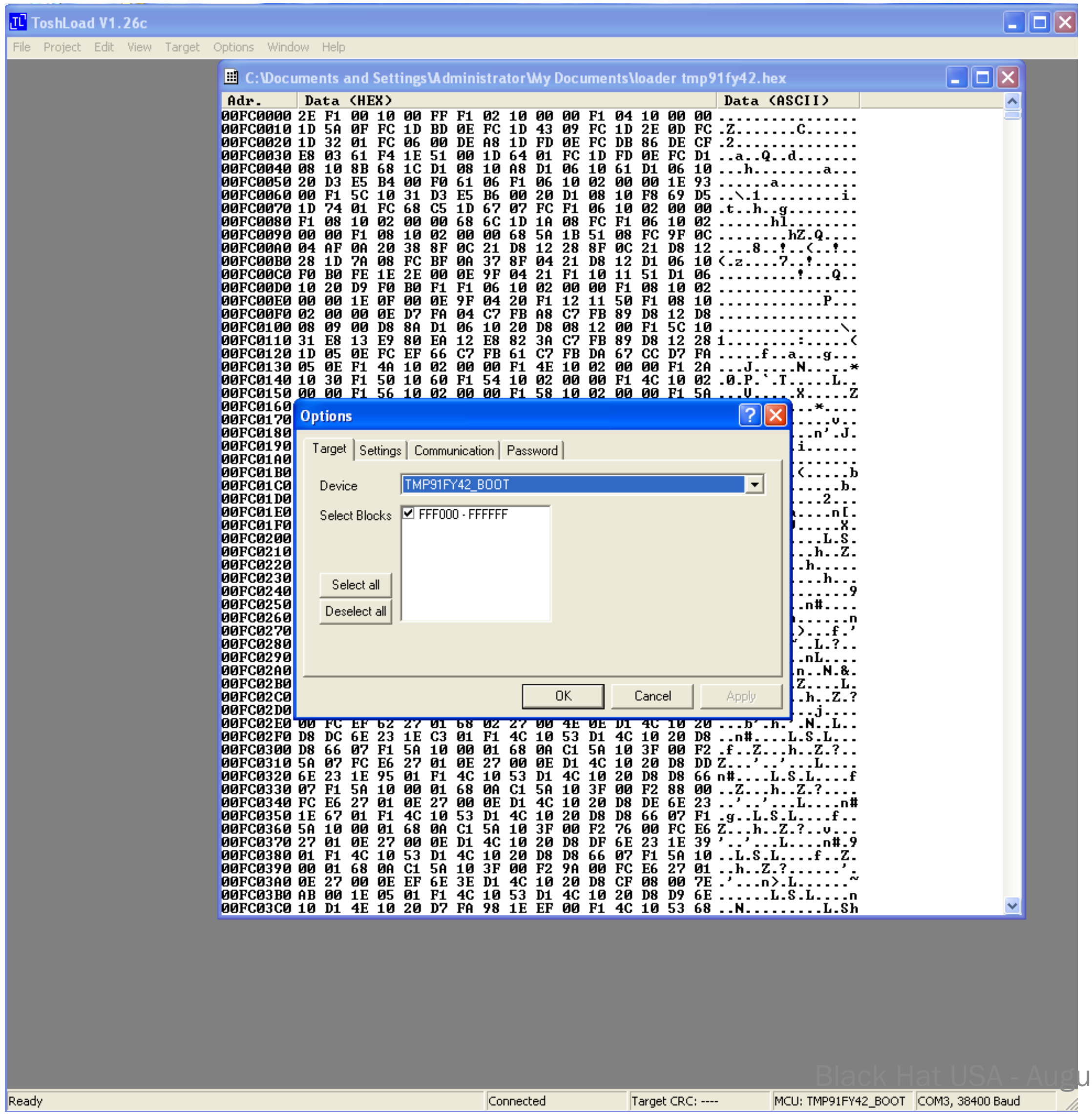
```
I-102-0100:Conne
I-102-0400:Resto
I-101-0D00:Anali
I-101-1000:Downl
I-101-1400:Objec
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
I-117-0103:SIMME
E-105-6400:Canno
E-105-6400:Canno
E-105-6400:Canno
E-105-6500:Canno
E-105-6400:Canno
E-105-6400:Canno
E-105-6400:Canno
E-105-6400:Canno
```

Windows XP?

Mirrored Here

https://github.com/colinoflynn/Toshiba-TLCS-900-L-Resources

Can you Read Back Bootloader?


ToshLoad V1.26c
File Project Edit View Target Options Window Help
C:\Documents and Settings\Administrator\My Documents\loader tmp91fy42.hex

```
Adr.     Data (HEX)                                       Data (ASCII)
00FC0000 2E F1 00 10 00 FF F1 02 10 00 00 F1 04 10 00 00 ................
00FC0010 1D 5A 0F FC 1D BD 0E FC 1D 43 09 FC 1D 2E 0D FC .Z.......C......
00FC0020 1D 32 01 FC 06 00 DE A8 1D FD 0E FC DB 86 DE CF .2..............
00FC0030 E8 03 61 F4 1E 51 00 1D 64 01 FC 1D FD 0E FC D1 ..a..Q..d.......
00FC0040 08 10 8B 68 1C D1 08 10 A8 D1 06 10 61 D1 06 10 ...h........a...
00FC0050 20 D3 E5 B4 00 F0 61 06 F1 06 10 02 00 00 1E 93 ......a.........
00FC0060 00 F1 5C 10 31 D3 E5 B6 00 20 D1 08 10 F8 69 D5 ..\.1.........i.
00FC0070 1D 74 01 FC 68 C5 1D 67 07 FC F1 06 10 02 00 00 .t..h..g........
00FC0080 F1 08 10 02 00 00 68 6C 1D 1A 08 FC F1 06 10 02 ......hl........
00FC0090 00 00 F1 08 10 02 00 00 68 5A 1B 51 08 FC 9F 0C ........hZ.Q....
00FC00A0 04 AF 0A 20 38 8F 0C 21 D8 12 28 8F 0C 21 D8 12 ....8..!..(..!..
00FC00B0 28 1D 7A 08 FC BF 0A 37 8F 04 21 D8 12 D1 06 10 (.z....7..!.....
00FC00C0 F0 B0 FE 1E 2E 00 0E 9F 04 21 F1 10 11 51 D1 06 .........!...Q..
00FC00D0 10 20 D9 F0 B0 F1 F1 06 10 02 00 00 F1 08 10 02 ................
00FC00E0 00 00 1E 0F 00 0E 9F 04 20 F1 12 11 50 F1 08 10 ............P...
00FC00F0 02 00 00 0E D7 FA 04 C7 FB A8 C7 FB 89 D8 12 D8 ................
00FC0100 08 09 00 D8 8A D1 06 10 20 D8 08 12 00 F1 5C 10 ..............\.
00FC0110 31 E8 13 E9 80 EA 12 E8 82 3A C7 FB 89 D8 12 28 1........:.....(
00FC0120 1D 05 0E FC EF 66 C7 FB 61 C7 FB DA 67 CC D7 FA .....f..a...g...
00FC0130 05 0E F1 4A 10 02 00 00 F1 4E 10 02 00 00 F1 2A ...J.....N.....*
00FC0140 10 30 F1 50 10 60 F1 54 10 02 00 00 F1 4C 10 02 .0.P.`.T.....L..
00FC0150 00 00 F1 56 10 02 00 00 F1 58 10 02 00 00 F1 5A ...V.....X.....Z
00FC0160                                                         ...*....
00FC0170                                                         .....v..
00FC0180                                                         ...n'.J.
00FC0190                                                         .i......
00FC01A0                                                         ........
00FC01B0                                                         .<.....b
00FC01C0                                                         ......b.
00FC01D0                                                         ....2...
00FC01E0                                                         .....n[.
00FC01F0                                                         ......X.
00FC0200                                                         ....L.S.
00FC0210                                                         ...h..Z.
00FC0220                                                         ..h.....
00FC0230                                                         .....h...
00FC0240                                                         ........9
00FC0250                                                         ..n#....
00FC0260                                                         .......n
00FC0270                                                         .>...f.'
00FC0280                                                         ...L.?..
00FC0290                                                         ..nL....
00FC02A0                                                         .n..N.&.
00FC02B0                                                         .Z....L.
00FC02C0                                                         ..h..Z.?
00FC02D0                                                         ...j....
00FC02E0 00 FC EF 62 27 01 68 02 27 00 4E 0E D1 4C 10 20 ...b'.h.'.N..L..
00FC02F0 D8 DC 6E 23 1E C3 01 F1 4C 10 53 D1 4C 10 20 D8 ..n#....L.S.L...
00FC0300 D8 66 07 F1 5A 10 00 01 68 0A C1 5A 10 3F 00 F2 .f..Z...h..Z.?..
00FC0310 5A 07 FC E6 27 01 0E 27 00 0E D1 4C 10 20 D8 DD Z...'..'...L....
00FC0320 6E 23 1E 95 01 F1 4C 10 53 D1 4C 10 20 D8 D8 66 n#....L.S.L....f
00FC0330 07 F1 5A 10 00 01 68 0A C1 5A 10 3F 00 F2 88 00 ..Z...h..Z.?....
00FC0340 FC E6 27 01 0E 27 00 0E D1 4C 10 20 D8 DE 6E 23 ..'..'...L....n#
00FC0350 1E 67 01 F1 4C 10 53 D1 4C 10 20 D8 D8 66 07 F1 .g..L.S.L....f..
00FC0360 5A 10 00 01 68 0A C1 5A 10 3F 00 F2 76 00 FC E6 Z...h..Z.?..v...
00FC0370 27 01 0E 27 00 0E D1 4C 10 20 D8 DF 6E 23 1E 39 '..'...L....n#.9
00FC0380 01 F1 4C 10 53 D1 4C 10 20 D8 D8 66 07 F1 5A 10 ..L.S.L....f..Z.
00FC0390 00 01 68 0A C1 5A 10 3F 00 F2 9A 00 FC E6 27 01 ..h..Z.?......'.
00FC03A0 0E 27 00 0E EF 6E 3E D1 4C 10 20 D8 CF 08 00 7E .'...n>.L......~
00FC03B0 AB 00 1E 05 01 F1 4C 10 53 D1 4C 10 20 D8 D9 6E ......L.S.L....n
00FC03C0 10 D1 4E 10 20 D7 FA 98 1E EF 00 F1 4C 10 53 68 ..N.........L.Sh
```

Device: TMP91FY42_BOOT

Select Blocks: ☑ FFF000 - FFFFFF

Ready | Connected | Target CRC: ---- | MCU: TMP91FY42_BOOT | COM3, 38400 Baud

Segger “ToshLoad” can read-back bootloader (ROM) section!

Watch for how ROM remaps when in bootloader (single boot) mode.

(I made a Python version of this program so you *don’t* need Windows XP)

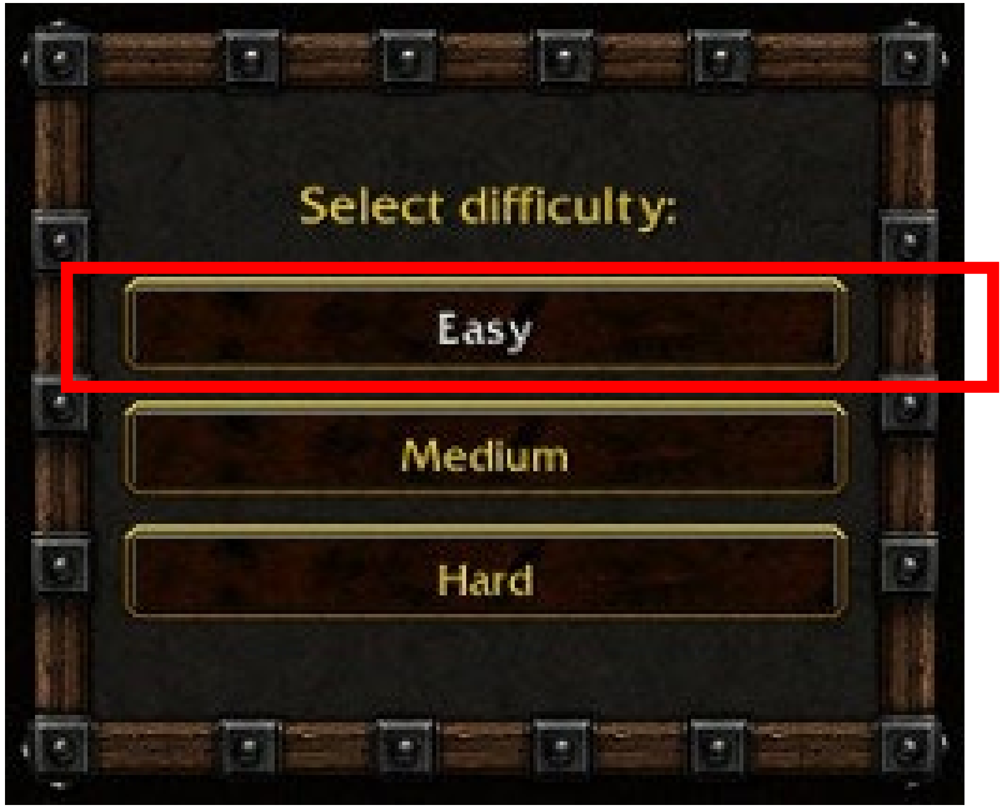
```
FUNCTION START: Receive & Verify Password
00fff2a2 CALR    0x0FFF5EF <-- RX
...
00fff2ce JR      NZ,0x0FFF2D5
00fff2d0 DJNZB   C,0x0FFF2C9
00fff2d3 JR      0x0FFF2D7
00fff2d5 LDB     L,0x1 <-- L is flag, if set to 1 comparison failed
00fff2d7 LDW     BC,0x0C <-- 12 bytes to compare
00fff2da LDL     XIX,(0x0FFF00C) <-- Points to 0004FEF4 (PW)
00fff2df LDB     RH1,0x0
00fff2e2 LDB     W,(XIX+) <--Load byte into W, inc XIX ptr (loop)
00fff2e5 CALR    0x0FFF635 <--- RX assumed
00fff2e8 CPB     W,A <--Compare W & A
00fff2ea JR      Z,0x0FFF2EE <-- Compare OK, skip fail set
00fff2ec LDB     L,0x1 <--Set 'fail' flag
00fff2ee DJNZW   BC,0x0FFF2E2 <--Jump to next byte (12 times)
00fff2f1 CALR    0x0FFF67B <-- checksum
00fff2f4 RET
```

```
FUNCTION START: RAM WRITE FUNCTION
00fff2f5 CALR    0x0FFF75F  <-- Load protection status
00fff2f8 CPB     A,0x0FF <-- Compare protection status
00fff2fb JR      NZ,0x0FFF290 <-- Send error if protection enabled
00fff2fd CALR    0x0FFF2A2 <-- PW Check
00fff300 CPB     RE1,0x0
00fff303 JR      NZ,0x0FFF28A
00fff305 CPB     RL1,0x0
00fff308 JR      NZ,0x0FFF29C <-- Error
00fff30a CPB     L,0x0
00fff30c JR      NZ,0x0FFF29C <-- Error
00fff30e CALR    0x0FFF5EF <- TX
00fff311 LDB     RH1,0x0
00fff314 CALR    0x0FFF635 <--
00fff317 LDB     QIXH,A
```

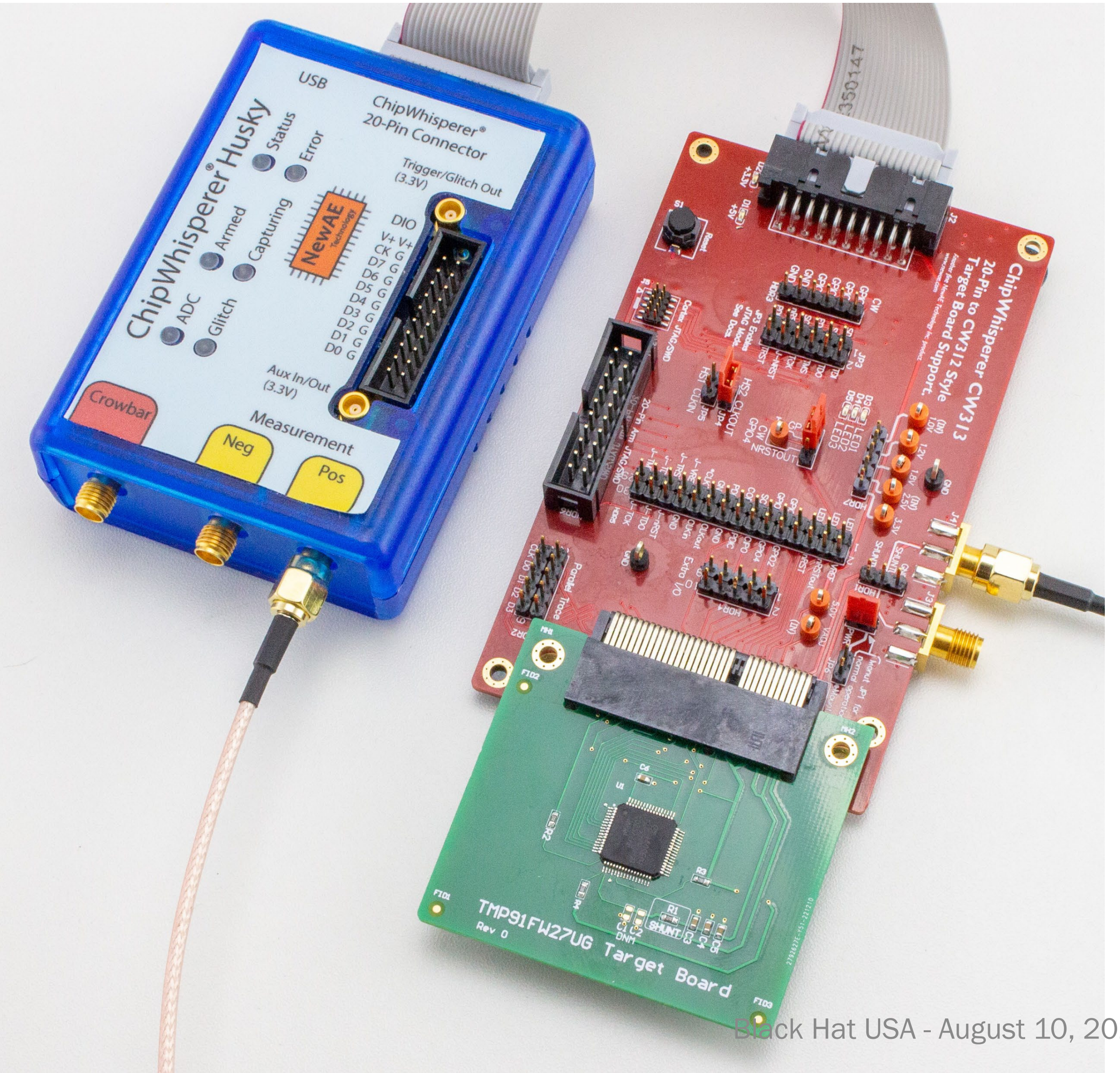
Important Take-Aways (for next stage)

1. Password check has slight code-flow dependency.

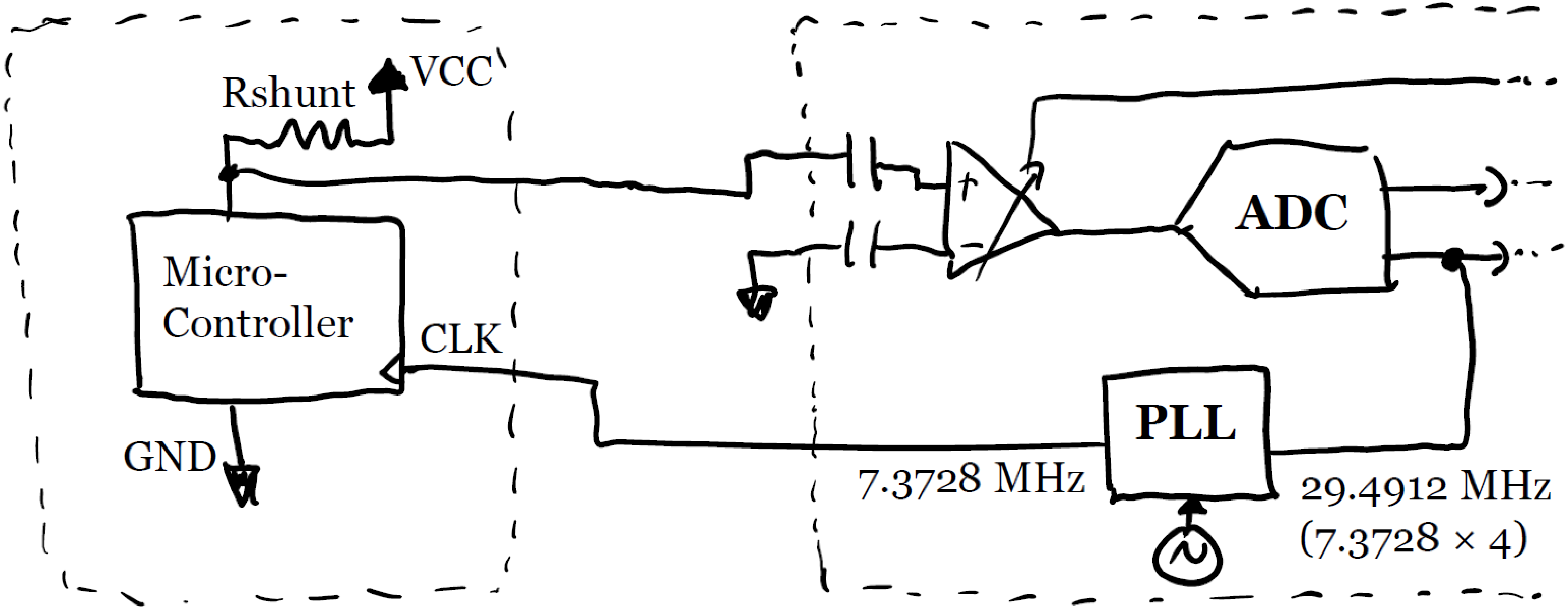
2. Fuse byte check has obvious fault injection location.


Select difficulty:
Easy
Medium
Hard

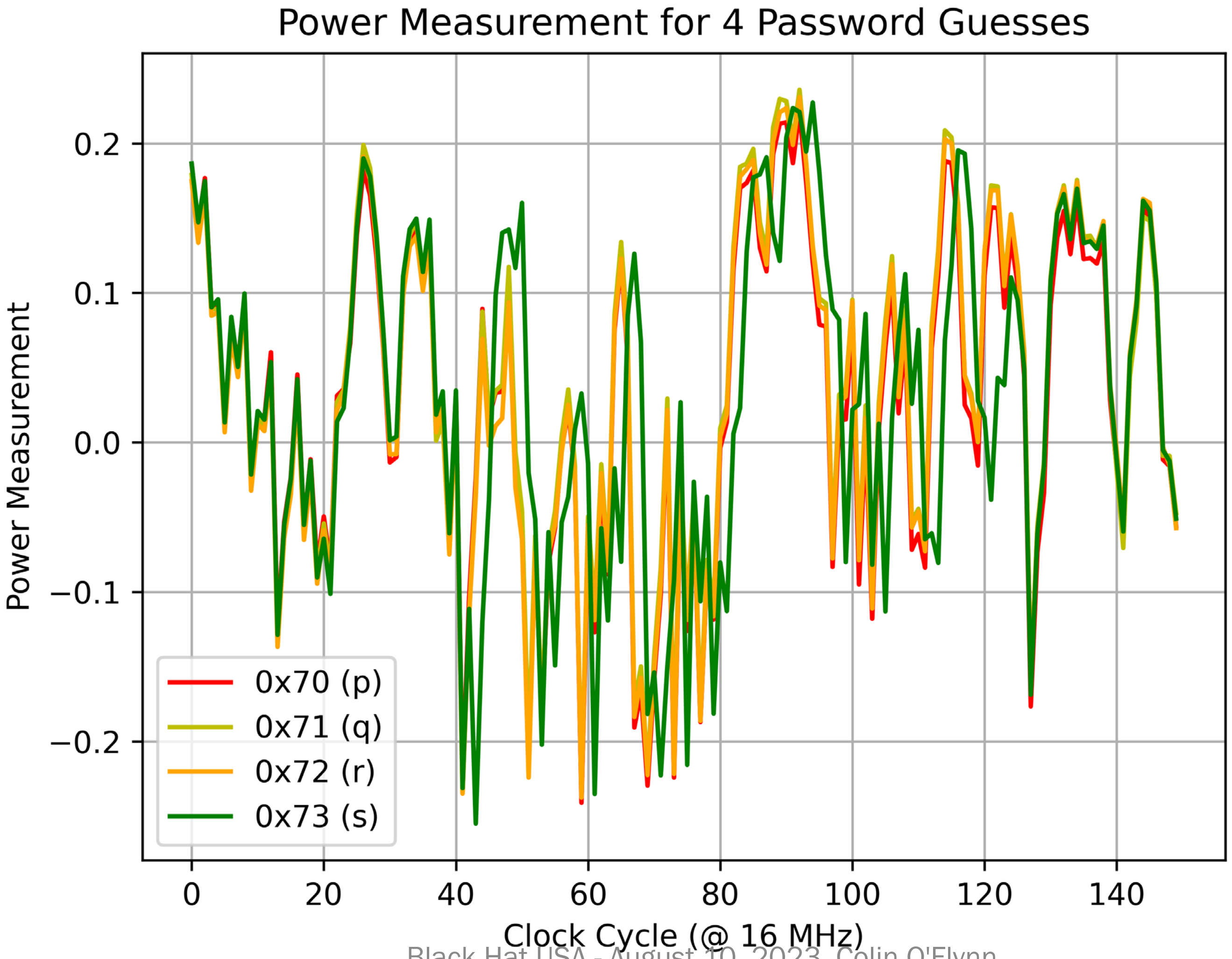
ChipWhisperer-Husky Intro

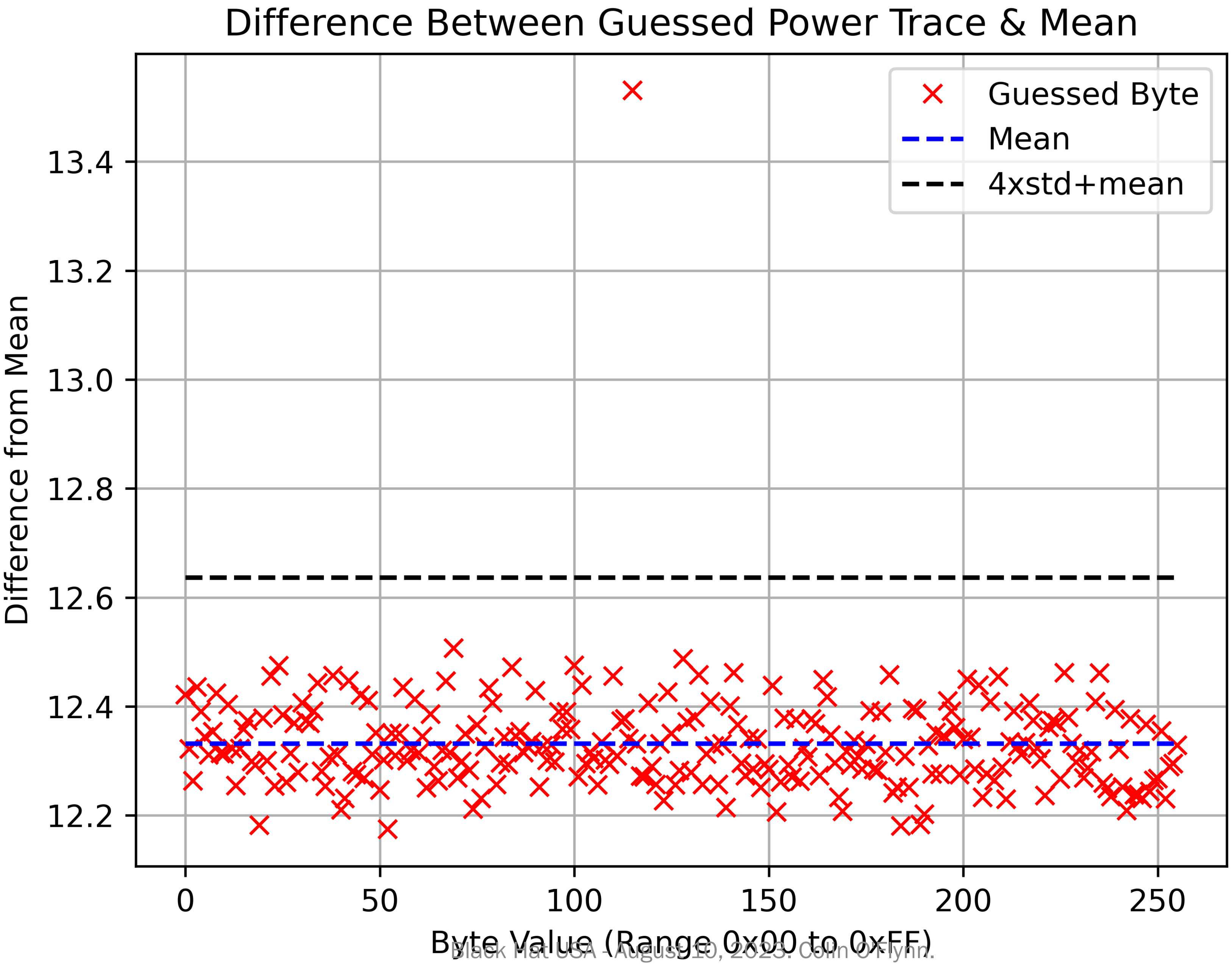


Power Analysis?



Easy-Mode Level 1: Password Power Analysis



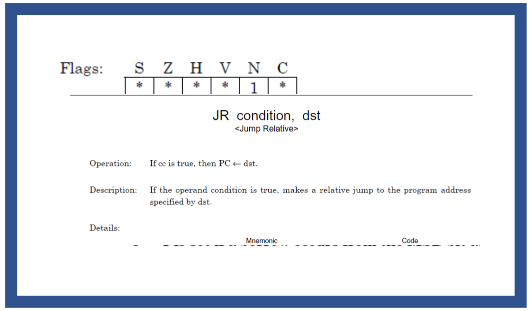
Difference Between Guessed Power Trace & Mean
Difference from Mean
13.4
13.2
13.0
12.8
12.6
12.4
12.2
0
50
100
150
200
250
Byte Value (Range 0x00 to 0xFF)
Guessed Byte
Mean
4xstd+mean

Fault Injection?

```
FUNCTION START: RAM WRITE FUNCTION
00fff2f5 CALR    0x0FFF75F  <-- Load protection status
00fff2f8 CPB     A,0x0FF <-- Compare protection status
00fff2fb JR      NZ,0x0FFF290 <-- Send error if protection enabled
00fff2fd CALR    0x0FFF2A2 <-- PW Check
```

Fetch

Decode



Flags: S Z H V N C

S	Z	H	V	N	C
*	*	*	*	1	*

JR condition, dst

<Jump Relative>

Operation: If cc is true, then PC ← dst.

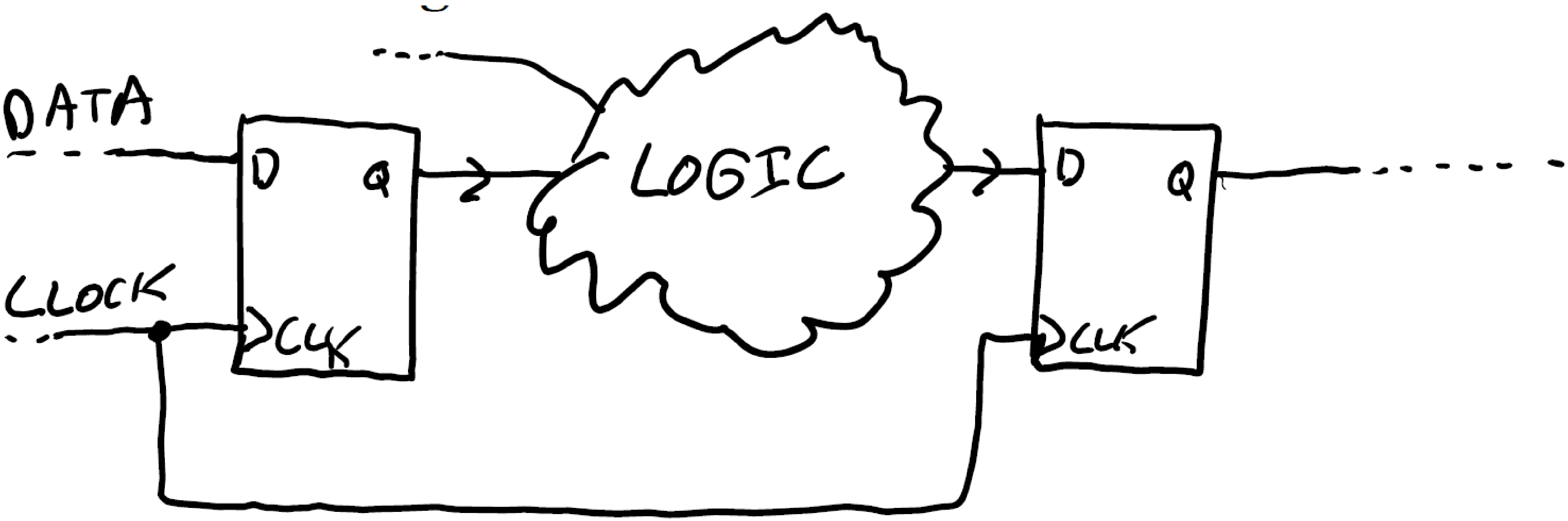
Description: If the operand condition is true, makes a relative jump to the program address specified by dst.

Details:

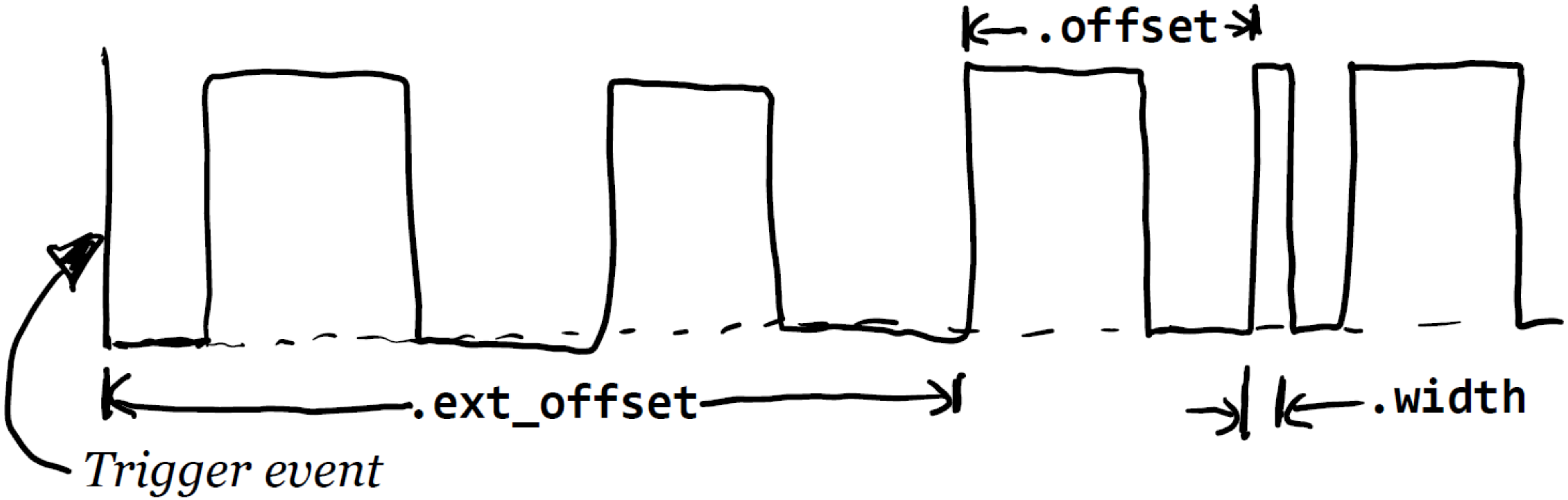
Mnemonic Code

Execute

Fault Injection?



Clock Fault Injection



Easy-Mode Level 2: Fault Injection Tuning

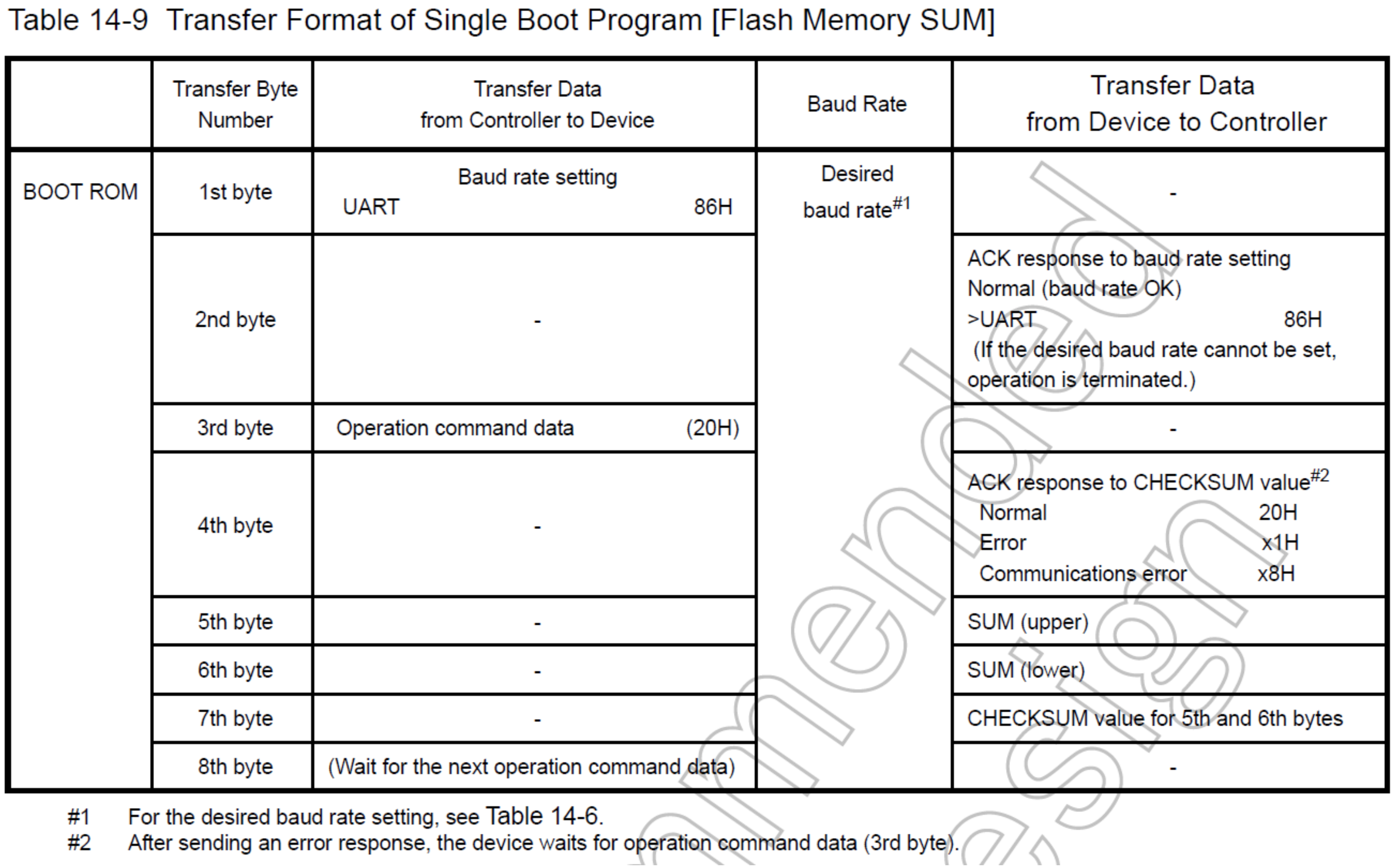


Table 14-9 Transfer Format of Single Boot Program [Flash Memory SUM]

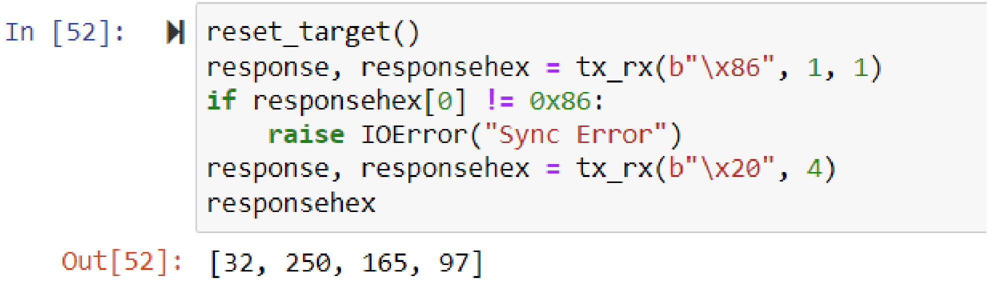
	Transfer Byte Number	Transfer Data from Controller to Device	Baud Rate	Transfer Data from Device to Controller
BOOT ROM	1st byte	Baud rate setting UART 86H	Desired baud rate[#1]	-
	2nd byte	-		ACK response to baud rate setting Normal (baud rate OK) >UART 86H (If the desired baud rate cannot be set, operation is terminated.)
	3rd byte	Operation command data (20H)		-
	4th byte	-		ACK response to CHECKSUM value[#2] Normal 20H Error x1H Communications error x8H
	5th byte	-		SUM (upper)
	6th byte	-		SUM (lower)
	7th byte	-		CHECKSUM value for 5th and 6th bytes
	8th byte	(Wait for the next operation command data)		-

#1 For the desired baud rate setting, see Table 14-6.

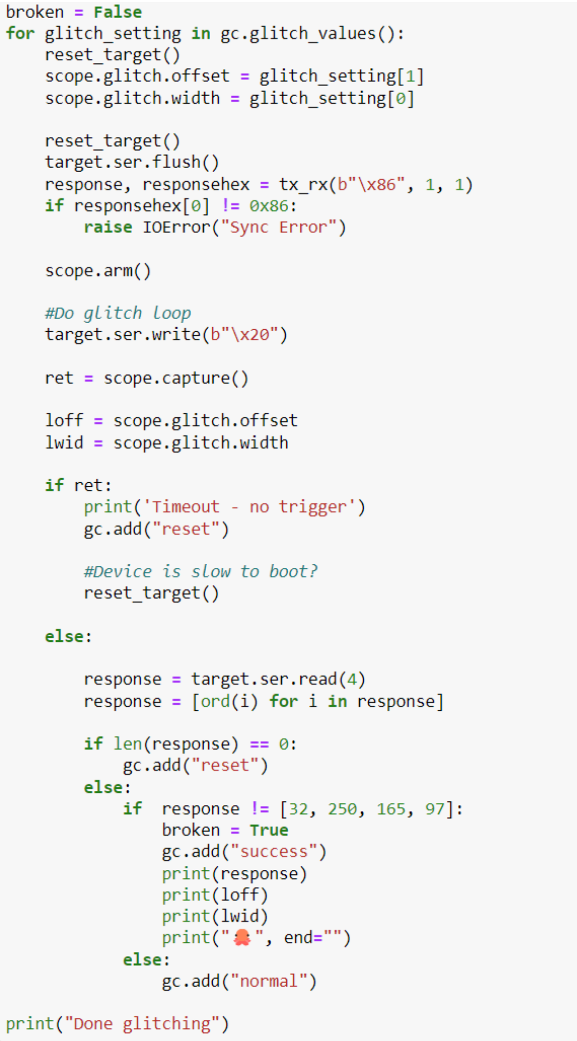
#2 After sending an error response, the device waits for operation command data (3rd byte).

Flash memory SUM = MANY opportunities to glitch result (entire SUM operation)

Fault Injection Setup / Demo

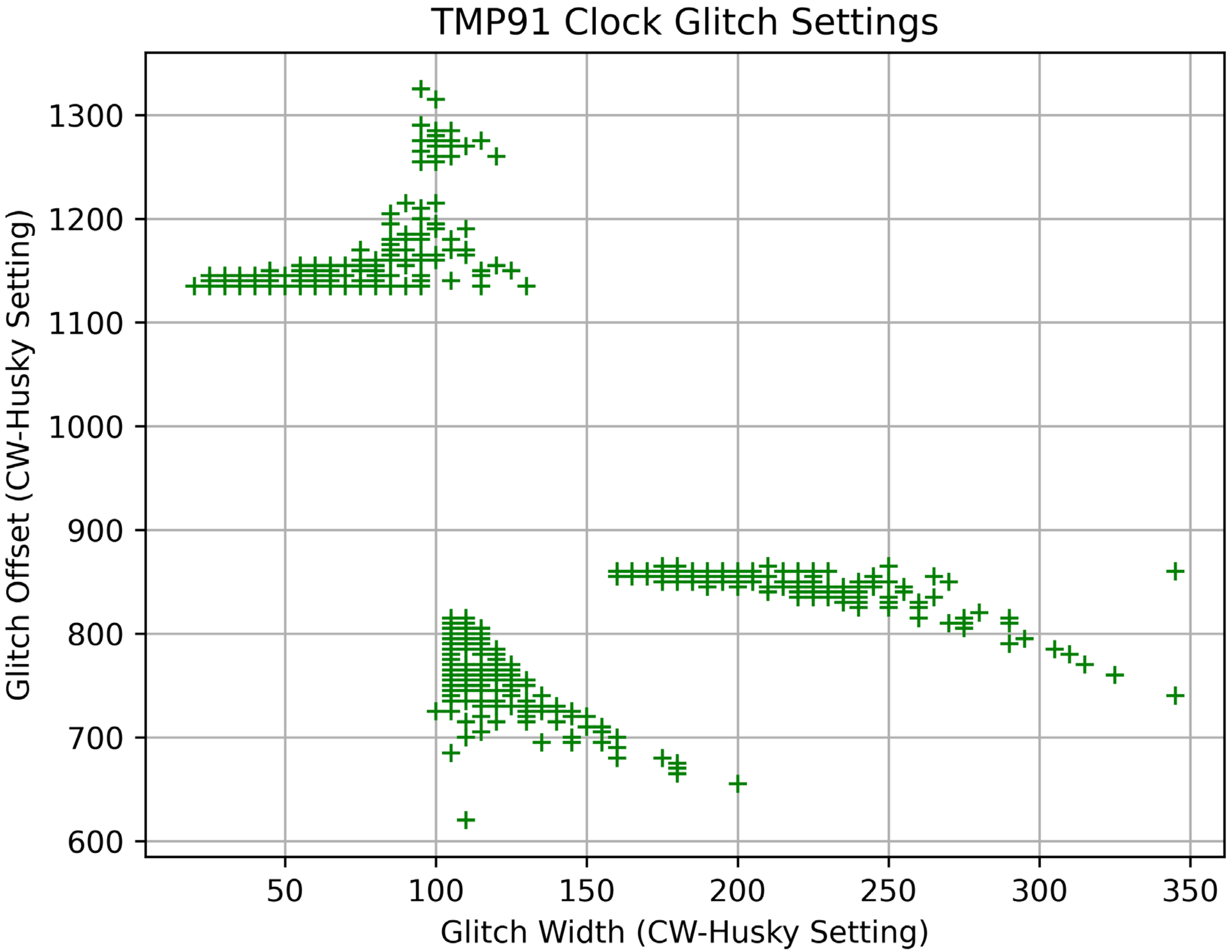


```
In [52]:  reset_target()
          response, responsehex = tx_rx(b"\x86", 1, 1)
          if responsehex[0] != 0x86:
              raise IOError("Sync Error")
          response, responsehex = tx_rx(b"\x20", 4)
          responsehex

Out[52]:  [32, 250, 165, 97]
```

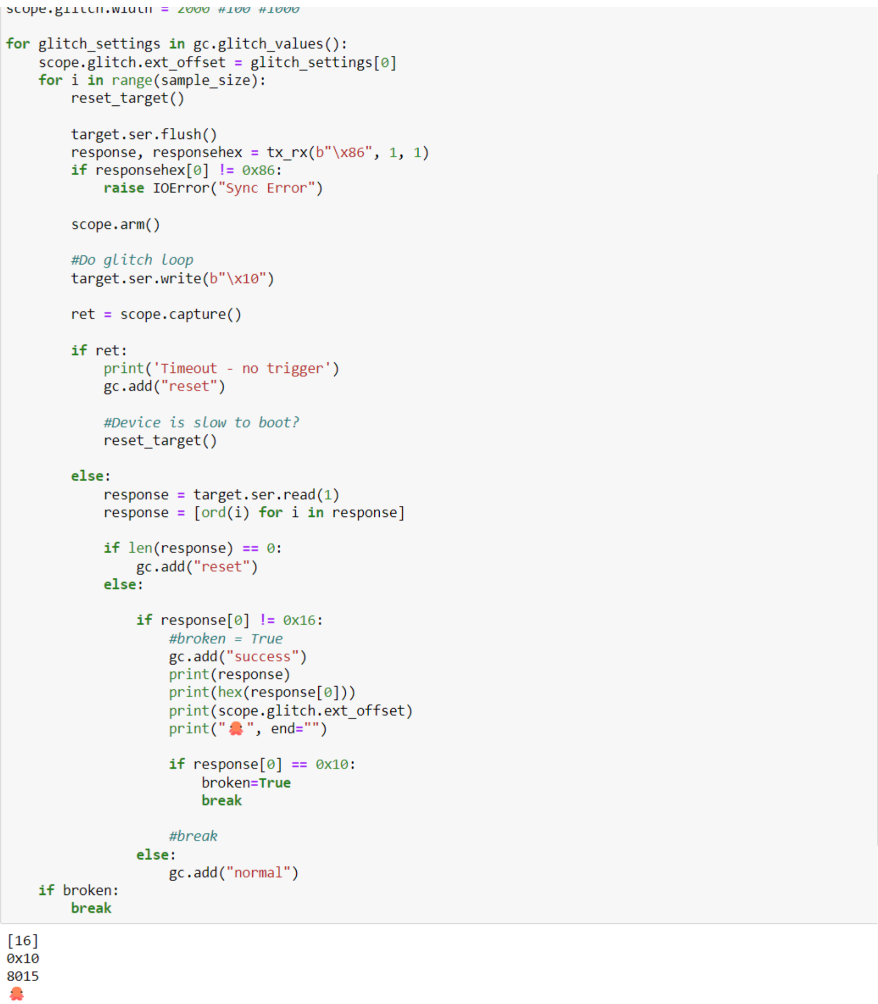


```
broken = False
for glitch_setting in gc.glitch_values():
    reset_target()
    scope.glitch.offset = glitch_setting[1]
    scope.glitch.width = glitch_setting[0]

    reset_target()
    target.ser.flush()
    response, responsehex = tx_rx(b"\x86", 1, 1)
    if responsehex[0] != 0x86:
        raise IOError("Sync Error")

    scope.arm()

    #Do glitch loop
    target.ser.write(b"\x20")

    ret = scope.capture()

    loff = scope.glitch.offset
    lwid = scope.glitch.width

    if ret:
        print('Timeout - no trigger')
        gc.add("reset")

        #Device is slow to boot?
        reset_target()

    else:

        response = target.ser.read(4)
        response = [ord(i) for i in response]

        if len(response) == 0:
            gc.add("reset")
        else:
            if  response != [32, 250, 165, 97]:
                broken = True
                gc.add("success")
                print(response)
                print(loff)
                print(lwid)
                print("🐙", end="")
            else:
                gc.add("normal")

print("Done glitching")
```

Fault Injection Results (SUM Corruption)

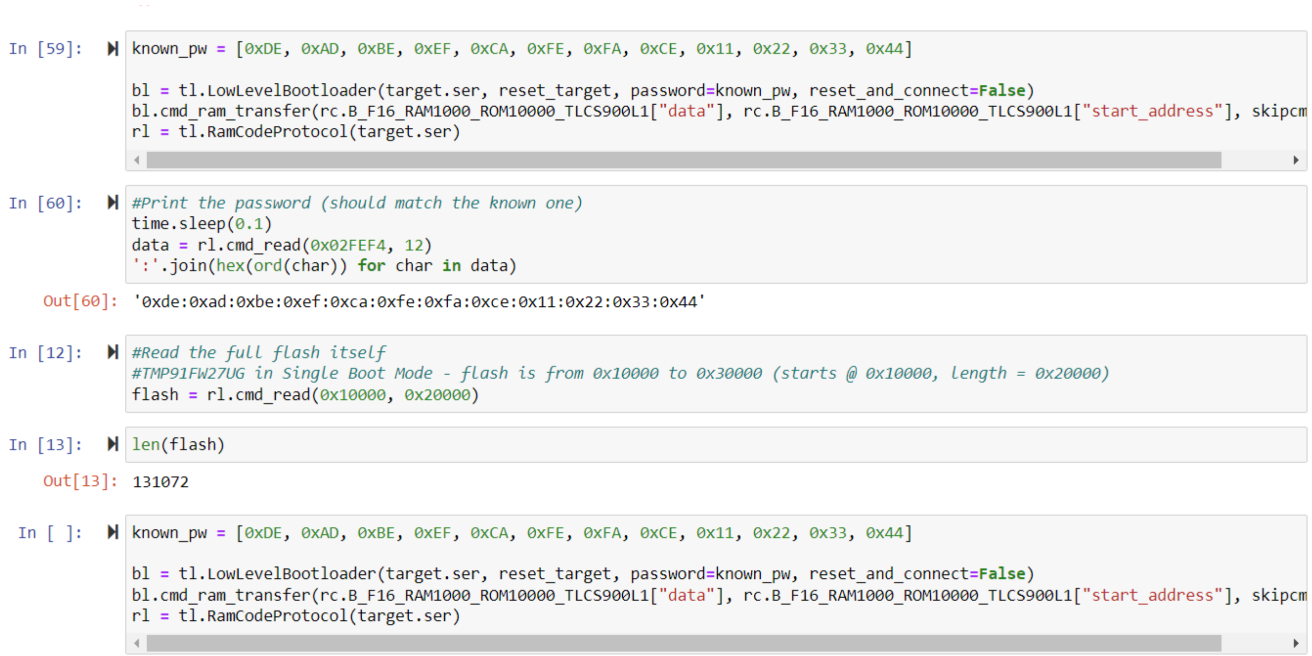


Easy-Mode Level 3: Fault Injection Attack



```
scope.glitch.width = 2000 #100 #1000

for glitch_settings in gc.glitch_values():
    scope.glitch.ext_offset = glitch_settings[0]
    for i in range(sample_size):
        reset_target()

        target.ser.flush()
        response, responsehex = tx_rx(b"\x86", 1, 1)
        if responsehex[0] != 0x86:
            raise IOError("Sync Error")

        scope.arm()

        #Do glitch loop
        target.ser.write(b"\x10")

        ret = scope.capture()

        if ret:
            print('Timeout - no trigger')
            gc.add("reset")

            #Device is slow to boot?
            reset_target()

        else:
            response = target.ser.read(1)
            response = [ord(i) for i in response]

            if len(response) == 0:
                gc.add("reset")
            else:

                if response[0] != 0x16:
                    #broken = True
                    gc.add("success")
                    print(response)
                    print(hex(response[0]))
                    print(scope.glitch.ext_offset)
                    print("🐙", end="")

                    if response[0] == 0x10:
                        broken=True
                        break

                    #break
                else:
                    gc.add("normal")
    if broken:
        break
```

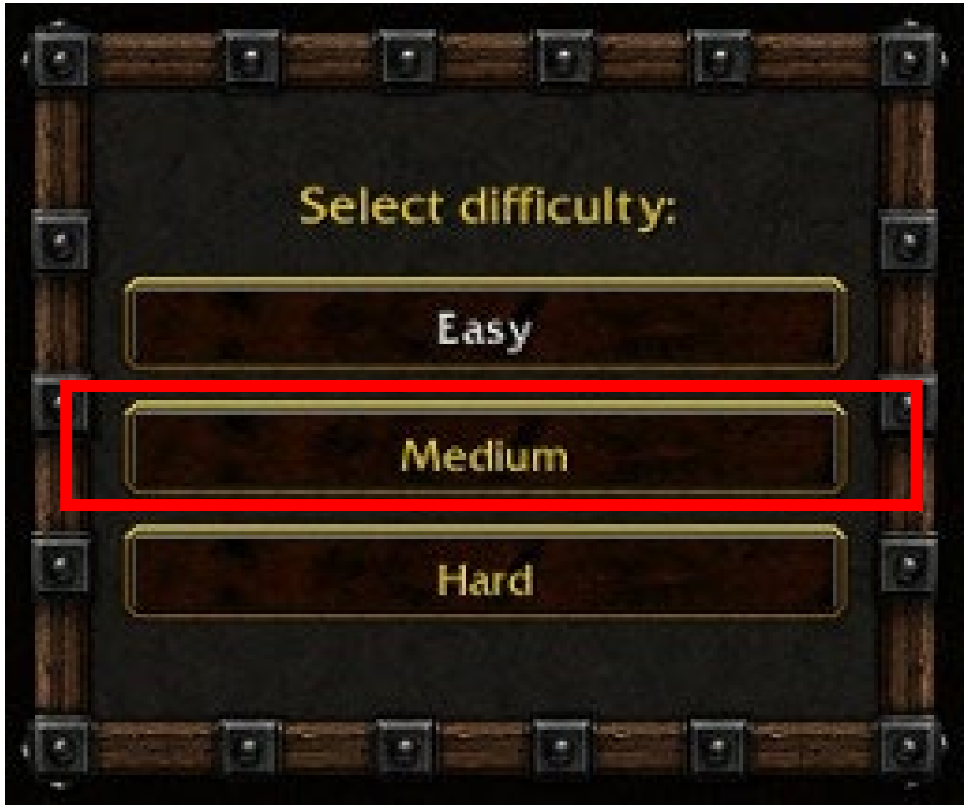
```
[16]
0x10
8015
🐙
```

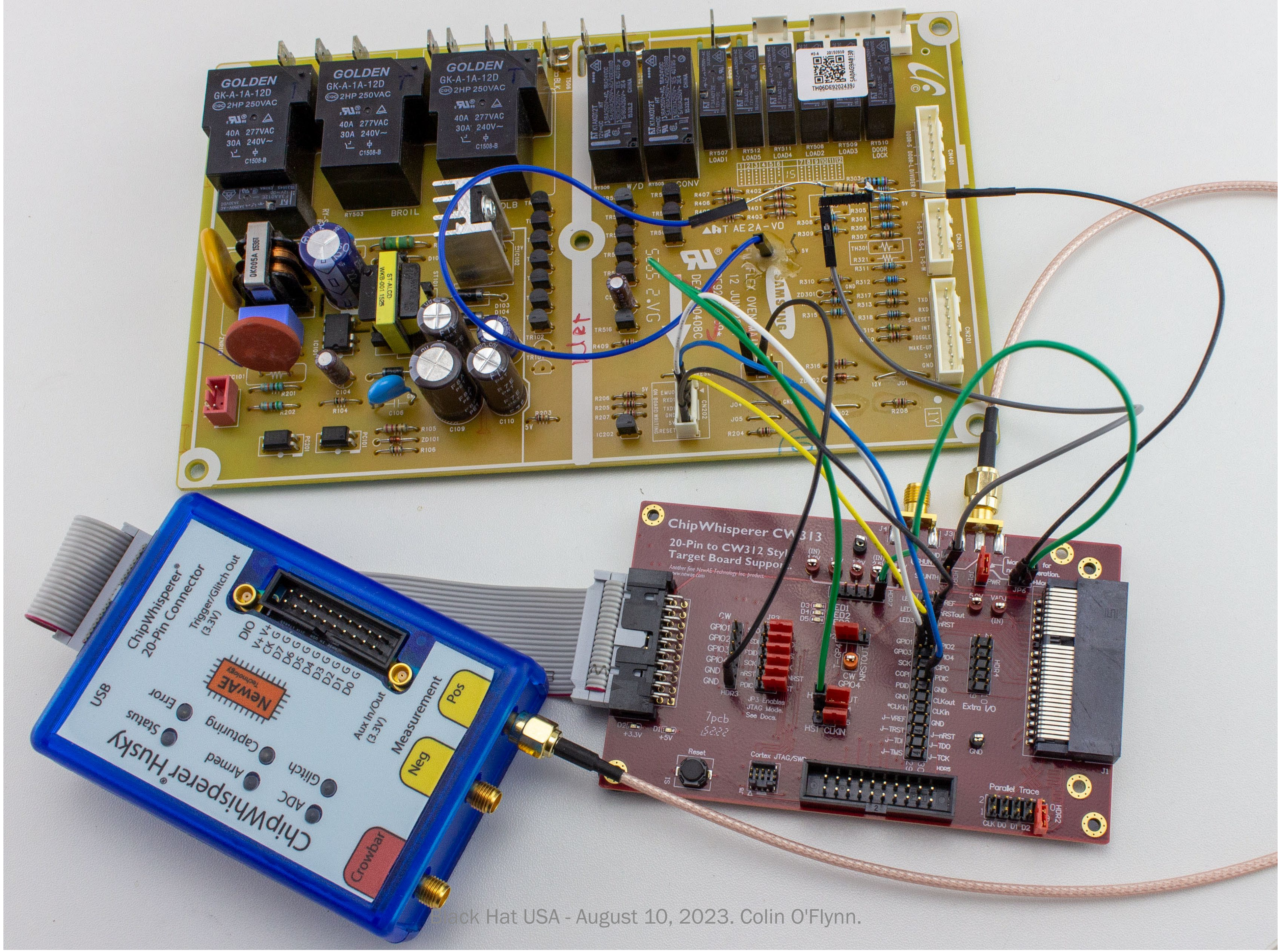


```
In [59]:  known_pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44]

           bl = tl.LowLevelBootloader(target.ser, reset_target, password=known_pw, reset_and_connect=False)
           bl.cmd_ram_transfer(rc.B_F16_RAM1000_ROM10000_TLCS900L1["data"], rc.B_F16_RAM1000_ROM10000_TLCS900L1["start_address"], skipcm
           rl = tl.RamCodeProtocol(target.ser)

In [60]:  #Print the password (should match the known one)
           time.sleep(0.1)
           data = rl.cmd_read(0x02FEF4, 12)
           ':'.join(hex(ord(char)) for char in data)

Out[60]:  '0xde:0xad:0xbe:0xef:0xca:0xfe:0xfa:0xce:0x11:0x22:0x33:0x44'

In [12]:  #Read the full flash itself
           #TMP91FW27UG in Single Boot Mode - flash is from 0x10000 to 0x30000 (starts @ 0x10000, length = 0x20000)
           flash = rl.cmd_read(0x10000, 0x20000)

In [13]:  len(flash)

Out[13]:  131072

In [ ]:   known_pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44]

           bl = tl.LowLevelBootloader(target.ser, reset_target, password=known_pw, reset_and_connect=False)
           bl.cmd_ram_transfer(rc.B_F16_RAM1000_ROM10000_TLCS900L1["data"], rc.B_F16_RAM1000_ROM10000_TLCS900L1["start_address"], skipcm
           rl = tl.RamCodeProtocol(target.ser)
```

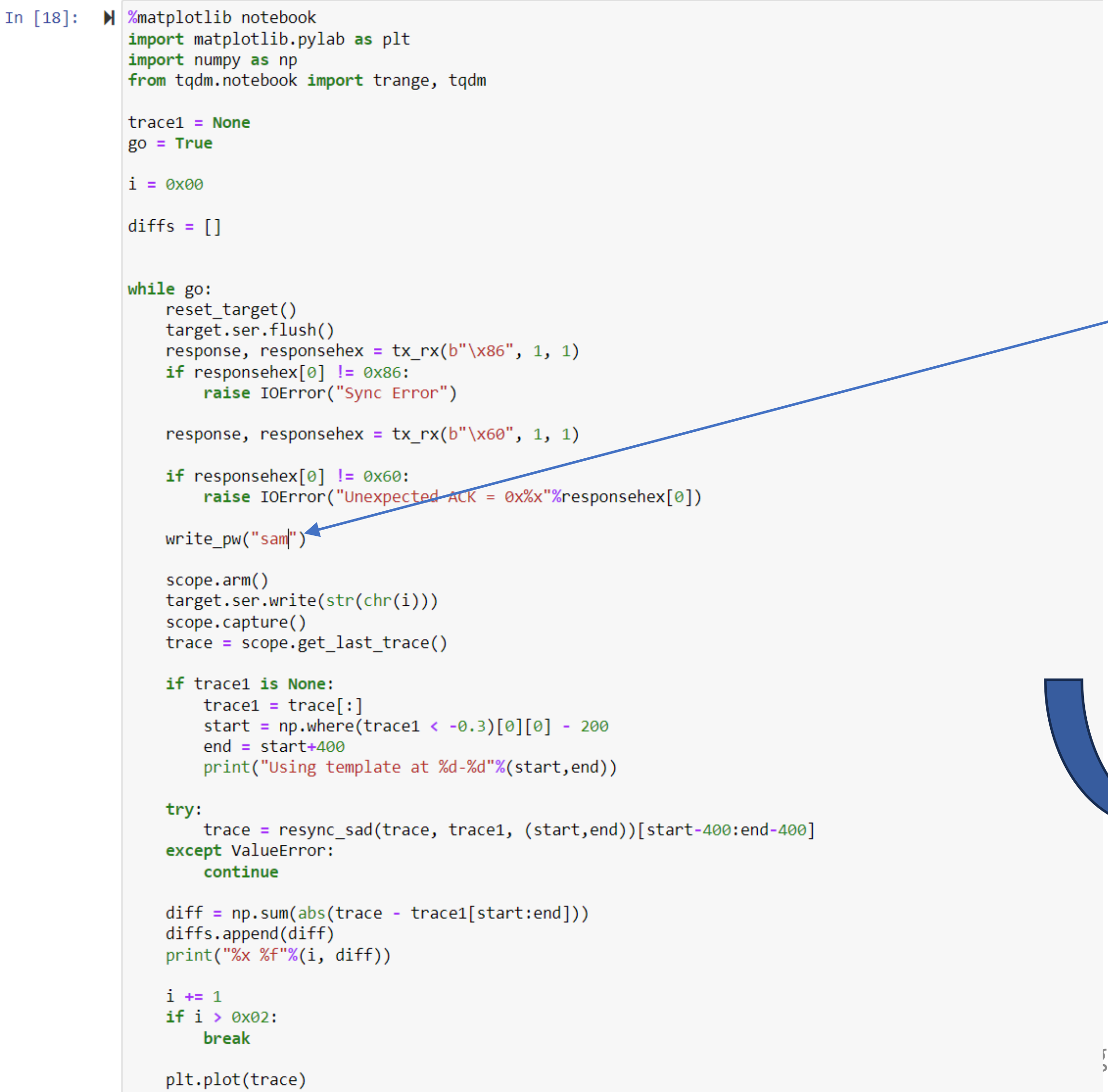
Skills & Resources

- Python class for communicating & programming TMP91 (including 2nd stage bootloader communications).
- Timing on power analysis.
- Rough timing / details on fault injection.


Select difficulty:
Easy
Medium
Hard


ChipWhisperer-Husky
ChipWhisperer 20-Pin Connector
Trigger/Glitch Out (3.3V)
Aux In/Out (3.3V)
Measurement
Pos
Neg
Crowbar
USB
Status
Error
Armed
Capturing
ADC
Glitch
NewAE Technology
ChipWhisperer CW313
20-Pin to CW312 Style Target Board Support
GOLDEN GK-A-1A-12D
Parallel Trace
Cortex JTAG/SWD
Reset
Extra I/O

Medium-Mode Level 1: Power Analysis

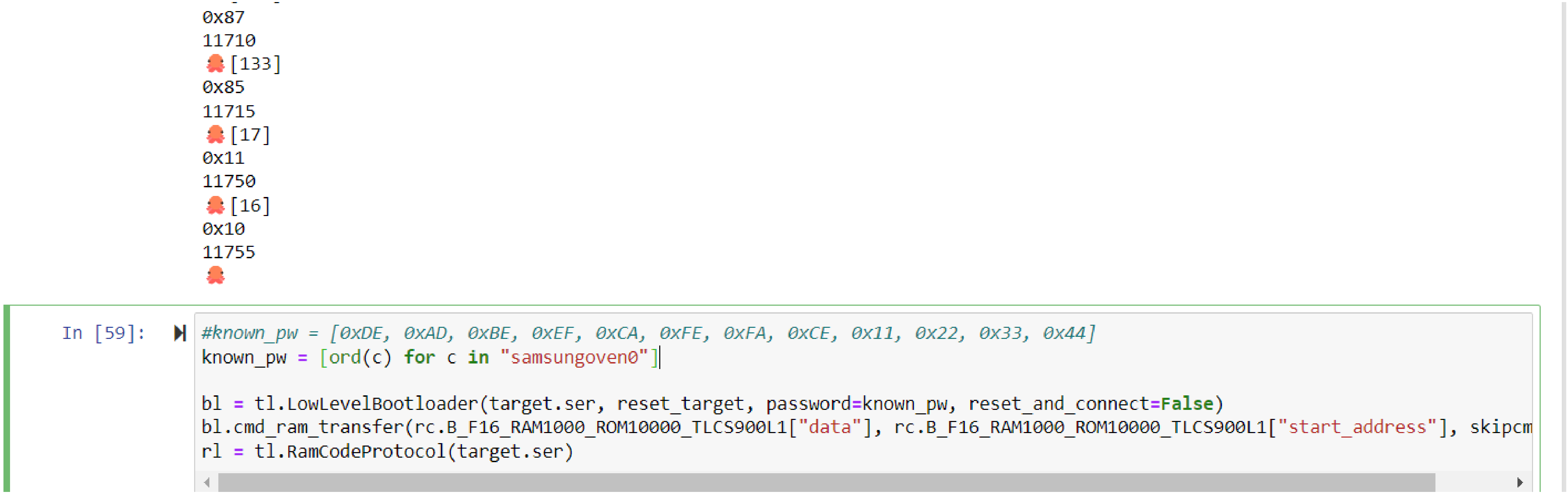


```
In [18]:  %matplotlib notebook
          import matplotlib.pylab as plt
          import numpy as np
          from tqdm.notebook import trange, tqdm

          trace1 = None
          go = True

          i = 0x00

          diffs = []


          while go:
              reset_target()
              target.ser.flush()
              response, responsehex = tx_rx(b"\x86", 1, 1)
              if responsehex[0] != 0x86:
                  raise IOError("Sync Error")

              response, responsehex = tx_rx(b"\x60", 1, 1)

              if responsehex[0] != 0x60:
                  raise IOError("Unexpected ACK = 0x%x"%responsehex[0])

              write_pw("sam")

              scope.arm()
              target.ser.write(str(chr(i)))
              scope.capture()
              trace = scope.get_last_trace()

              if trace1 is None:
                  trace1 = trace[:]
                  start = np.where(trace1 < -0.3)[0][0] - 200
                  end = start+400
                  print("Using template at %d-%d"%(start,end))

              try:
                  trace = resync_sad(trace, trace1, (start,end))[start-400:end-400]
              except ValueError:
                  continue

              diff = np.sum(abs(trace - trace1[start:end]))
              diffs.append(diff)
              print("%x %f"%(i, diff))

              i += 1
              if i > 0x02:
                  break

              plt.plot(trace)
```

Sending known part of password, then do the attack on next unknown byte

s..a..m..s..u..n..g..o..v..e..n..0

Medium-Mode Level 2: Fault Injection



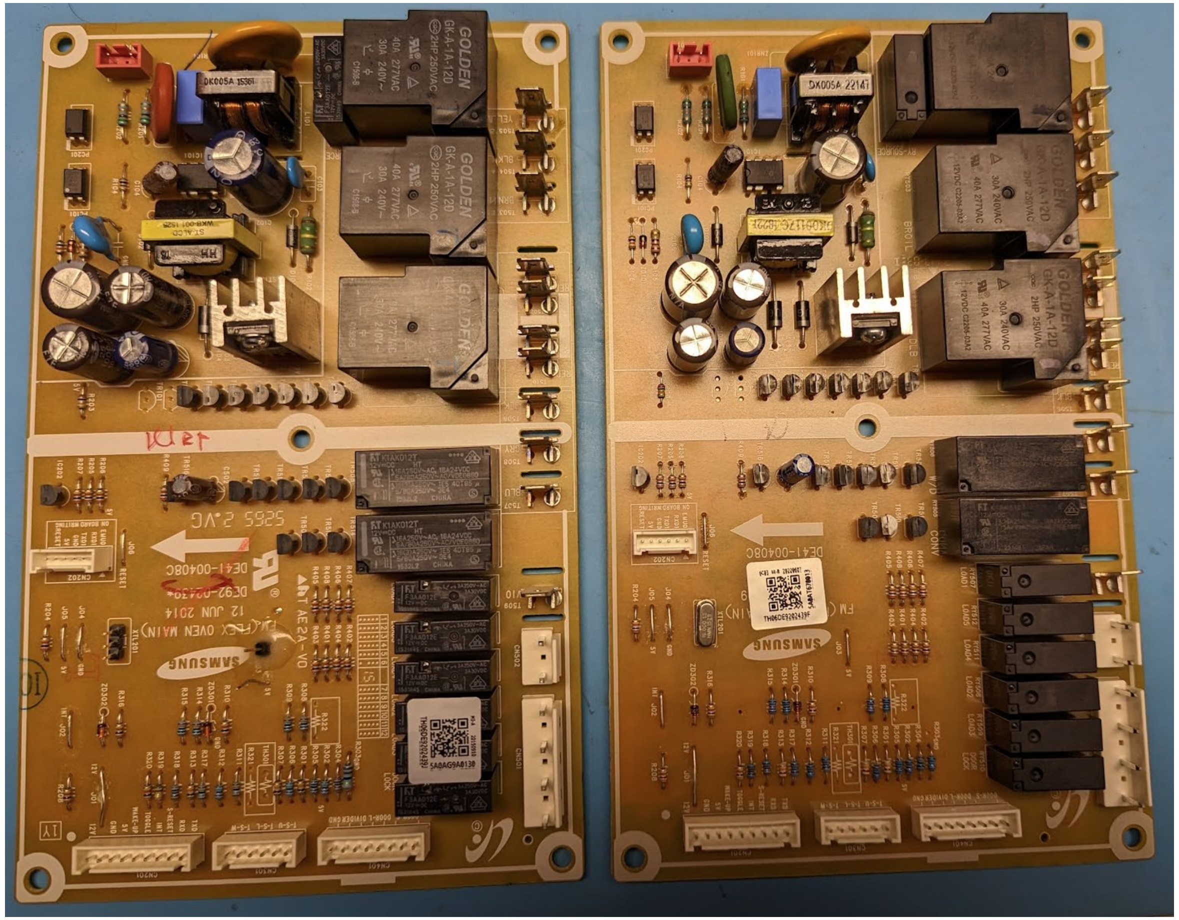
```
0x87
11710
🐙[133]
0x85
11715
🐙[17]
0x11
11750
🐙[16]
0x10
11755
🐙
```

```
In [59]:
#known_pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44]
known_pw = [ord(c) for c in "samsungoven0"]

bl = tl.LowLevelBootloader(target.ser, reset_target, password=known_pw, reset_and_connect=False)
bl.cmd_ram_transfer(rc.B_F16_RAM1000_ROM10000_TLCS900L1["data"], rc.B_F16_RAM1000_ROM10000_TLCS900L1["start_address"], skipcm
rl = tl.RamCodeProtocol(target.ser)
```

```
In [11]:  resp = rcp.cmd_read(0x10000, 0x100)

In [12]:  resp

Out[12]: 'ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ                                                         ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
          ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ                                                        ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
          ÿÿÿÿÿÿÿÿÿ'
```

```
In [7]:  bl = tl
         #bl.cmd
         #bl.cmd
         bl.cmd_

         Read: n
         Write:
```

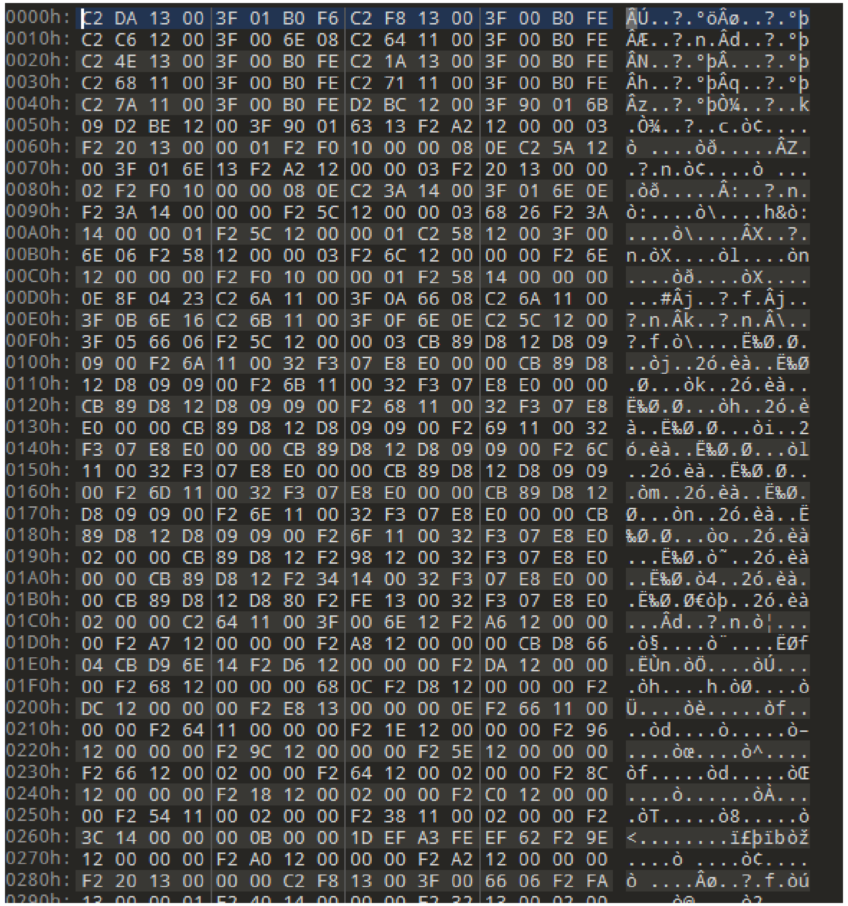
$$ → Samsung Parts Department



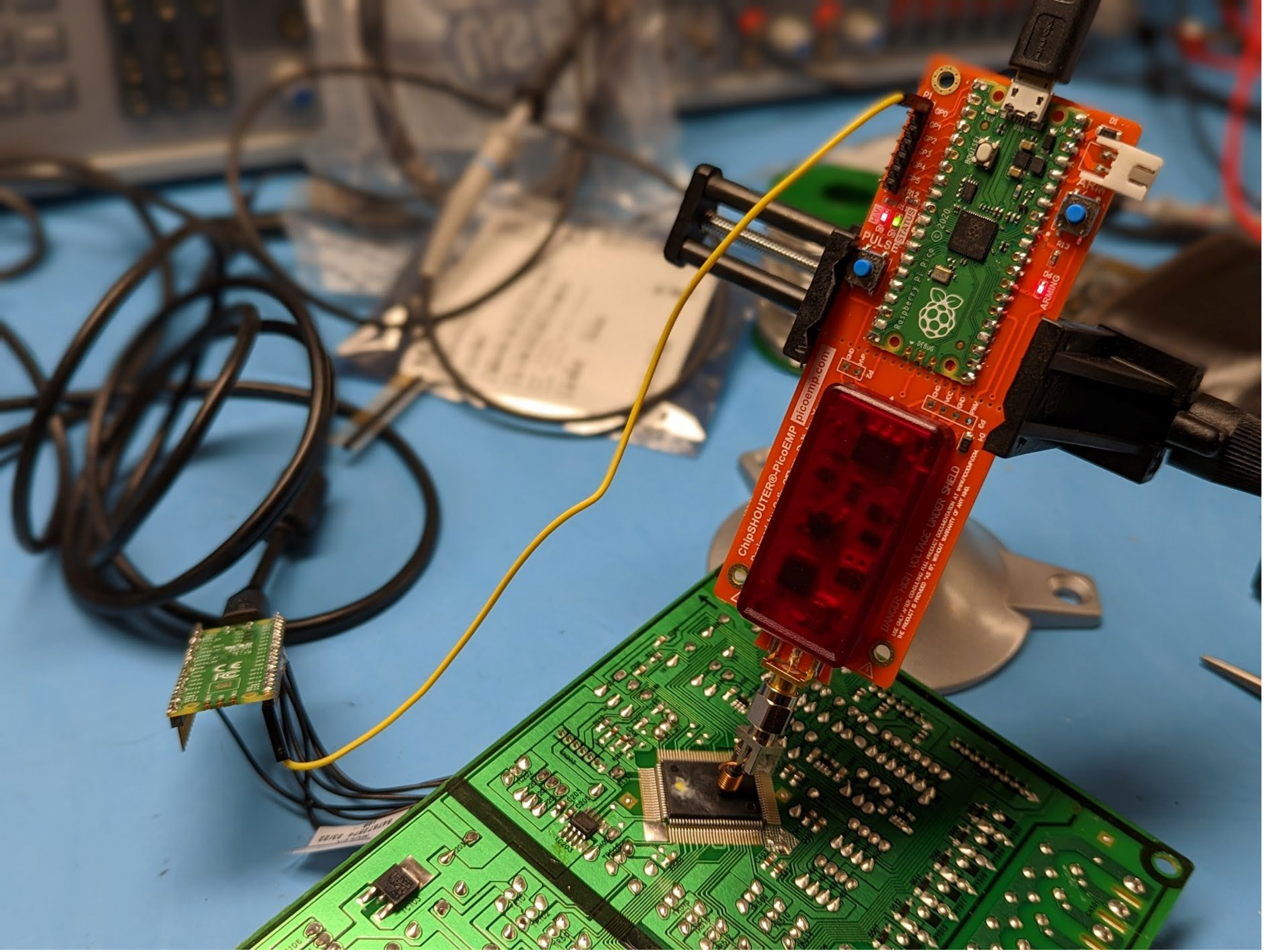
Did they have problems with returns?

Or eeded on the

ards!!



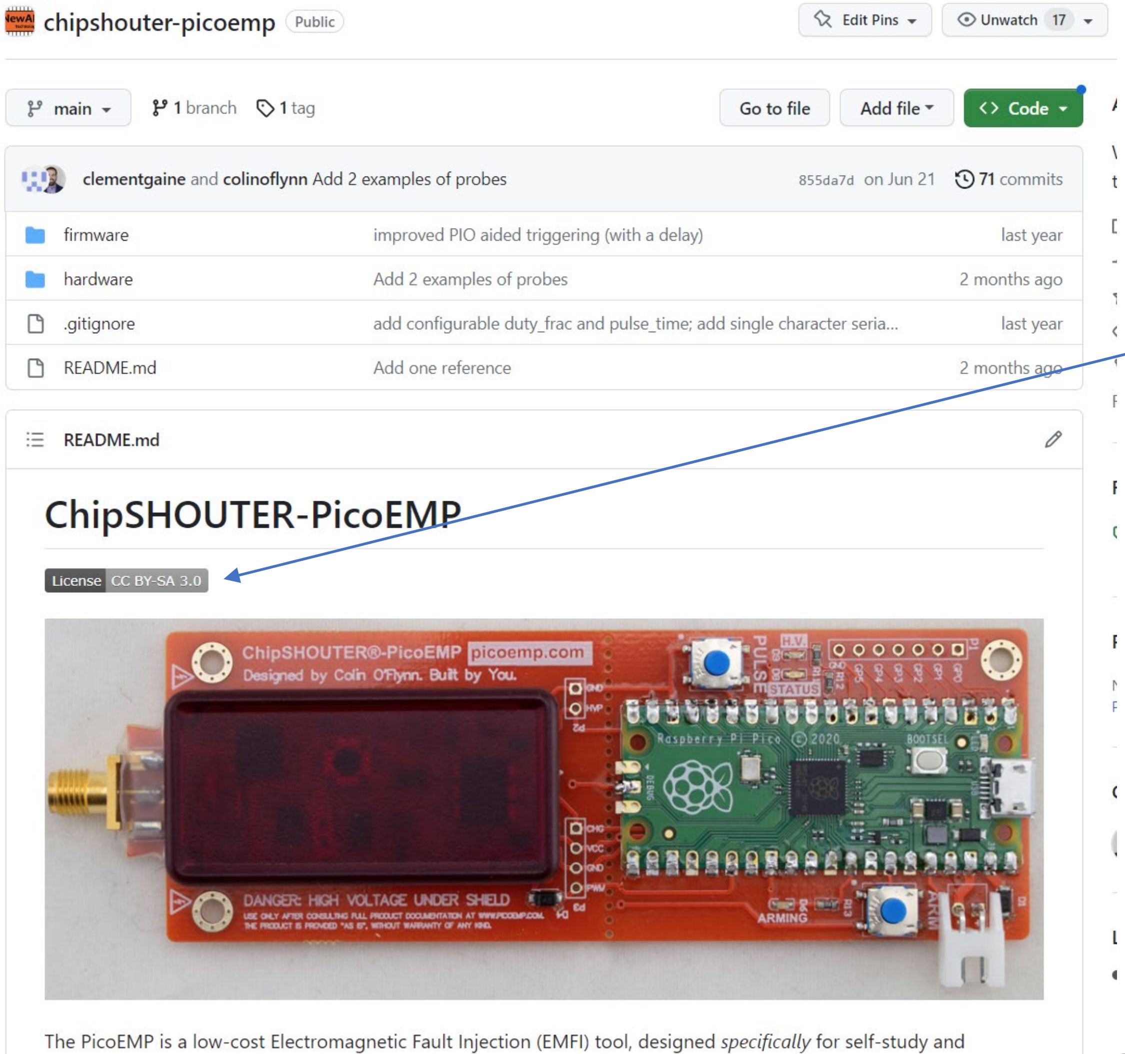
```
0000h: C2 DA 13 00 3F 01 B0 F6 C2 F8 13 00 3F 00 B0 FE  ÂÚ..?.°öÂø..?.°þ
0010h: C2 C6 12 00 3F 00 6E 08 C2 64 11 00 3F 00 B0 FE  ÂÆ..?.n.Âd..?.°þ
0020h: C2 4E 13 00 3F 00 B0 FE C2 1A 13 00 3F 00 B0 FE  ÂN..?.°þÂ...?.°þ
0030h: C2 68 11 00 3F 00 B0 FE C2 71 11 00 3F 00 B0 FE  Âh..?.°þÂq..?.°þ
0040h: C2 7A 11 00 3F 00 B0 FE D2 BC 12 00 3F 90 01 6B  Âz..?.°þÒ¼..?..k
0050h: 09 D2 BE 12 00 3F 90 01 63 13 F2 A2 12 00 00 03  .Ò¾..?..c.ò¢....
0060h: F2 20 13 00 00 01 F2 F0 10 00 00 08 0E C2 5A 12  ò ....òð.....ÂZ.
0070h: 00 3F 01 6E 13 F2 A2 12 00 00 03 F2 20 13 00 00  .?.n.ò¢....ò ...
0080h: 02 F2 F0 10 00 00 08 0E C2 3A 14 00 3F 01 6E 0E  .òð.....Â:..?.n.
0090h: F2 3A 14 00 00 00 F2 5C 12 00 00 03 68 26 F2 3A  ò:....ò\....h&ò:
00A0h: 14 00 00 01 F2 5C 12 00 00 01 C2 58 12 00 3F 00  ....ò\....ÂX..?.
00B0h: 6E 06 F2 58 12 00 00 03 F2 6C 12 00 00 00 F2 6E  n.òX....òl....òn
00C0h: 12 00 00 00 F2 F0 10 00 00 01 F2 58 14 00 00 00  ....òð....òX....
00D0h: 0E 8F 04 23 C2 6A 11 00 3F 0A 66 08 C2 6A 11 00  ...#Âj..?.f.Âj..
00E0h: 3F 0B 6E 16 C2 6B 11 00 3F 0F 6E 0E C2 5C 12 00  ?.n.Âk..?.n.Â\..
00F0h: 3F 05 66 06 F2 5C 12 00 00 03 CB 89 D8 12 D8 09  ?.f.ò\....Ë‰Ø.Ø.
0100h: 09 00 F2 6A 11 00 32 F3 07 E8 E0 00 00 CB 89 D8  ..òj..2ó.èà..Ë‰Ø
0110h: 12 D8 09 09 00 F2 6B 11 00 32 F3 07 E8 E0 00 00  .Ø...òk..2ó.èà..
0120h: CB 89 D8 12 D8 09 09 00 F2 68 11 00 32 F3 07 E8  Ë‰Ø.Ø...òh..2ó.è
0130h: E0 00 00 CB 89 D8 12 D8 09 09 00 F2 69 11 00 32  à..Ë‰Ø.Ø...òi..2
0140h: F3 07 E8 E0 00 00 CB 89 D8 12 D8 09 09 00 F2 6C  ó.èà..Ë‰Ø.Ø...òl
0150h: 11 00 32 F3 07 E8 E0 00 00 CB 89 D8 12 D8 09 09  ..2ó.èà..Ë‰Ø.Ø..
0160h: 00 F2 6D 11 00 32 F3 07 E8 E0 00 00 CB 89 D8 12  .òm..2ó.èà..Ë‰Ø.
0170h: D8 09 09 00 F2 6E 11 00 32 F3 07 E8 E0 00 00 CB  Ø...òn..2ó.èà..Ë
0180h: 89 D8 12 D8 09 09 00 F2 6F 11 00 32 F3 07 E8 E0  ‰Ø.Ø...òo..2ó.èà
0190h: 02 00 00 CB 89 D8 12 F2 98 12 00 32 F3 07 E8 E0  ...Ë‰Ø.ò˜..2ó.èà
01A0h: 00 00 CB 89 D8 12 F2 34 14 00 32 F3 07 E8 E0 00  ..Ë‰Ø.ò4..2ó.èà.
01B0h: 00 CB 89 D8 12 D8 80 F2 FE 13 00 32 F3 07 E8 E0  .Ë‰Ø.Ø€òþ..2ó.èà
01C0h: 02 00 00 C2 64 11 00 3F 00 6E 12 F2 A6 12 00 00  ...Âd..?.n.ò¦...
01D0h: 00 F2 A7 12 00 00 00 F2 A8 12 00 00 00 CB D8 66  .ò§....ò¨....ËØf
01E0h: 04 CB D9 6E 14 F2 D6 12 00 00 00 F2 DA 12 00 00  .ËÙn.òÖ....òÚ...
01F0h: 00 F2 68 12 00 00 00 68 0C F2 D8 12 00 00 00 F2  .òh....h.òØ....ò
0200h: DC 12 00 00 00 F2 E8 13 00 00 00 0E F2 66 11 00  Ü....òè.....òf..
0210h: 00 00 F2 64 11 00 00 00 F2 1E 12 00 00 00 F2 96  ..òd....ò.....ò–
0220h: 12 00 00 00 F2 9C 12 00 00 00 F2 5E 12 00 00 00  ....òœ....ò^....
0230h: F2 66 12 00 02 00 00 F2 64 12 00 02 00 00 F2 8C  òf.....òd.....òŒ
0240h: 12 00 00 00 F2 18 12 00 02 00 00 F2 C0 12 00 00  ....ò......òÀ...
0250h: 00 F2 54 11 00 02 00 00 F2 38 11 00 02 00 00 F2  .òT.....ò8.....ò
0260h: 3C 14 00 00 00 0B 00 00 1D EF A3 FE EF 62 F2 9E  <........ï£þïbòž
0270h: 12 00 00 00 F2 A0 12 00 00 00 F2 A2 12 00 00 00  ....ò ....ò¢....
0280h: F2 20 13 00 00 00 C2 F8 13 00 3F 00 66 06 F2 FA  ò ....Âø..?.f.òú
0290h: 13 00 00 01 F2 40 14 00 00 00 F2 32 13 00 02 00  ....ò@....ò2....
```



EMFI POC

- R-Pi Pico implements serial protocol.
- PicoEMP triggers an electromagnetic fault injection (EMFI).
- Tested on checksum request from bootloader → successfully corrupted checksums.
- Code available in repo (linked later).

Sidenote: PicoEMP is Open Source!



clementgaine and colinoflynn Add 2 examples of probes 855da7d on Jun 21 71 commits

firmware	improved PIO aided triggering (with a delay)	last year
hardware	Add 2 examples of probes	2 months ago
.gitignore	add configurable duty_frac and pulse_time; add single character seria...	last year
README.md	Add one reference	2 months ago

README.md

ChipSHOUTER-PicoEMP

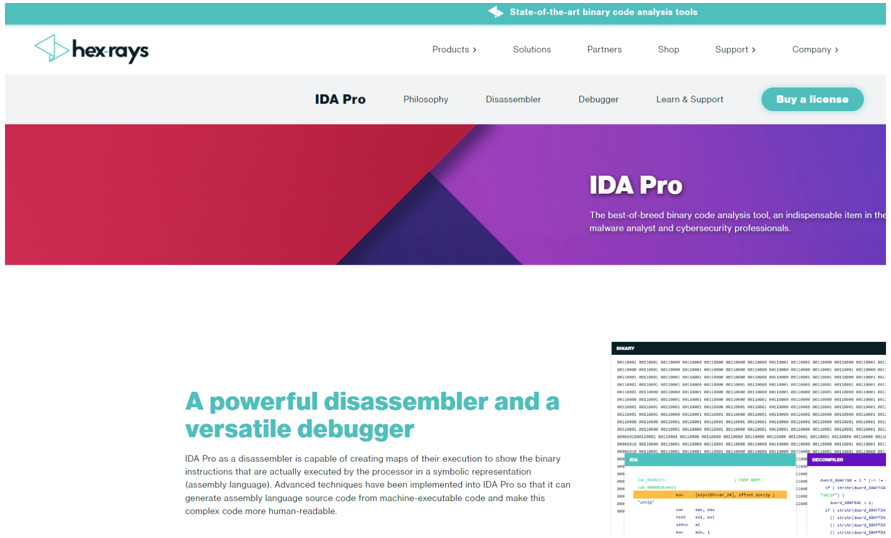
License CC BY-SA 3.0

The PicoEMP is a low-cost Electromagnetic Fault Injection (EMFI) tool, designed *specifically* for self-study and

CC-BA-SA 3.0 License!

Remix (but share per CC-BY-SA 3.0)

Reverse Engineering Tools



IDA Pro

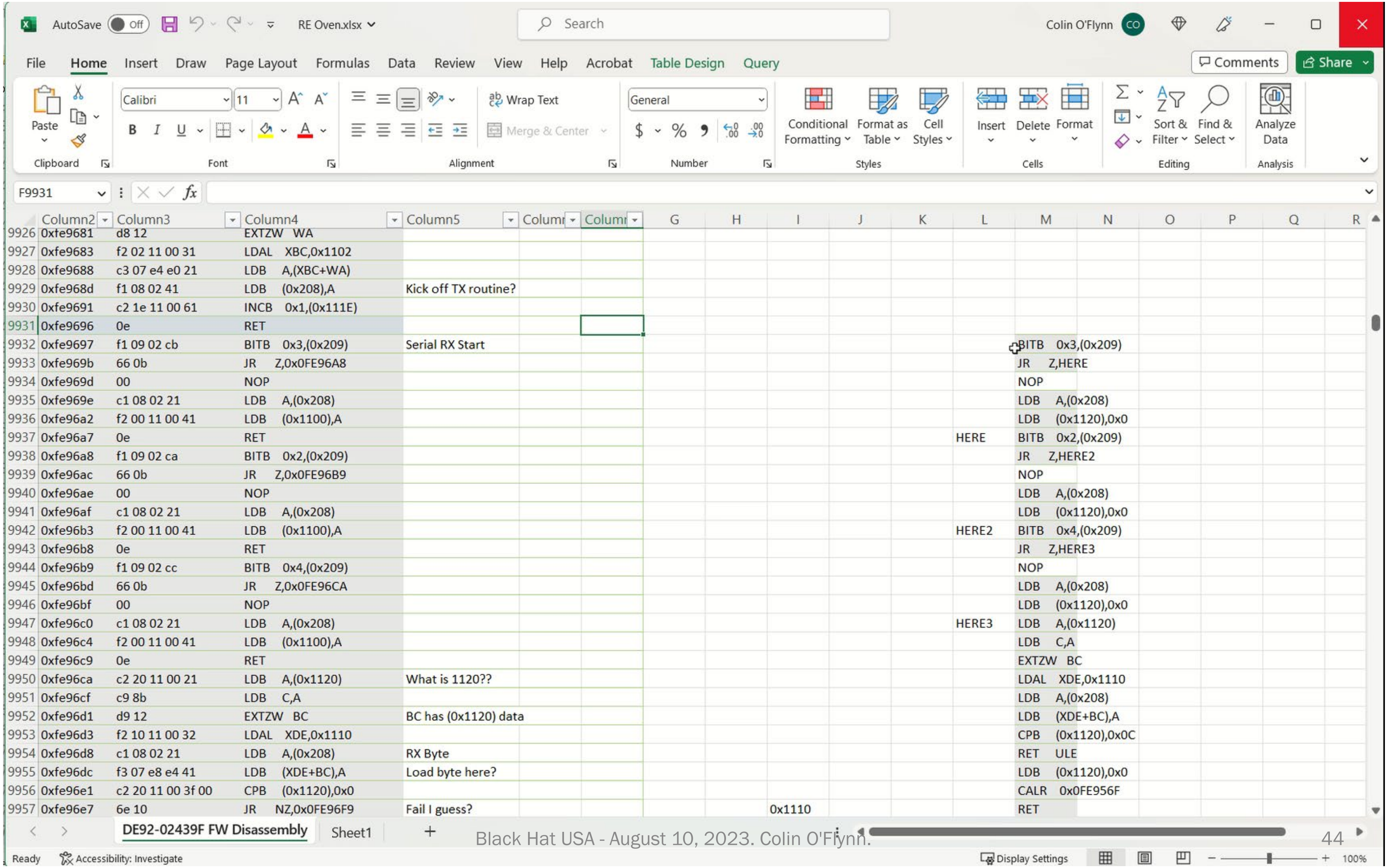
The best-of-breed binary code analysis tool, an indispensable item in the malware analyst and cybersecurity professionals.

A powerful disassembler and a versatile debugger

IDA Pro as a disassembler is capable of creating maps of their execution to show the binary instructions that are actually executed by the processor in a symbolic representation (assembly language). Advanced techniques have been implemented into IDA Pro so that it can generate assembly language source code from machine-executable code and make this complex code more human-readable.

```
loc_8048CC7:                              ; CODE XREF:
sub_8048810+A42j
          mov     [esp+28h+var_24], offset aUnzip ;
"unzip"
          xor     eax, eax
          test    esi, esi
          setnz   al
          mov     edx, 1
```

```
dword_804F780 = 2 * (v9 != 
  if ( strstr(dword_804FFD4
"UNZIP") )
    dword_804FBAC = 2;
  if ( strstr(dword_804FFD4
    || strstr(dword_804FFD4
    || strstr(dword_804FFD4
    || strstr(dword_804FFD4
```



	Column2	Column3	Column4	Column5	I	L	M
9926	0xfe9681	d8 12	EXTZW WA				
9927	0xfe9683	f2 02 11 00 31	LDAL XBC,0x1102				
9928	0xfe9688	c3 07 e4 e0 21	LDB A,(XBC+WA)				
9929	0xfe968d	f1 08 02 41	LDB (0x208),A	Kick off TX routine?			
9930	0xfe9691	c2 1e 11 00 61	INCB 0x1,(0x111E)				
9931	0xfe9696	0e	RET				
9932	0xfe9697	f1 09 02 cb	BITB 0x3,(0x209)	Serial RX Start			BITB 0x3,(0x209)
9933	0xfe969b	66 0b	JR Z,0x0FE96A8				JR Z,HERE
9934	0xfe969d	00	NOP				NOP
9935	0xfe969e	c1 08 02 21	LDB A,(0x208)				LDB A,(0x208)
9936	0xfe96a2	f2 00 11 00 41	LDB (0x1100),A				LDB (0x1120),0x0
9937	0xfe96a7	0e	RET			HERE	BITB 0x2,(0x209)
9938	0xfe96a8	f1 09 02 ca	BITB 0x2,(0x209)				JR Z,HERE2
9939	0xfe96ac	66 0b	JR Z,0x0FE96B9				NOP
9940	0xfe96ae	00	NOP				LDB A,(0x208)
9941	0xfe96af	c1 08 02 21	LDB A,(0x208)				LDB (0x1120),0x0
9942	0xfe96b3	f2 00 11 00 41	LDB (0x1100),A			HERE2	BITB 0x4,(0x209)
9943	0xfe96b8	0e	RET				JR Z,HERE3
9944	0xfe96b9	f1 09 02 cc	BITB 0x4,(0x209)				NOP
9945	0xfe96bd	66 0b	JR Z,0x0FE96CA				LDB A,(0x208)
9946	0xfe96bf	00	NOP				LDB (0x1120),0x0
9947	0xfe96c0	c1 08 02 21	LDB A,(0x208)			HERE3	LDB A,(0x1120)
9948	0xfe96c4	f2 00 11 00 41	LDB (0x1100),A				LDB C,A
9949	0xfe96c9	0e	RET				EXTZW BC
9950	0xfe96ca	c2 20 11 00 21	LDB A,(0x1120)	What is 1120??			LDAL XDE,0x1110
9951	0xfe96cf	c9 8b	LDB C,A				LDB A,(0x208)
9952	0xfe96d1	d9 12	EXTZW BC	BC has (0x1120) data			LDB (XDE+BC),A
9953	0xfe96d3	f2 10 11 00 32	LDAL XDE,0x1110				CPB (0x1120),0x0C
9954	0xfe96d8	c1 08 02 21	LDB A,(0x208)	RX Byte			RET ULE
9955	0xfe96dc	f3 07 e8 e4 41	LDB (XDE+BC),A	Load byte here?			LDB (0x1120),0x0
9956	0xfe96e1	c2 20 11 00 3f 00	CPB (0x1120),0x0				CALR 0x0FE956F
9957	0xfe96e7	6e 10	JR NZ,0x0FE96F9	Fail I guess?	0x1110		RET

Serial Monitor Built-In!?

- Not documented anywhere I could find (service docs).
- Could be useful for repair technicians!
    - Seems to only show status of various flags however, doesn’t seem to take any input.
- We could patch it to make a simple memory-dump monitor.

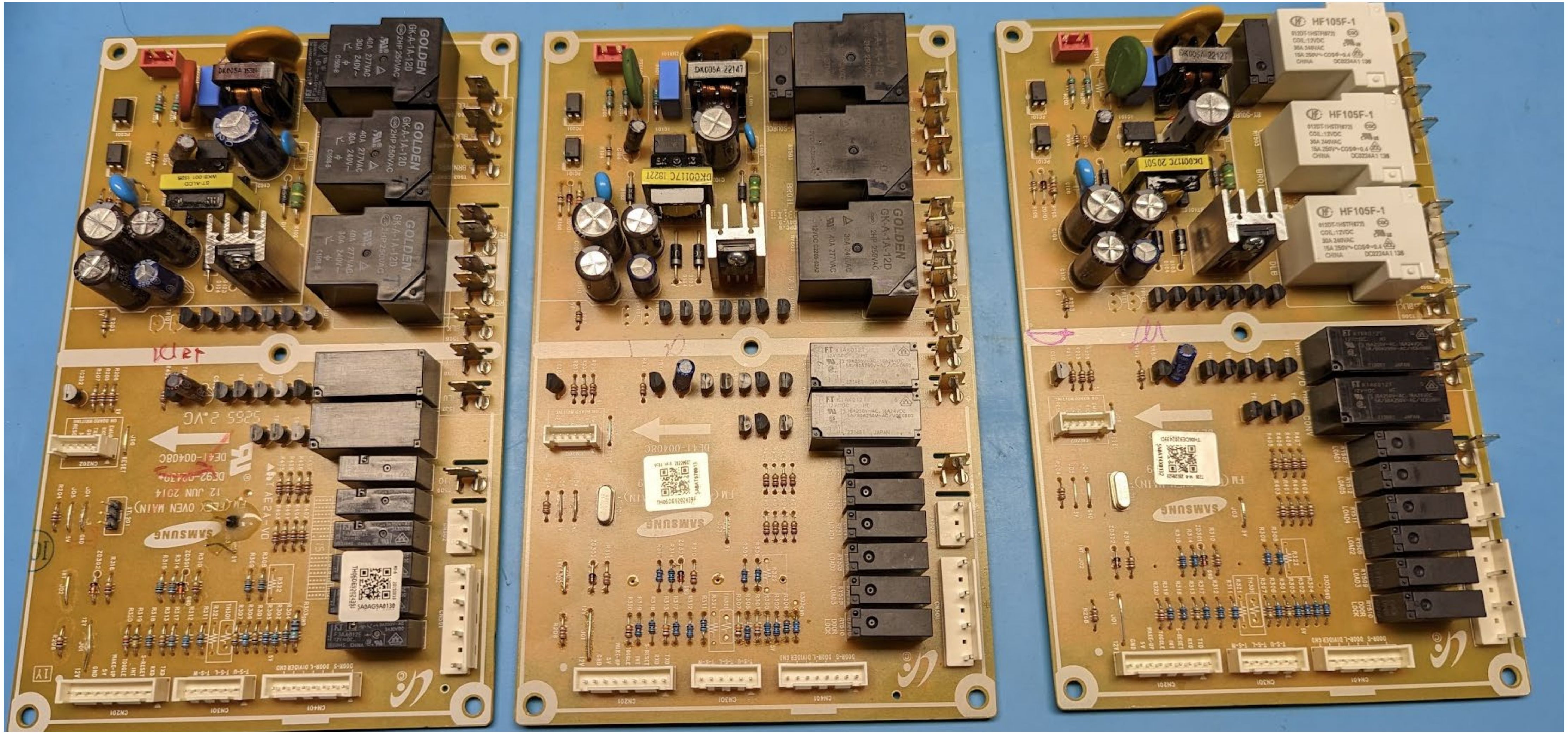
Bake
Roast
Bake
Broil
Pizza
Casserole
Chicken Nuggets
Warm
On/Off
E-84
Upper
2 3 4 5
7 8 9 0

DE 02439F

OK, Just F

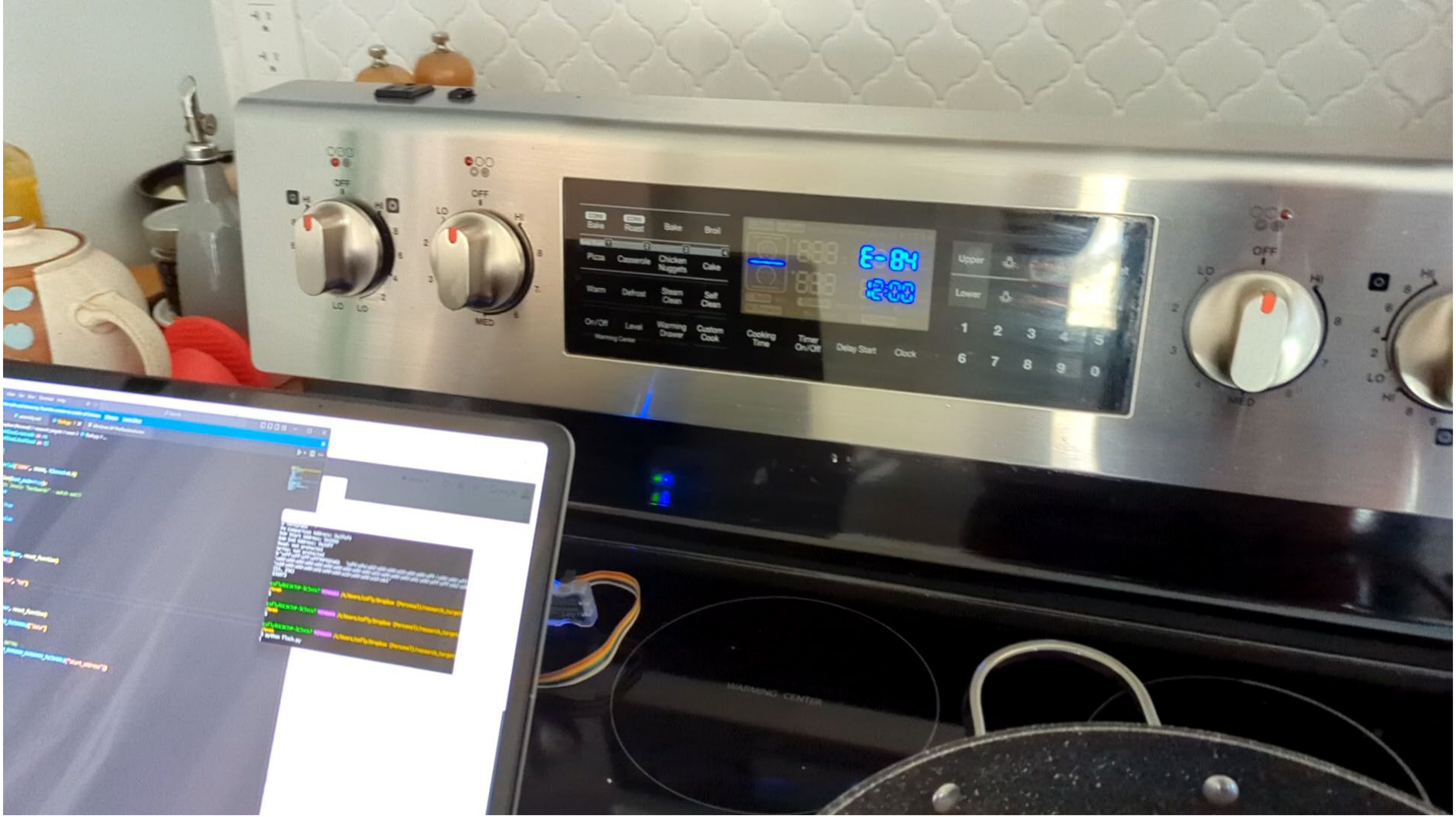
```
In [11]:  resp = rcp.cmd_read

In [12]:  resp

Out[12]:  'ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ                                        ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
          ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ                                        ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
          ÿÿÿÿÿÿÿÿÿ'


In [7]:  bl = tl
         #bl.cmd
         #bl.cmd
         bl.cmd_

         Read: r
         Write:
```

$$$ → Samsung Parts Department



Sidenote on Glitch Reliability

- Hitting too *early* seems more likely to trigger erase.
- my code tends to sweep early->late.
- Can increase reliability on specific targets (oven control board), I didn’t do that as thought it was just bad luck the 1st time…


OFF
HI
LO
MED
Conv Bake
Conv Roast
Bake
Broil
Pizza
Casserole
Chicken Nuggets
Cake
Warm
Defrost
Steam Clean
Self Clean
On/Off
Level
Warming Drawer
Custom Cook
Warming Center
E-84
12:00
Cooking Time
Timer On/Off
Delay Start
Clock
Upper
Lower
1 2 3 4 5
6 7 8 9 0
WARMING CENTER

Have there been Firmware Fixes?

MY OVEN (REVISION D FIRMWARE)

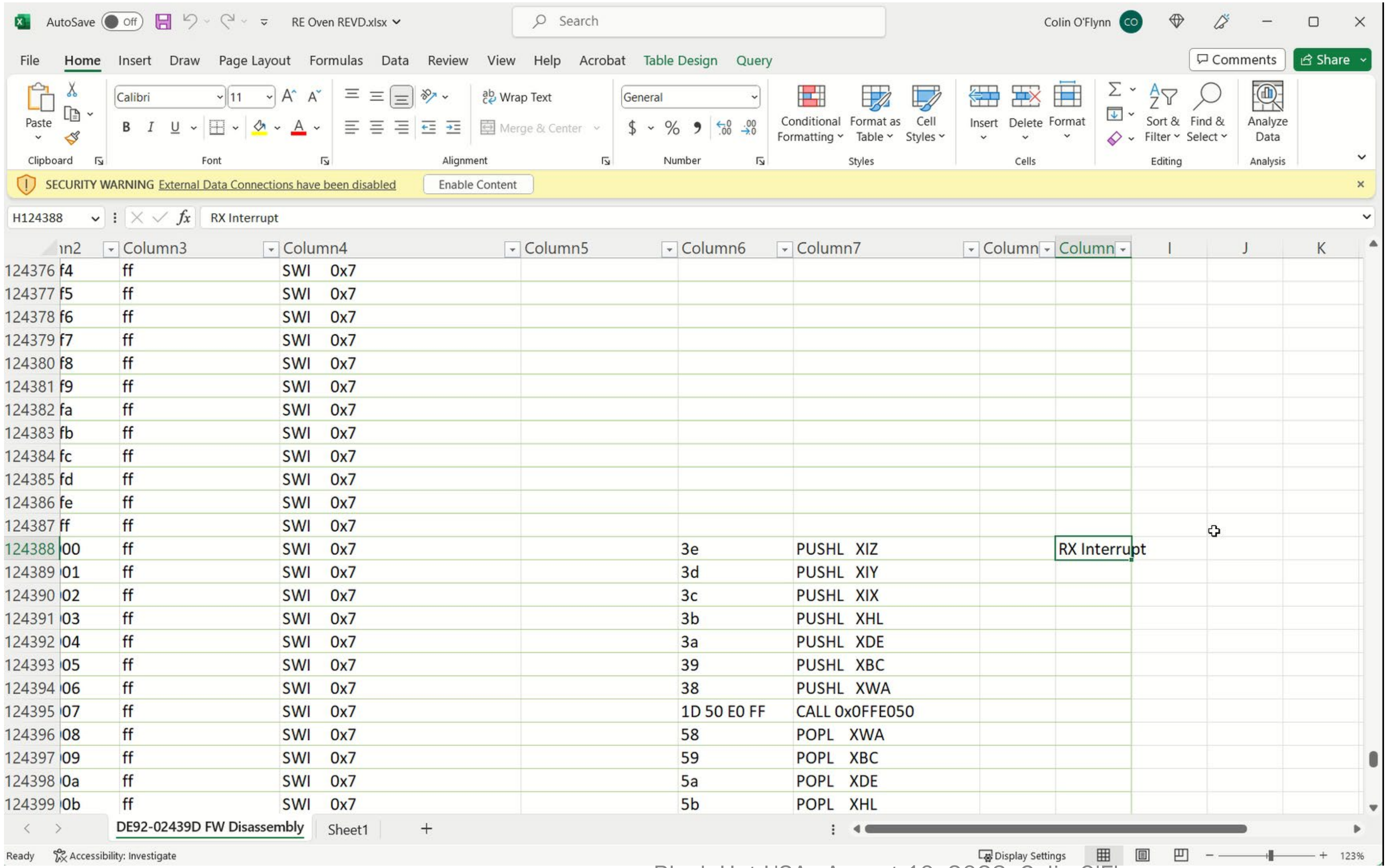
```
$ python print_status.py
b'TMP91FW60   '
PW Comparison Address: 0x2fef4
RAM Start Address: 0x1000
RAM End Address: 0x2dff
Read: protected
Write: protected
29171
```

NEW BOARD (REVISION D)

```
$ python print_status.py
b'TMP91FW60   '
PW Comparison Address: 0x2fef4
RAM Start Address: 0x1000
RAM End Address: 0x2dff
Read: not protected
Write: not protected
29238
```

Checksums Differ!

..Add the Serial Monitor

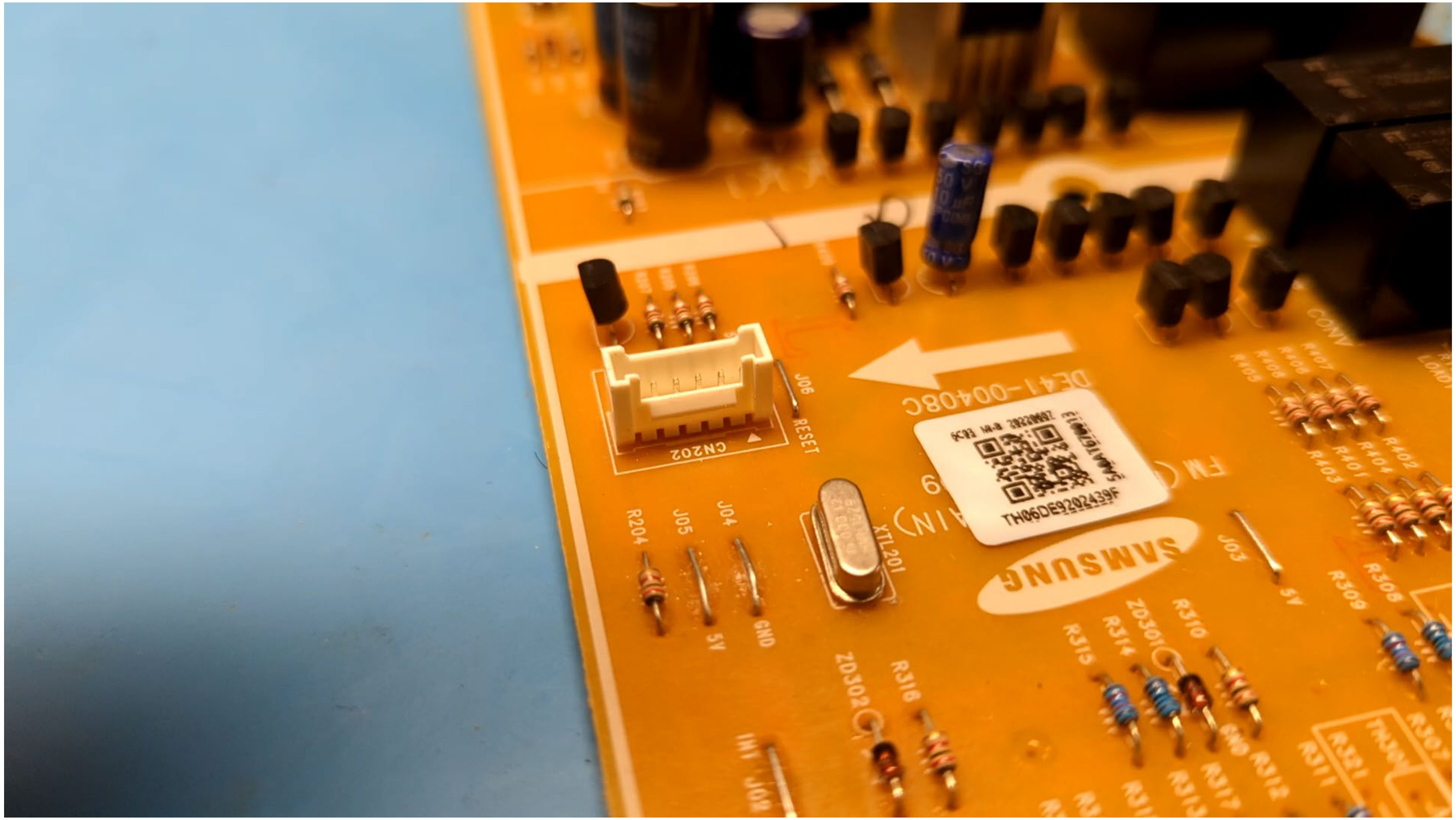


H124388 | RX Interrupt

	ın2	Column3	Column4	Column5	Column6	Column7	Column	Column	I	J	K
124376	f4	ff	SWI 0x7								
124377	f5	ff	SWI 0x7								
124378	f6	ff	SWI 0x7								
124379	f7	ff	SWI 0x7								
124380	f8	ff	SWI 0x7								
124381	f9	ff	SWI 0x7								
124382	fa	ff	SWI 0x7								
124383	fb	ff	SWI 0x7								
124384	fc	ff	SWI 0x7								
124385	fd	ff	SWI 0x7								
124386	fe	ff	SWI 0x7								
124387	ff	ff	SWI 0x7								
124388	00	ff	SWI 0x7		3e	PUSHL XIZ		RX Interrupt			
124389	01	ff	SWI 0x7		3d	PUSHL XIY					
124390	02	ff	SWI 0x7		3c	PUSHL XIX					
124391	03	ff	SWI 0x7		3b	PUSHL XHL					
124392	04	ff	SWI 0x7		3a	PUSHL XDE					
124393	05	ff	SWI 0x7		39	PUSHL XBC					
124394	06	ff	SWI 0x7		38	PUSHL XWA					
124395	07	ff	SWI 0x7		1D 50 E0 FF	CALL 0x0FFE050					
124396	08	ff	SWI 0x7		58	POPL XWA					
124397	09	ff	SWI 0x7		59	POPL XBC					
124398	0a	ff	SWI 0x7		5a	POPL XDE					
124399	0b	ff	SWI 0x7		5b	POPL XHL					

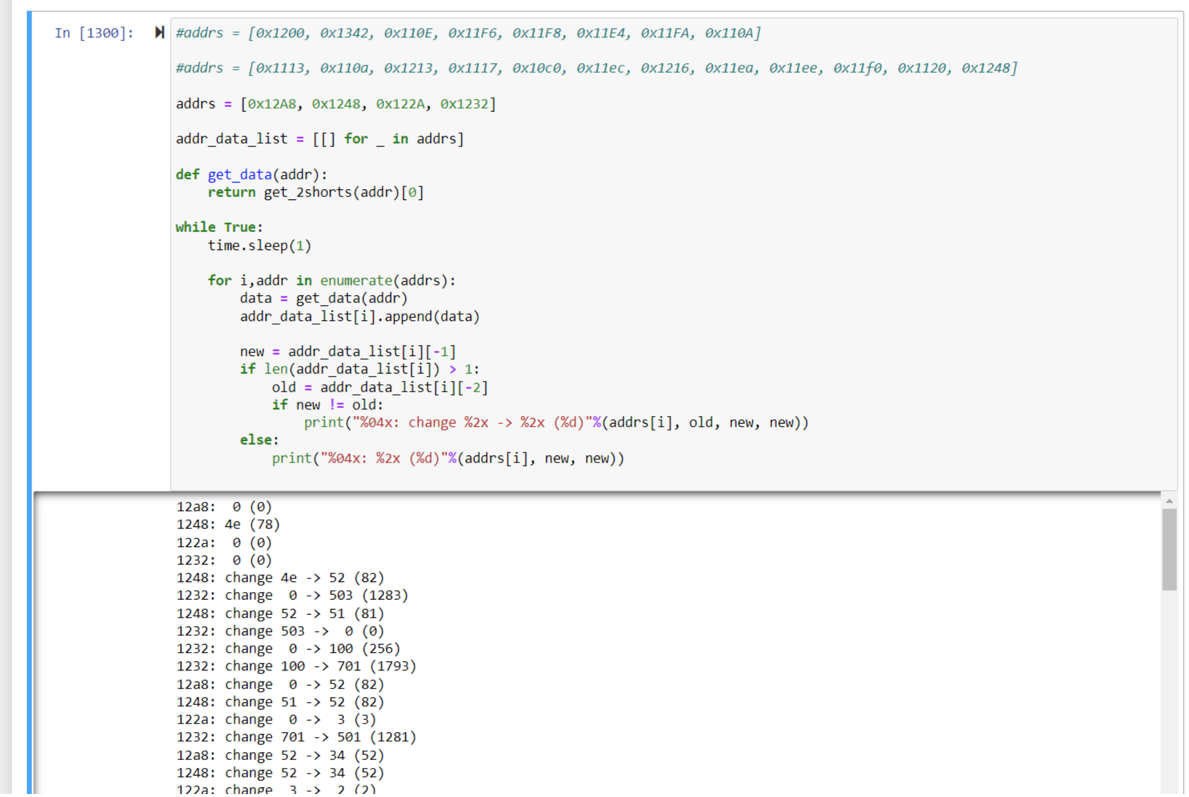
Slight risk of overwriting something else important….

“Production” Serial Interface



R.E. Data Storage Locations

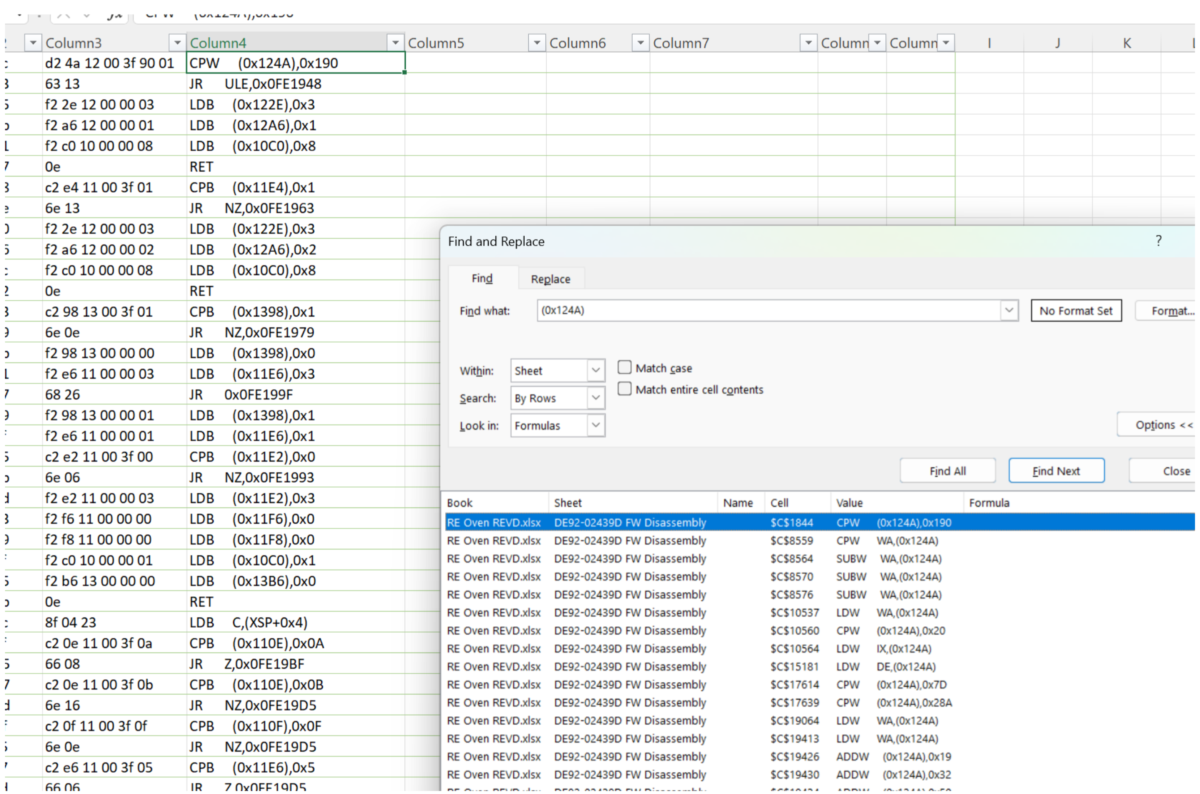
Can find data blocks from R.E. work. Then find changing data as you do different things (start/stop oven, change temp, etc).



In [1300]:

```
#addrs = [0x1200, 0x1342, 0x110E, 0x11F6, 0x11F8, 0x11E4, 0x11FA, 0x110A]

#addrs = [0x1113, 0x110a, 0x1213, 0x1117, 0x10c0, 0x11ec, 0x1216, 0x11ea, 0x11ee, 0x11f0, 0x1120, 0x1248]

addrs = [0x12A8, 0x1248, 0x122A, 0x1232]

addr_data_list = [[] for _ in addrs]

def get_data(addr):
    return get_2shorts(addr)[0]

while True:
    time.sleep(1)

    for i,addr in enumerate(addrs):
        data = get_data(addr)
        addr_data_list[i].append(data)

        new = addr_data_list[i][-1]
        if len(addr_data_list[i]) > 1:
            old = addr_data_list[i][-2]
            if new != old:
                print("%04x: change %2x -> %2x (%d)"%(addrs[i], old, new, new))
        else:
            print("%04x: %2x (%d)"%(addrs[i], new, new))
```

```
12a8:  0 (0)
1248: 4e (78)
122a:  0 (0)
1232:  0 (0)
1248: change 4e -> 52 (82)
1232: change  0 -> 503 (1283)
1248: change 52 -> 51 (81)
1232: change 503 ->  0 (0)
1232: change  0 -> 100 (256)
1232: change 100 -> 701 (1793)
12a8: change  0 -> 52 (82)
1248: change 51 -> 52 (82)
122a: change  0 ->  3 (3)
1232: change 701 -> 501 (1281)
12a8: change 52 -> 34 (52)
1248: change 52 -> 34 (52)
122a: change  3 ->  2 (2)
```



CPW (0x124A),0x190

Column3	Column4	Column5	Column6	Column7	Column	Column	I	J	K
d2 4a 12 00 3f 90 01	CPW (0x124A),0x190								
63 13	JR ULE,0x0FE1948								
f2 2e 12 00 00 03	LDB (0x122E),0x3								
f2 a6 12 00 00 01	LDB (0x12A6),0x1								
f2 c0 10 00 00 08	LDB (0x10C0),0x8								
0e	RET								
c2 e4 11 00 3f 01	CPB (0x11E4),0x1								
6e 13	JR NZ,0x0FE1963								
f2 2e 12 00 00 03	LDB (0x122E),0x3								
f2 a6 12 00 00 02	LDB (0x12A6),0x2								
f2 c0 10 00 00 08	LDB (0x10C0),0x8								
0e	RET								
c2 98 13 00 3f 01	CPB (0x1398),0x1								
6e 0e	JR NZ,0x0FE1979								
f2 98 13 00 00 00	LDB (0x1398),0x0								
f2 e6 11 00 00 03	LDB (0x11E6),0x3								
68 26	JR 0x0FE199F								
f2 98 13 00 00 01	LDB (0x1398),0x1								
f2 e6 11 00 00 01	LDB (0x11E6),0x1								
c2 e2 11 00 3f 00	CPB (0x11E2),0x0								
6e 06	JR NZ,0x0FE1993								
f2 e2 11 00 00 03	LDB (0x11E2),0x3								
f2 f6 11 00 00 00	LDB (0x11F6),0x0								
f2 f8 11 00 00 00	LDB (0x11F8),0x0								
f2 c0 10 00 00 01	LDB (0x10C0),0x1								
f2 b6 13 00 00 00	LDB (0x13B6),0x0								
0e	RET								
8f 04 23	LDB C,(XSP+0x4)								
c2 0e 11 00 3f 0a	CPB (0x110E),0x0A								
66 08	JR Z,0x0FE19BF								
c2 0e 11 00 3f 0b	CPB (0x110E),0x0B								
6e 16	JR NZ,0x0FE19D5								
c2 0f 11 00 3f 0f	CPB (0x110F),0x0F								
6e 0e	JR NZ,0x0FE19D5								
c2 e6 11 00 3f 05	CPB (0x11E6),0x5								
66 06	JR Z,0x0FE19D5								

Find and Replace

Find what: (0x124A)

Within: Sheet

Search: By Rows

Look in: Formulas

☐ Match case

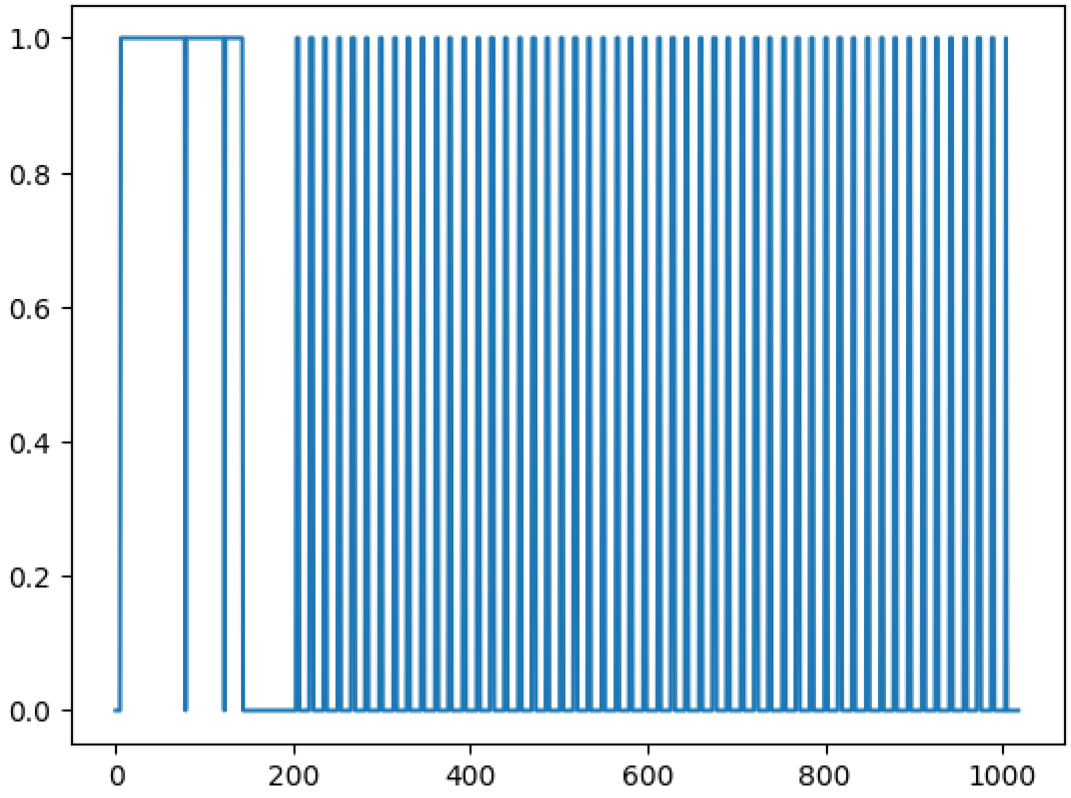
☐ Match entire cell contents

Book	Sheet	Name	Cell	Value	Formula
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C1844	CPW (0x124A),0x190	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C8559	CPW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C8564	SUBW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C8570	SUBW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C8576	SUBW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C10537	LDW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C10560	CPW (0x124A),0x20	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C10564	LDW IX,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C15181	LDW DE,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C17614	CPW (0x124A),0x7D	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C17639	CPW (0x124A),0x28A	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C19064	LDW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C19413	LDW WA,(0x124A)	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C19426	ADDW (0x124A),0x19	
RE Oven REVD.xlsx	DE92-02439D FW Disassembly		C19430	ADDW (0x124A),0x32	

Examples of Global Variables

0x1248 = Top Temp in F

0x120a = Heater “ON” Flag



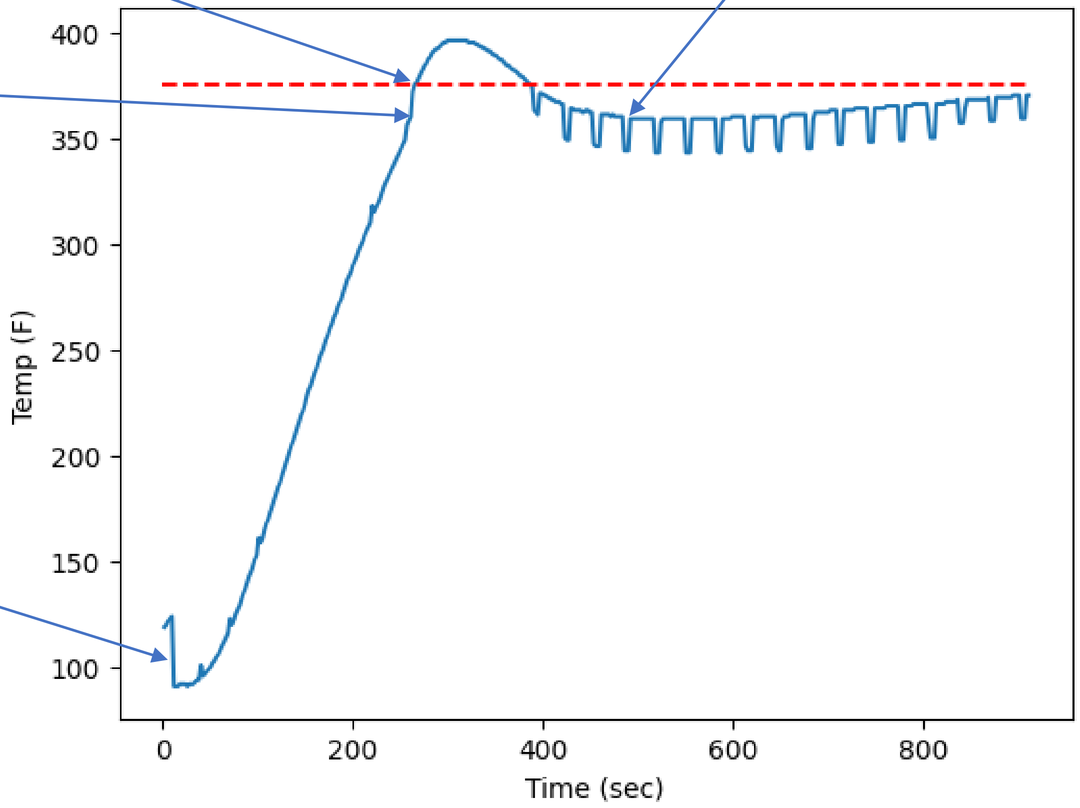
Set 375F, Cold Start, No Load

Oven starts reading real temp (which is higher)

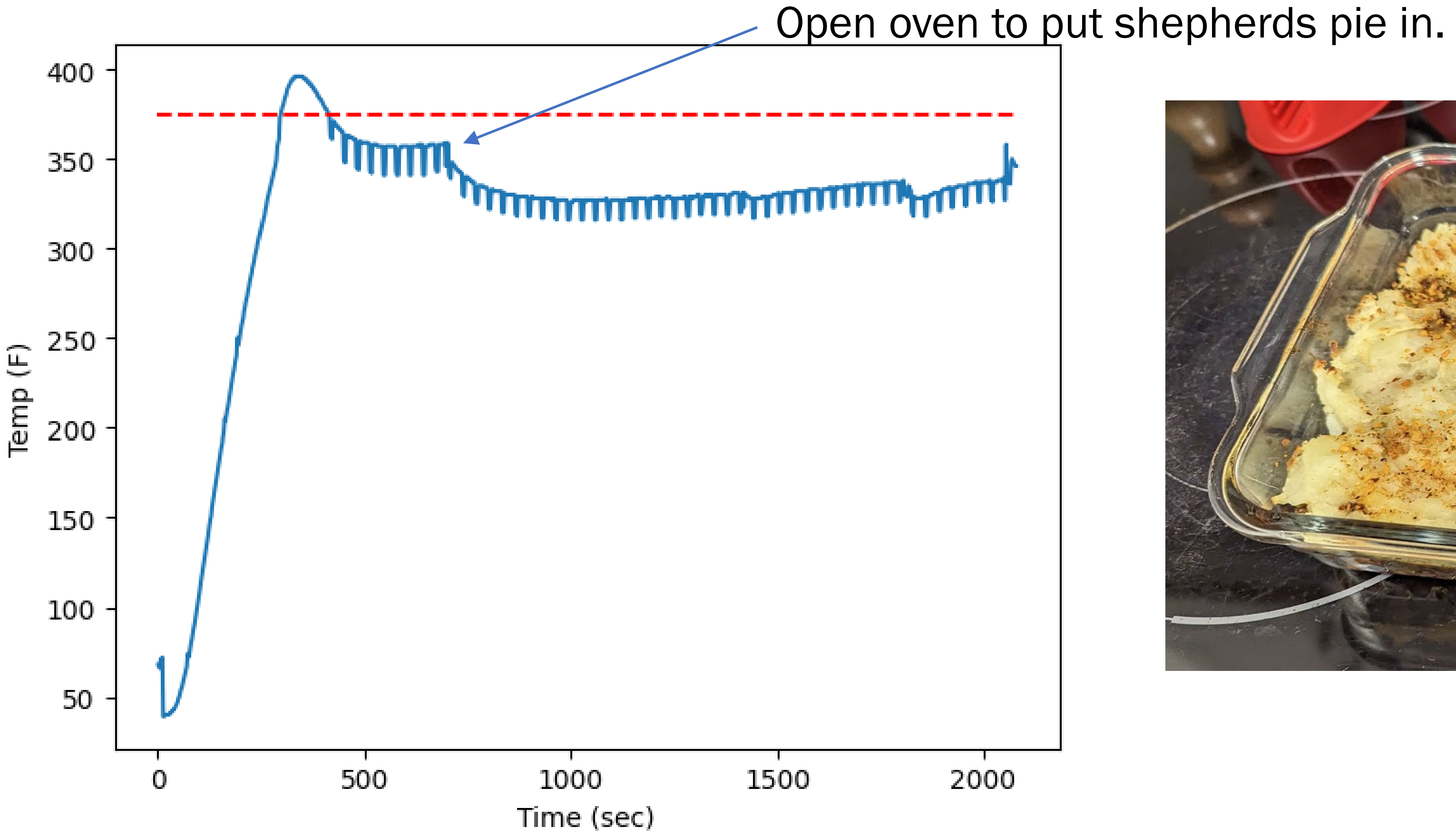
Short pulses to maintain temp (spikes down are when heater is on)

Element turns off.

When heating element turns ON, measured temp drops.

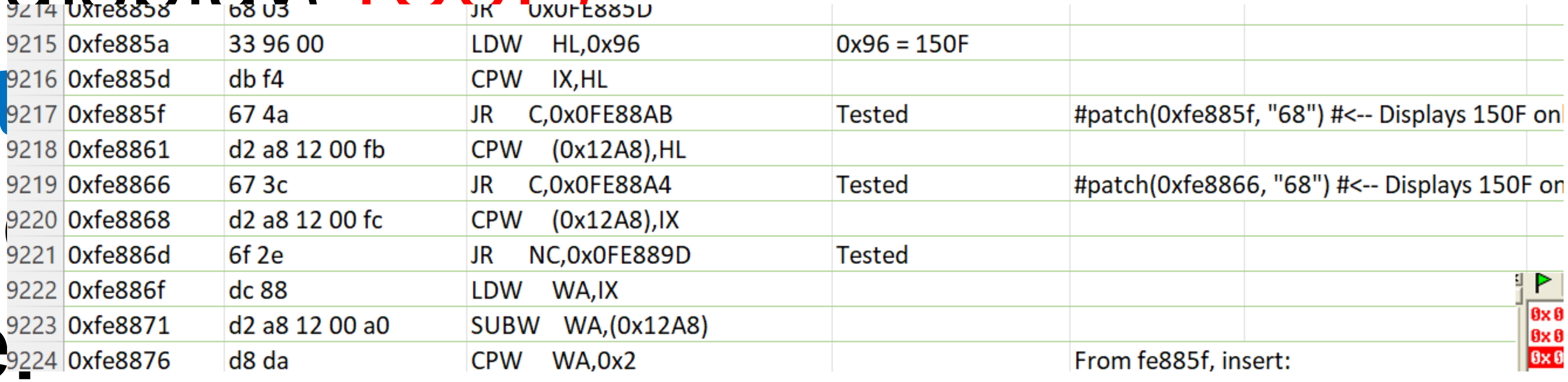


Set 375F, Cold Start, Load (Shepherds Pie)


Open oven to put shepherds pie in.

Observed Display Logic During Pre-Heat

```
if temp < 150F:
        display(150F)

elif t

else:
        display(temp)
        old_temp = temp
```

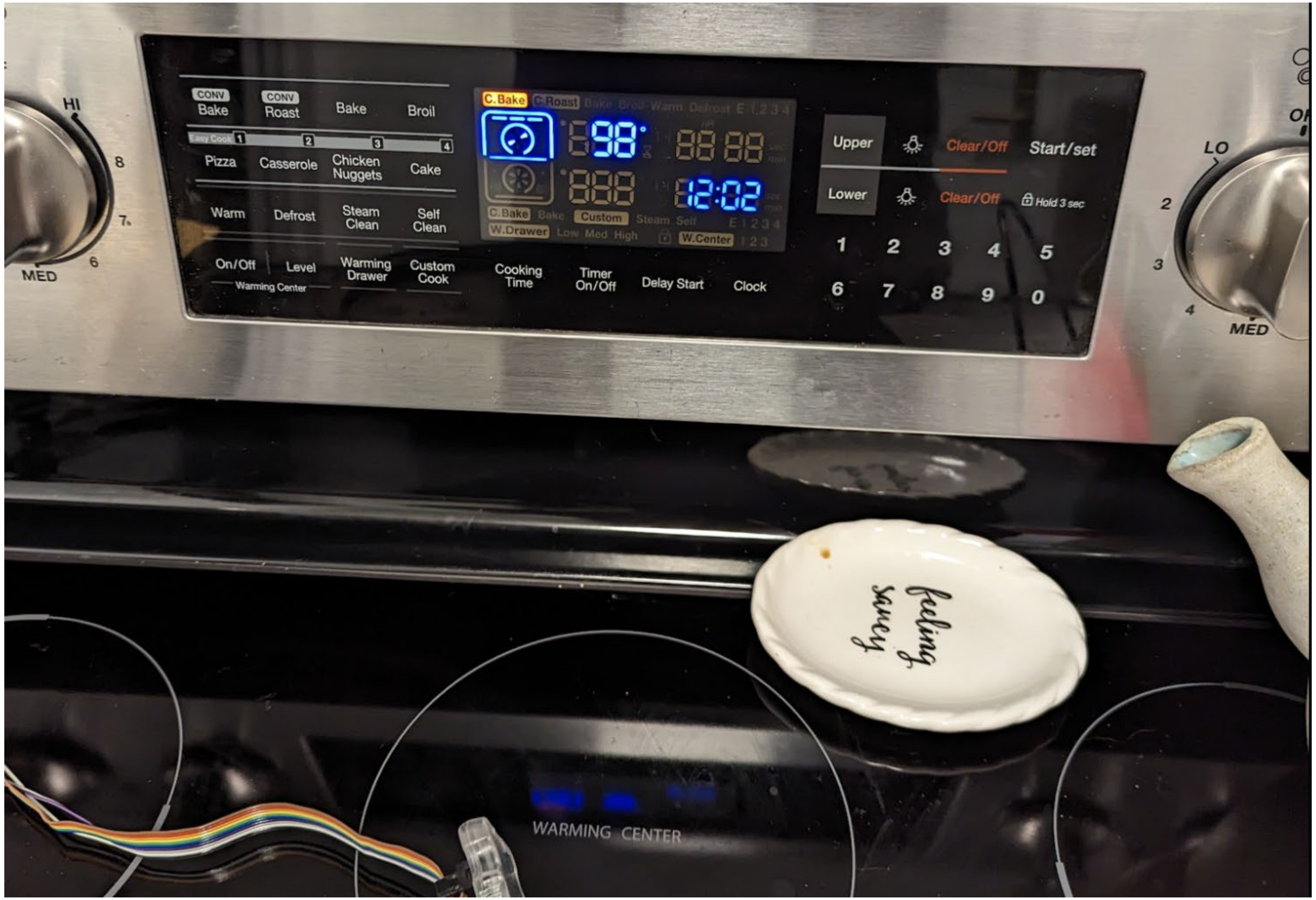


9214	0xfe8858	68 03	JR 0x0FE885D		
9215	0xfe885a	33 96 00	LDW HL,0x96	0x96 = 150F	
9216	0xfe885d	db f4	CPW IX,HL		
9217	0xfe885f	67 4a	JR C,0x0FE88AB	Tested	#patch(0xfe885f, "68") #<-- Displays 150F onl
9218	0xfe8861	d2 a8 12 00 fb	CPW (0x12A8),HL		
9219	0xfe8866	67 3c	JR C,0x0FE88A4	Tested	#patch(0xfe8866, "68") #<-- Displays 150F on
9220	0xfe8868	d2 a8 12 00 fc	CPW (0x12A8),IX		
9221	0xfe886d	6f 2e	JR NC,0x0FE889D	Tested	
9222	0xfe886f	dc 88	LDW WA,IX		
9223	0xfe8871	d2 a8 12 00 a0	SUBW WA,(0x12A8)		
9224	0xfe8876	d8 da	CPW WA,0x2		From fe885f, insert:

Observed Display Logic During Cooking

display(set_temp)

Patched Display Logic



Best Guess for Display Logic Design?

- Confusing for customers if temperature drops suddenly when heater is on.
  - Easier to lie to customers & show the max temp.

- Don’t want customers to worry about “peaking”
  - Switch to “maintain” mode once temp > set_temp, after that only show set_temp. Customer feels like they see preheat working, now they “see” oven working. Happy Customer!

Burning Cookies

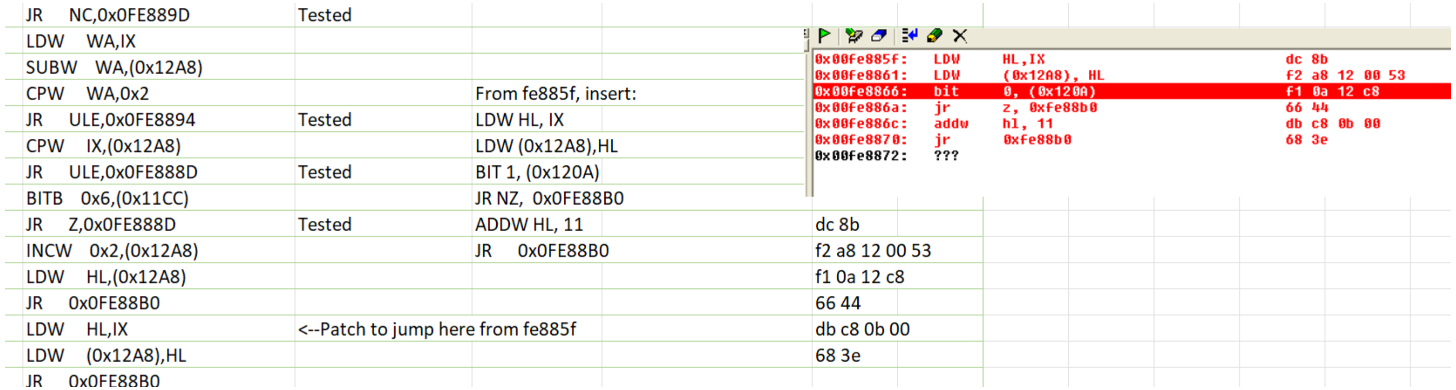
Known work-around for these ovens is to stop & restart them.

- This shows you the “true” temperature again.
- This puts them back into “pre-heat” mode where they have enough power.
- If you are lucky it now can stabilize around the right temperature.

PROBLEM: The “peak” tends to still happen -> can burn items in the oven! This was also observed in practice…

New Cooking/Display Logic (old-school thermostat)

```
if temp < setpoint:
        heater(on)
        display(temp+11)
else:
        heater(off)
        display(temp)
```

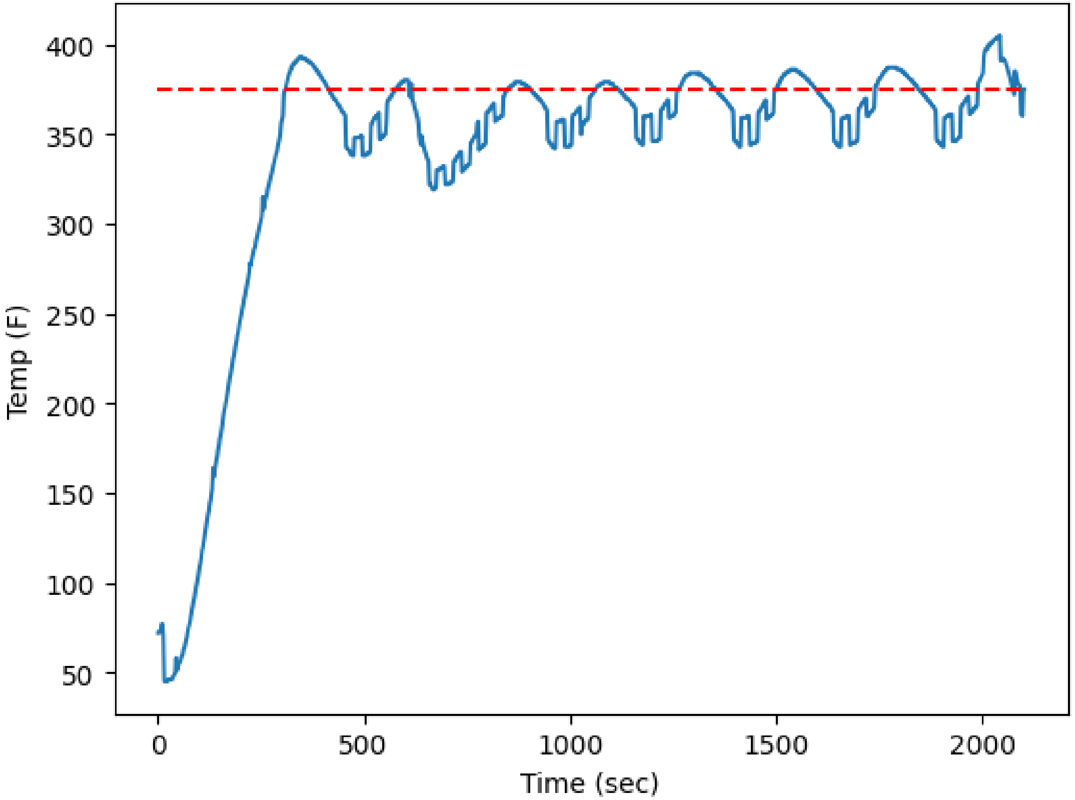


JR NC,0x0FE889D	Tested		
LDW WA,IX			
SUBW WA,(0x12A8)			
CPW WA,0x2		From fe885f, insert:	
JR ULE,0x0FE8894	Tested	LDW HL, IX	
CPW IX,(0x12A8)		LDW (0x12A8),HL	
JR ULE,0x0FE888D	Tested	BIT 1, (0x120A)	
BITB 0x6,(0x11CC)		JR NZ, 0x0FE88B0	
JR Z,0x0FE888D	Tested	ADDW HL, 11	dc 8b
INCW 0x2,(0x12A8)		JR 0x0FE88B0	f2 a8 12 00 53
LDW HL,(0x12A8)			f1 0a 12 c8
JR 0x0FE88B0			66 44
LDW HL,IX	<--Patch to jump here from fe885f		db c8 0b 00
LDW (0x12A8),HL			68 3e
JR 0x0FE88B0			

```
0x00fe885f:   LDW    HL,IX              dc 8b
0x00fe8861:   LDW    (0x12A8), HL       f2 a8 12 00 53
0x00fe8866:   bit    0, (0x120A)        f1 0a 12 c8
0x00fe886a:   jr     z, 0xfe88b0        66 44
0x00fe886c:   addw   hl, 11             db c8 0b 00
0x00fe8870:   jr     0xfe88b0           68 3e
0x00fe8872:   ???
```

Code also stops it from going into the "maintain" temperature mode, leaves it in "preheat" mode.

Set 375F, Cold Start, Load (Shepherds Pie)

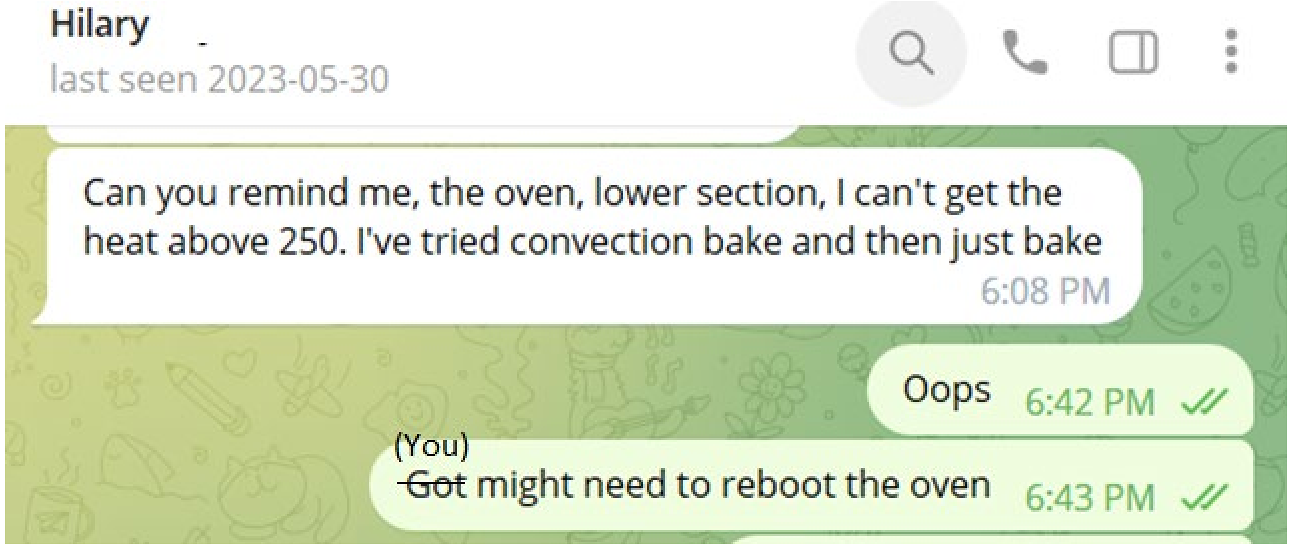


Soufflé Test

WARMING CENTER
Fast Boil

https://www.myrecipes.com/recipe/individual-chocolate-souffl-cakes

Known Bugs



Hilary

last seen 2023-05-30

Can you remind me, the oven, lower section, I can't get the heat above 250. I've tried convection bake and then just bake 6:08 PM

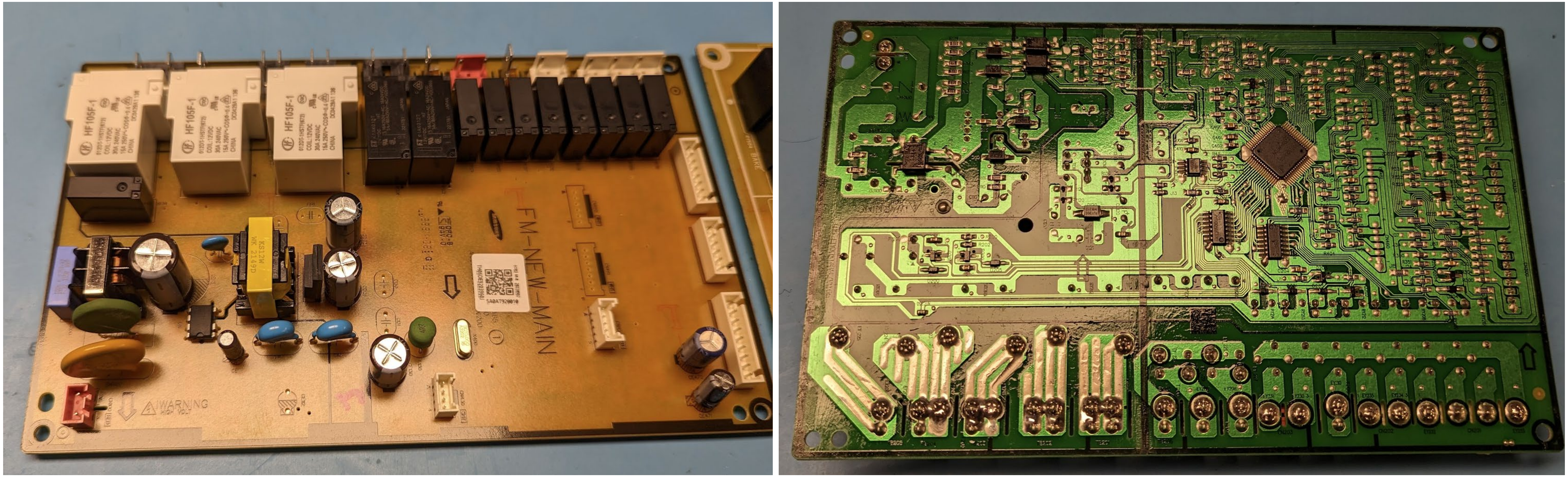
Oops 6:42 PM

(You)
~~Got~~ might need to reboot the oven 6:43 PM

With my patches: after the oven is plugged in for some length of time, seems it stops heating correctly. Need to power cycle at circuit breakers and will work again for a while.

Future Work

DE92-03960J Controller Board:

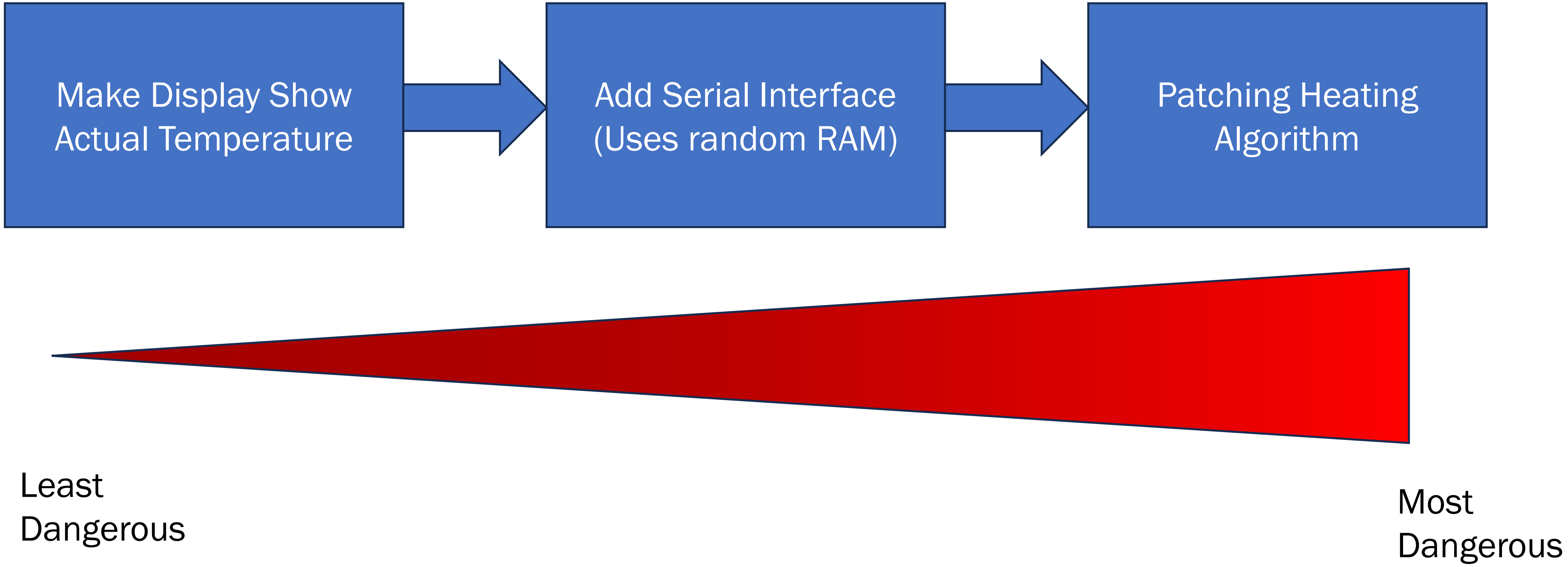


- “Newer” ovens based on R5F100LEAFB#V0 (RL78/G13)
- No protection (can read-out with debugger)
- Supported Ghidra plugin!

Playing with Your Own Oven

- Confirm it's correct version using TMP91 (not newer board)
- Need serial interface cable, if running in-place need 5V compatible + isolated due to mains input (suggest *µArt*, https://uart-adapter.com/)
- Script in repo can check status of oven (if write protection enabled).
  - If no write protection, need only known password.
  - If write protection enabled, need firmware image first OR glitch.
- Feel free to try some fixes (at your own risk)

Playing with Your Own Oven



Make Display Show Actual Temperature

Add Serial Interface (Uses random RAM)

Patching Heating Algorithm

Least Dangerous

Most Dangerous

Important Design Reminder

The range elements are knob controlled (mechanical action needed).

The heating elements IN the oven are **100% firmware controlled**.

What I learned?

- Might not be your fault having trouble with receipies & cooking time.
- Many ovens *actively lie to you* to hide their issues.
- Lots of wasted electronic waste generated from this problem (at minimum parts, at worst full ovens).
- Just reflashing boards should be a repair item (but isn't).

Questions? Details?

https://github.com/colinoflynn/samsung-ovens-deconstructed

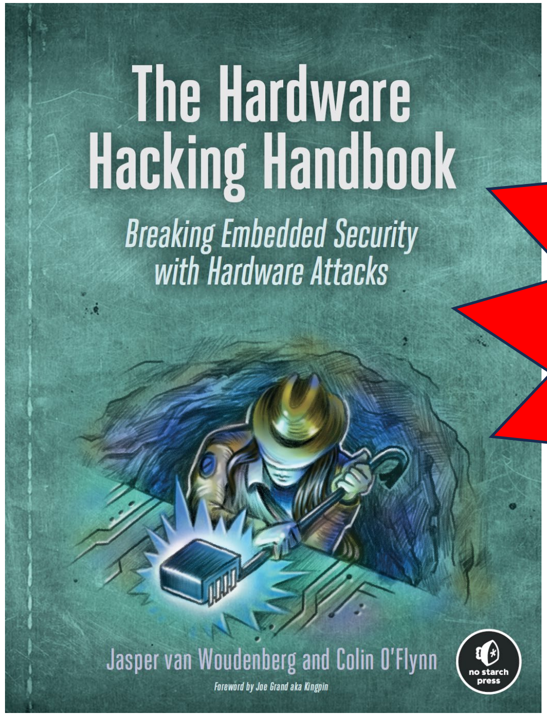
https://github.com/colinoflynn/Toshiba-TLCS-900-L-Resources

General overview at blog post on:

https://www.oflynn.com

@colinoflynn.bsky.social

colinoflynn@bluenoser.me


The Hardware Hacking Handbook
Breaking Embedded Security with Hardware Attacks
Jasper van Woudenberg and Colin O'Flynn
Foreword by Joe Grand aka Kingpin
no starch press

Book Signing! TODAY @ 11AM!