



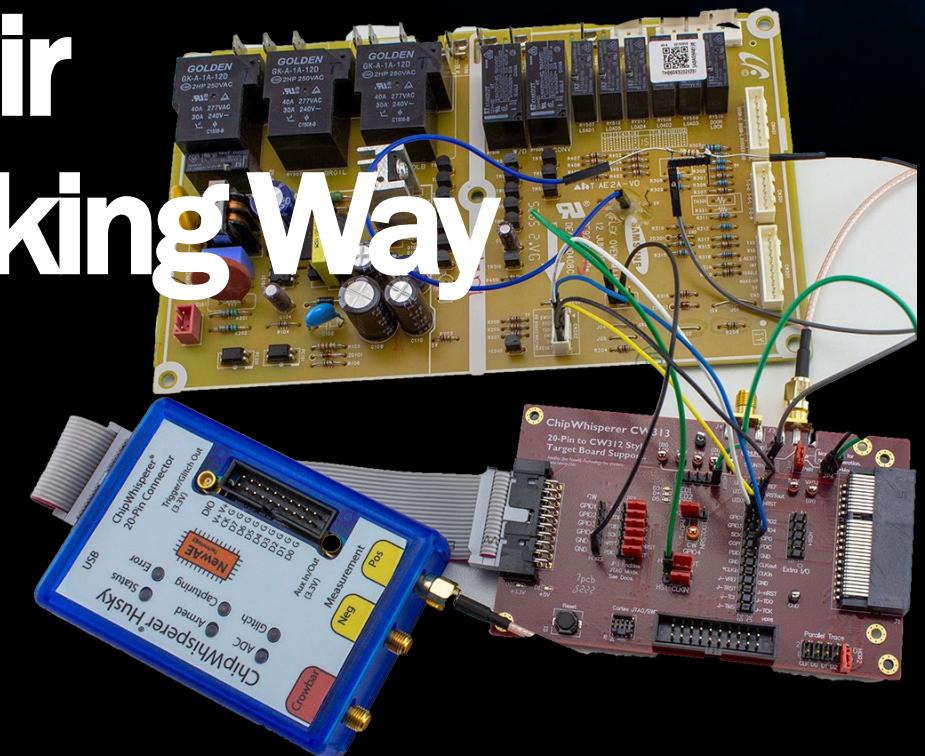
AUGUST 9-10, 2023

BRIEFINGS



Oven Repair The Hardware Hacking Way

Speaker: Colin O'Flynn

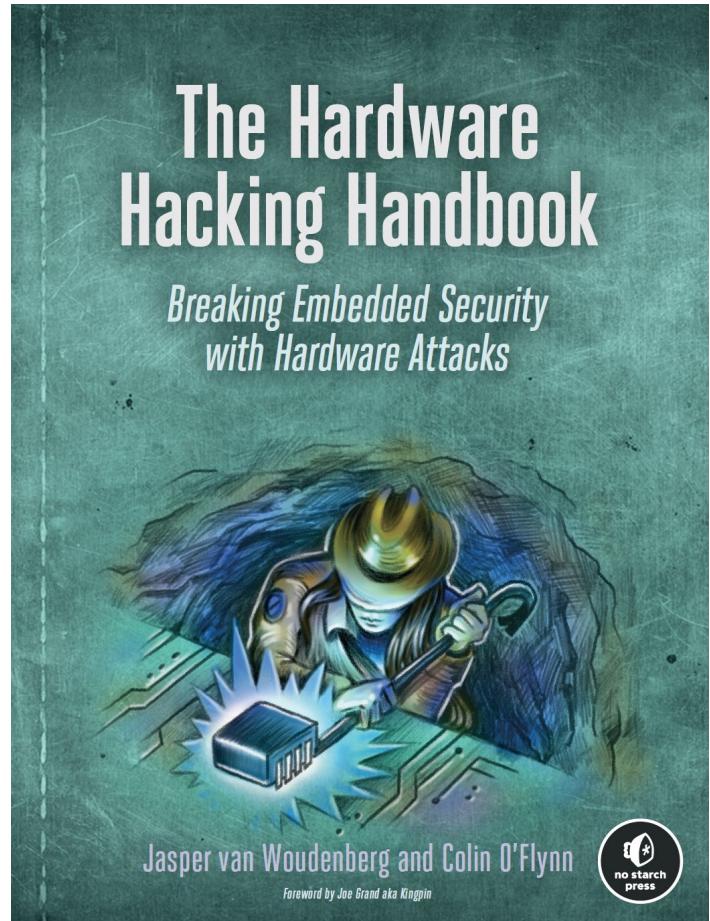




Black Hat USA - August 10, 2023. Colin O'Flynn.



About Me



- 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

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



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.

Parsons

77 comments

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

Black Hat USA - August 10, 2023. Colin O'Flynn.

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:



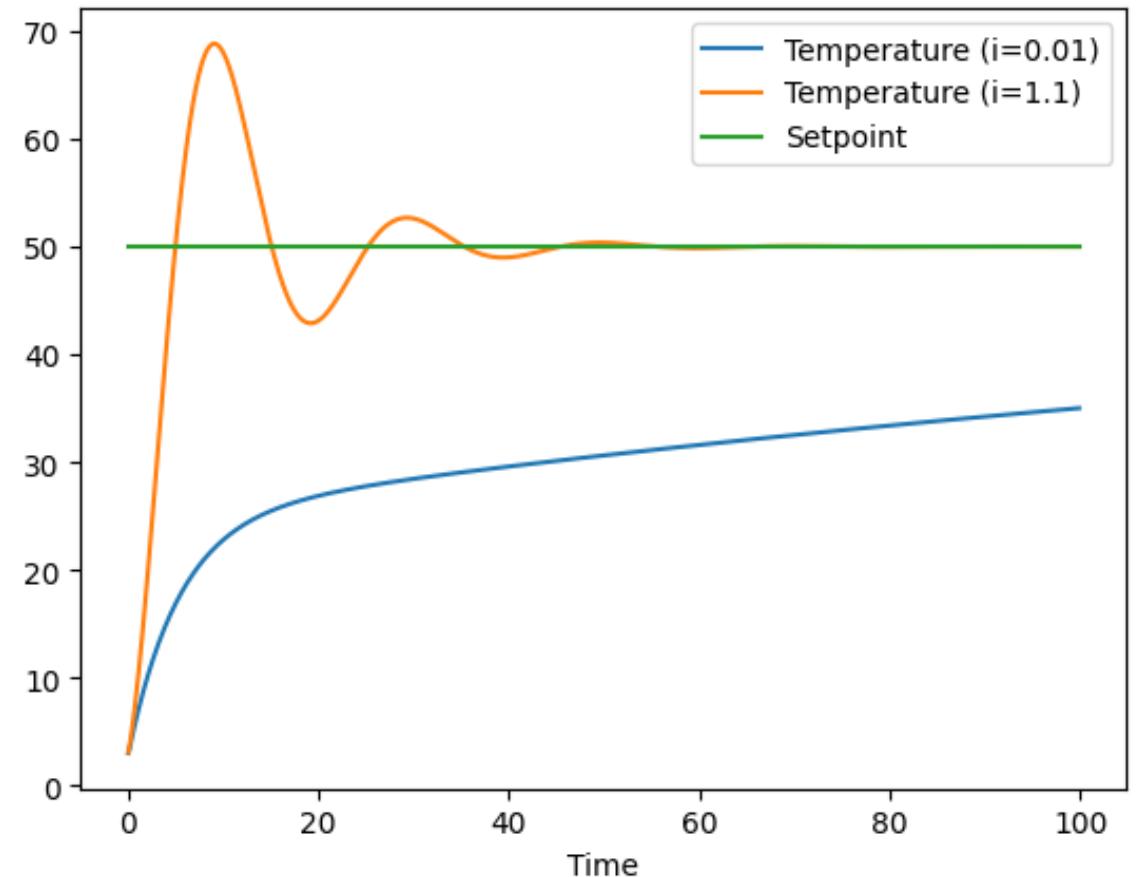
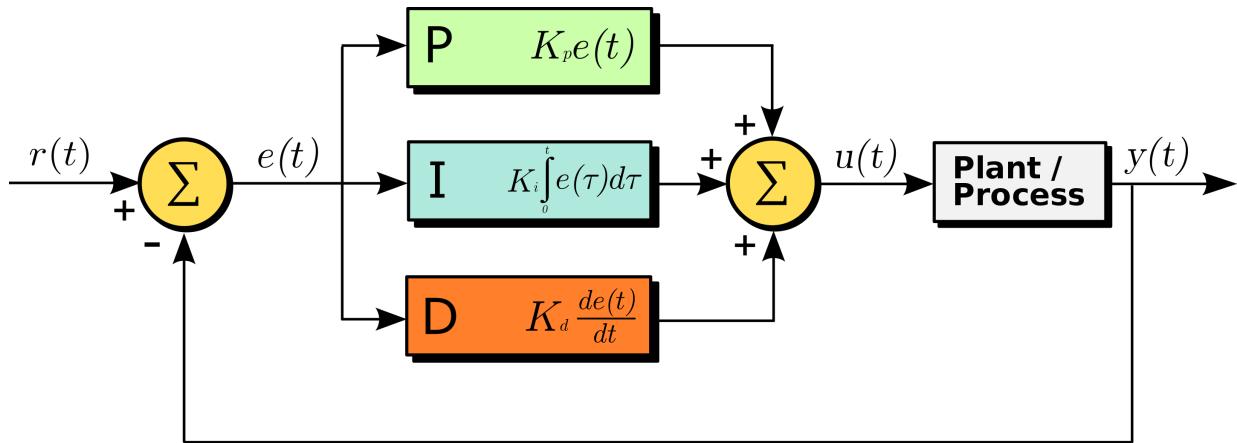
8
3
A

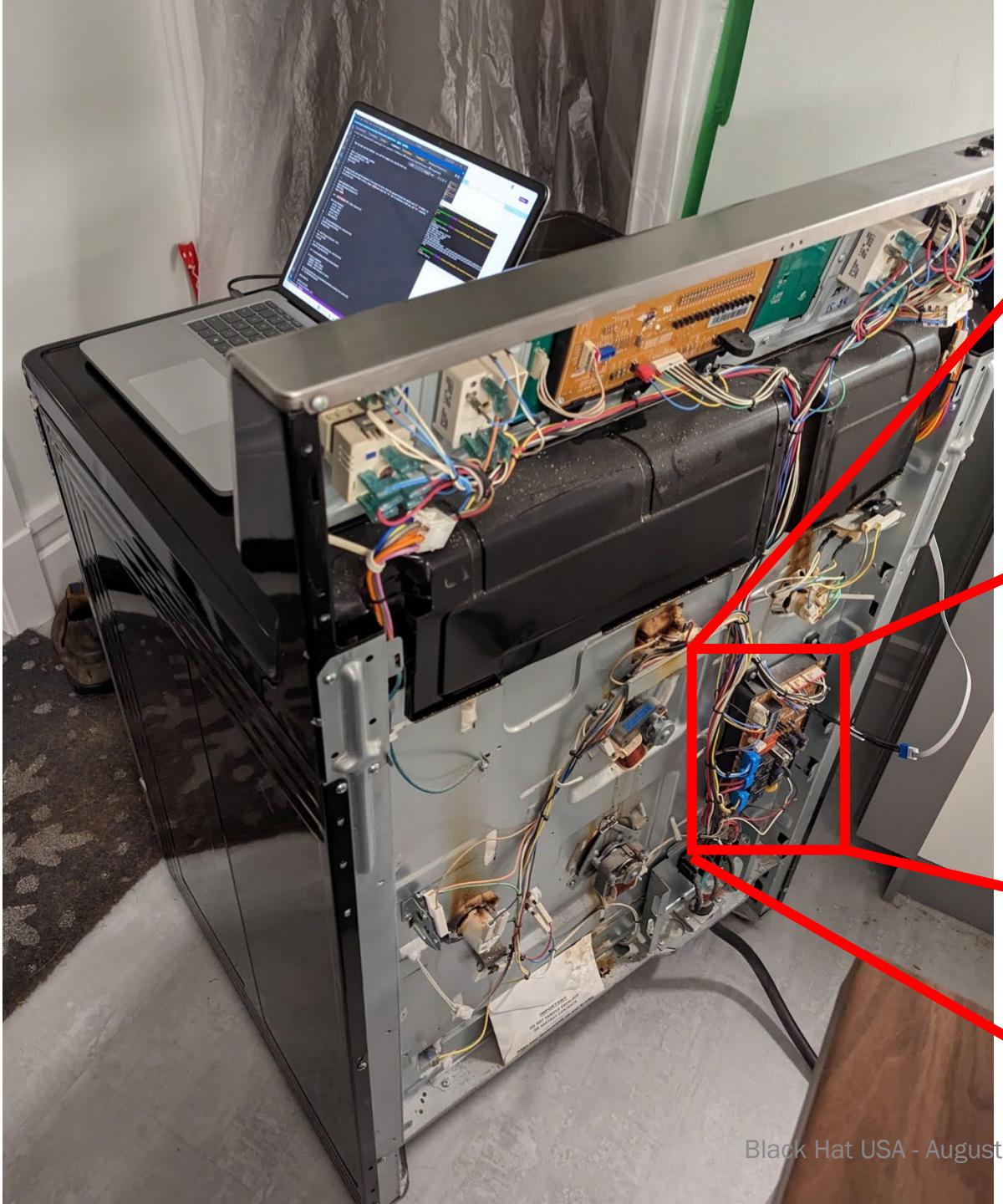
Oct 14, 2018

#4

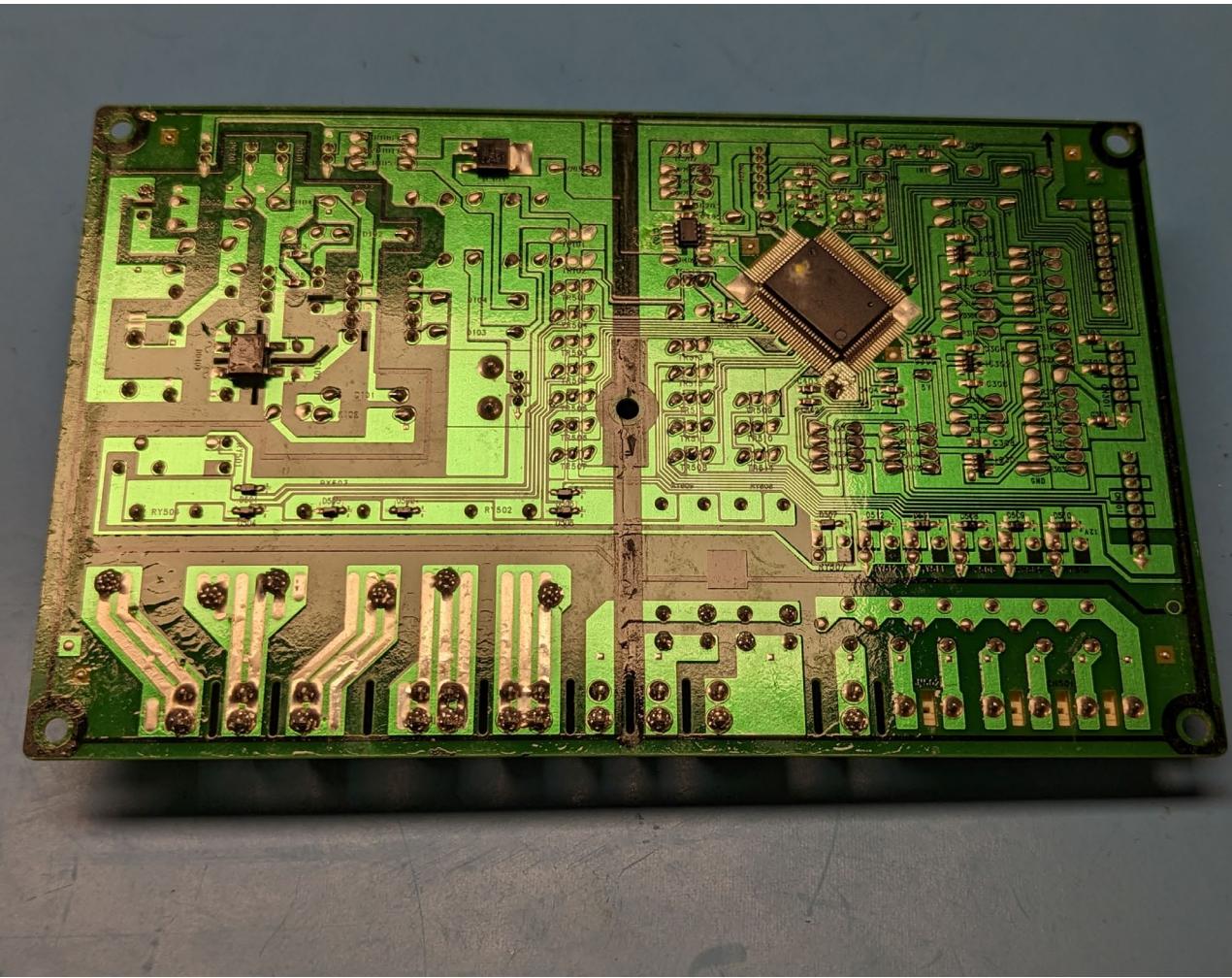
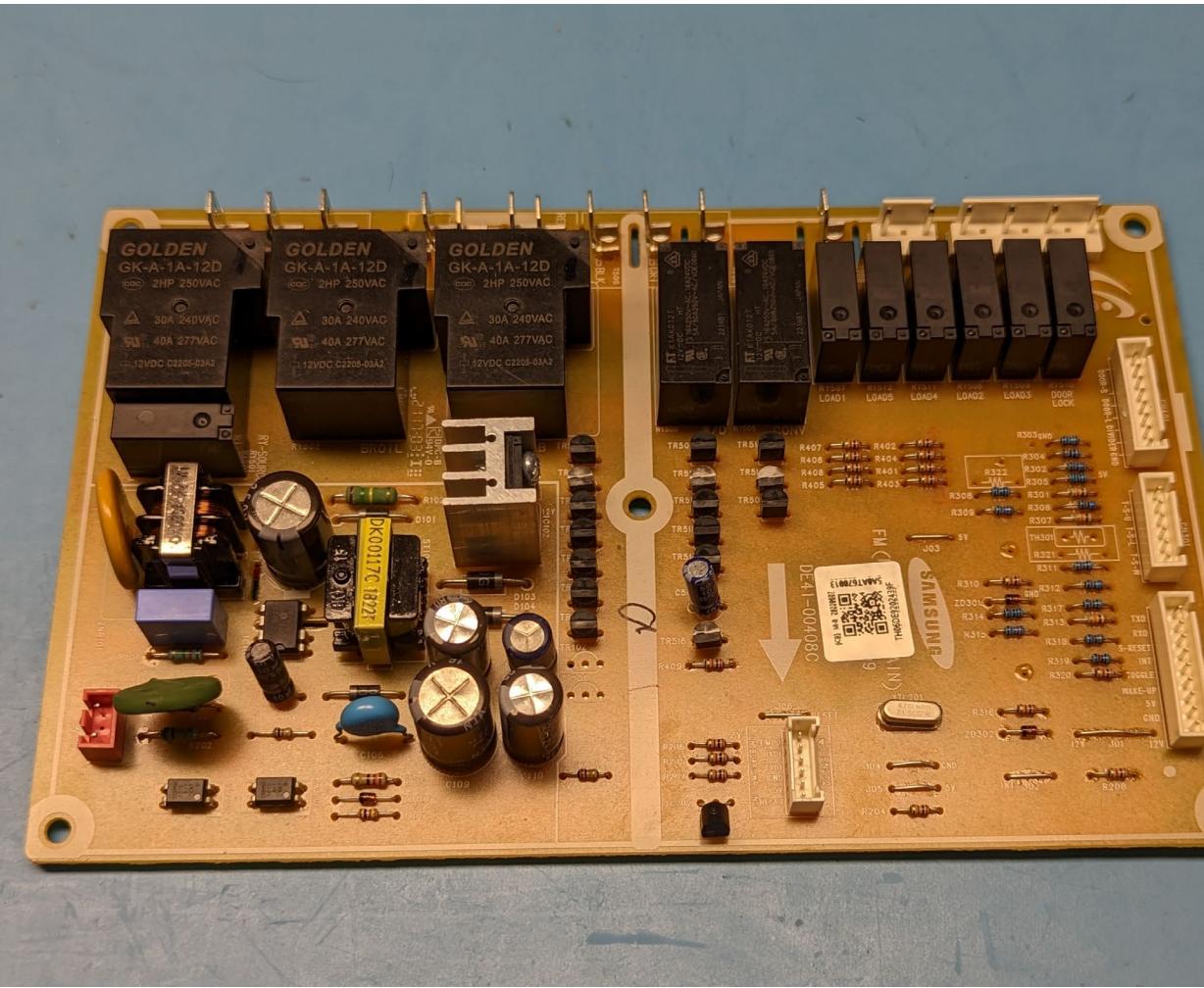
<https://www.applianceblog.com/mainforums/threads/samsung-fer300sx-will-not-maintain-temperature.68145/>

PID Controller?





Black Hat USA - August 10, 2023. Colin O'Flynn.



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

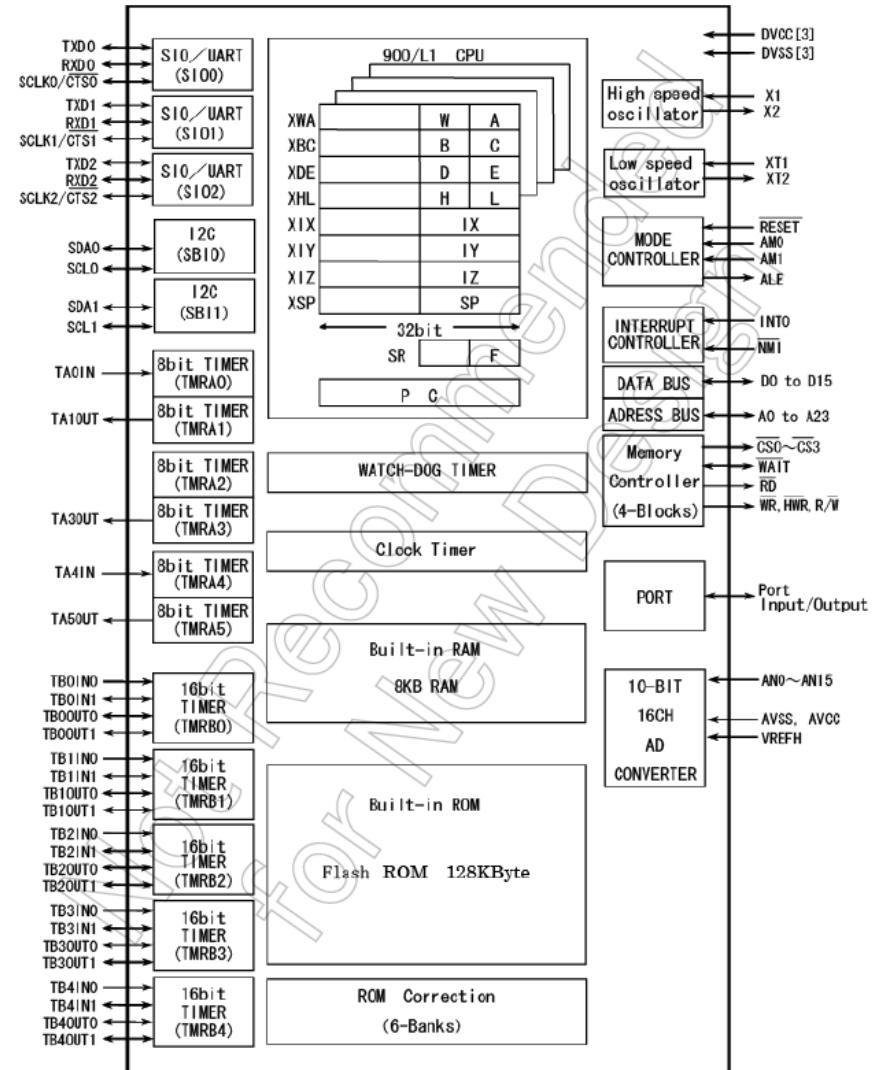


Figure 1-3 Block Diagram

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	UART Baud rate setting 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}		-

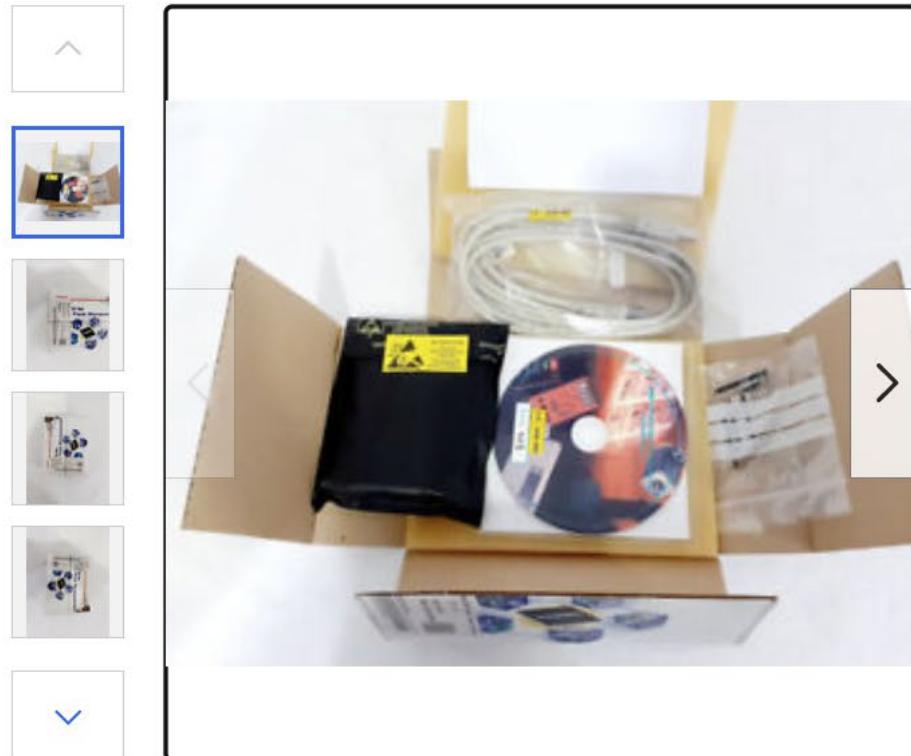
Table 14-12 Transfer Format of Single Boot Program [Flash Memory Protect Set]

	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

Buy another

Add to cart

\$ Have one to sell?

Sell now

Best Offer:

oven revd - TOSHIBA Integrated Development Environment

File Edit View Project Build Debug Tool Window Help

Debug Trace

Mixed Mode

Start Address : 0xff41e3

```
00FF41e3 PUSHL XIZ
00FF41e4 PUSHL XIY
00FF41e5 PUSHL XIX
00FF41e6 PUSHL XHL
00FF41e7 PUSHL XDE
00FF41e8 PUSHL XBC
00FF41e9 PUSHL XWA
00FF41ea CALL 0x0FF3C01
00FF41eb 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 LDH XHI,XDF
```

Address(A) : 0xffff00

0x000e885f: LDW HL,IX dc 8b
0x000e8861: LDW (0x12A8), HL F2 a8 12 00 53
0x000e8866: bit 0, (0x1200)
0x000e886a: jr z, 0xe88b0
0x000e886c: addw h1, 11 db c8 0b 00
0x000e8870: jr 0xFe88b0 68 3e
0x000e8872: ???

XWA : 0x000000c
XBC : 0x0000210
XDE : 0x0000670
XHL : 0x0000001
XIX : 0x0000015
XIY : 0x0000010
XIZ : 0x0000000
XSP : 0x0000010
INTNEST: 0x78
SR : 0xF80
PC : 0x00Fe8864
SYSH : 0x
IFF : 0x
MAX : 0x
RFP : 0x
SF : 0x
ZF : 0x
HF : 0x
UF : 0x
NF : 0x
CF : 0x

I-102-0100:Conn
I-102-0400:Restc
I-101-0000:Analj
I-101-1000:DownJ
I-101-1400:Objec
I-117-0103:SINME
I-117-0103:SINME
I-117-0103:SINME
I-117-0103:SINME
I-117-0103:SINME
I-117-0103:SINME
E-105-6400:Cannc
E-105-6400:Cannc
E-105-6400:Cannc
E-105-6500:Cannc
E-105-6400:Cannc
E-105-6400:Cannc
E-105-6400:Cannc
E-105-6400:Cannc

Black Hat USA - August 10, 2023. Colin O'Flynn.

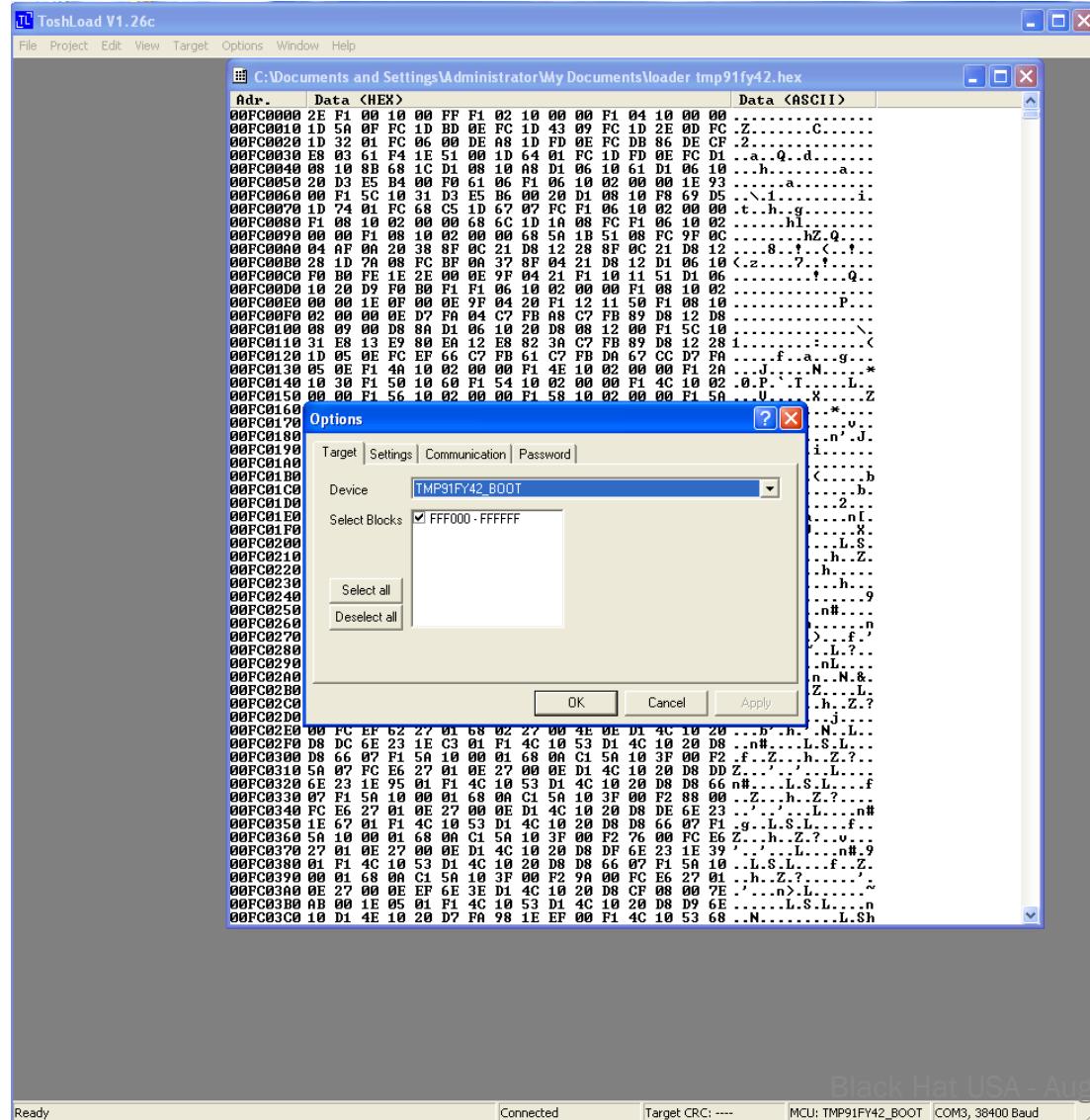
Windows XP?

Mirrored Here

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

15

Can you Read Back Bootloader?



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

00ffff2f5	CALR	0x0FFF75F <-- Load protection status
00ffff2f8	CPB	A,0x0FF <-- Compare protection status
00ffff2fb	JR	NZ,0x0FFF290 <-- Send error if protection enabled
00ffff2fd	CALR	0x0FFF2A2 <-- PW Check
00ffff300	CPB	RE1,0x0
00ffff303	JR	NZ,0x0FFF28A
00ffff305	CPB	RL1,0x0
00ffff308	JR	NZ,0x0FFF29C <-- Error
00ffff30a	CPB	L,0x0
00ffff30c	JR	NZ,0x0FFF29C <-- Error
00ffff30e	CALR	0x0FFF5EF <- TX
00ffff311	LDB	RH1,0x0
00ffff314	CALR	0x0FFF635 <--
00ffff317	LDB	QIXH,A

Important Take-Aways (for next stage)

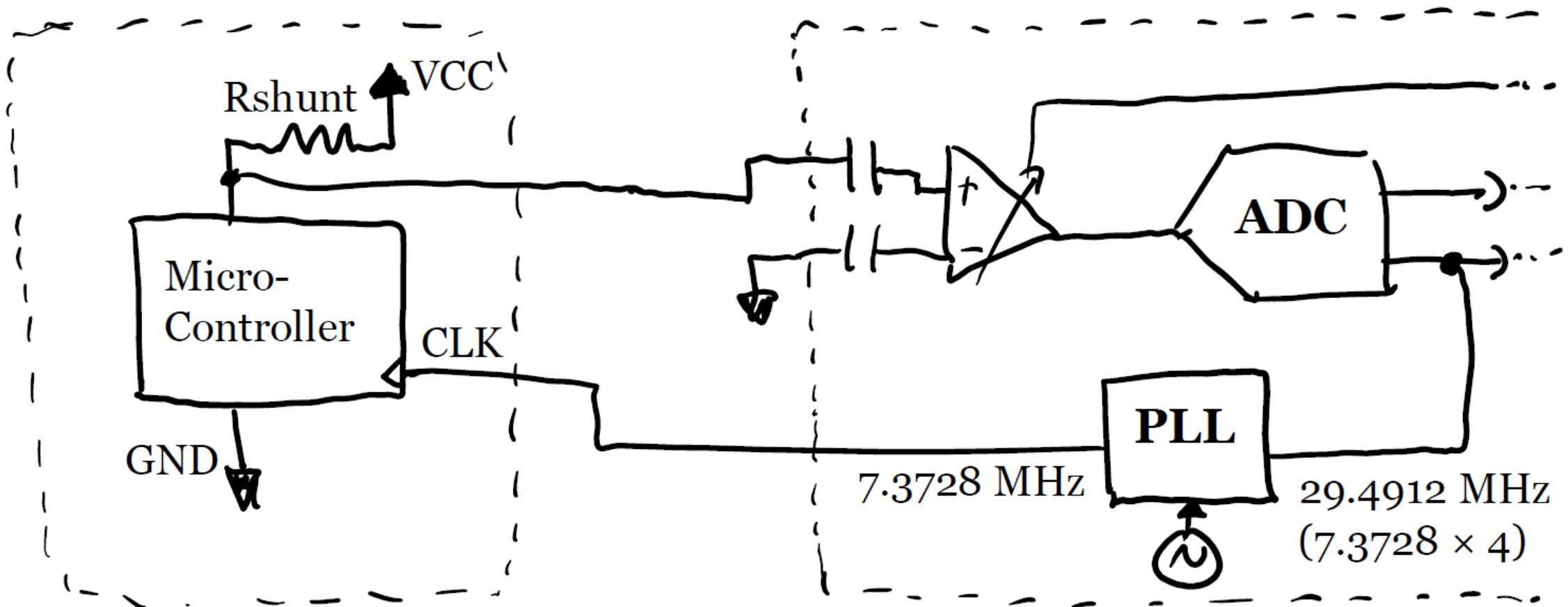
1. Password check has slight code-flow dependency.
2. Fuse byte check has obvious fault injection location.



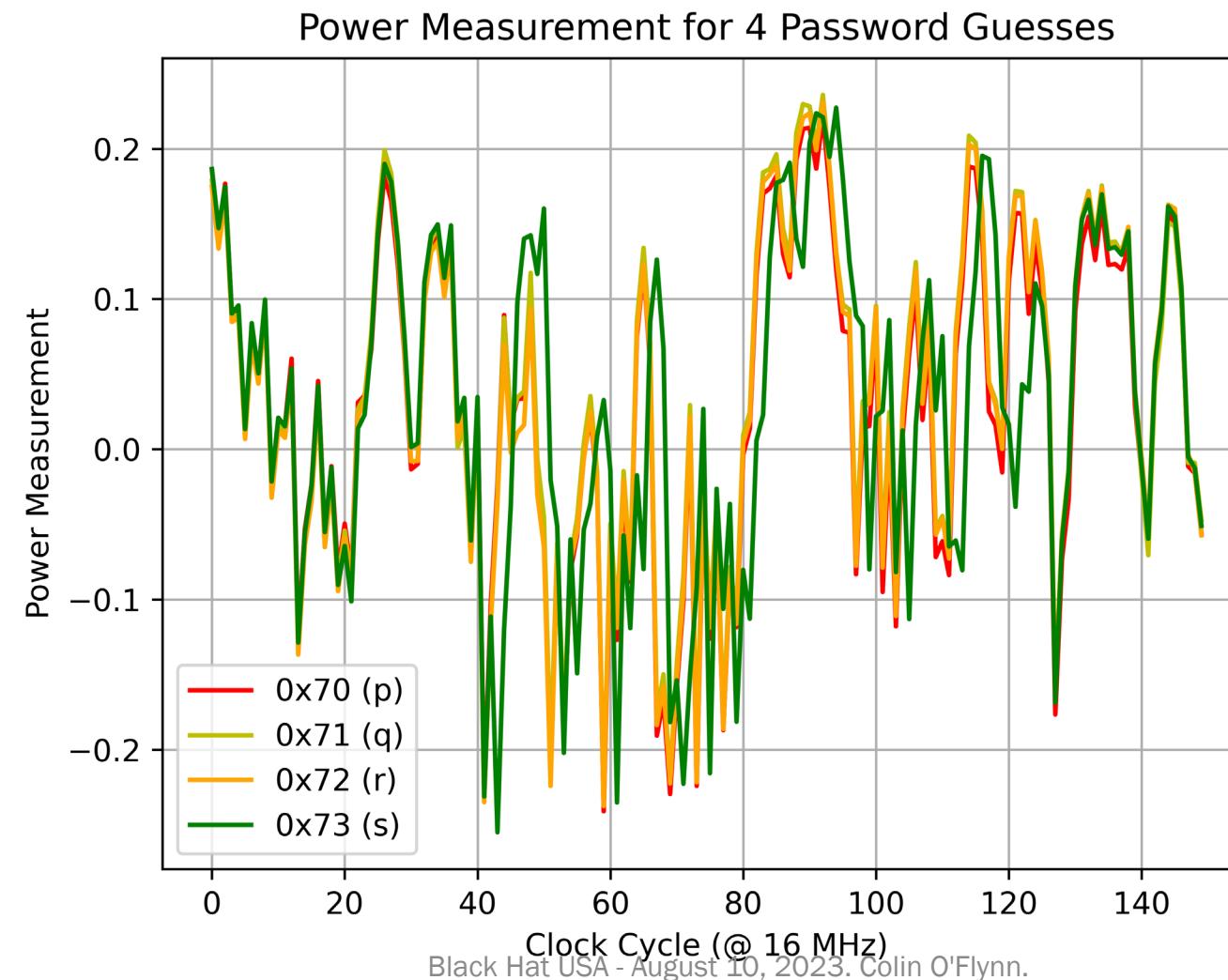
ChipWhisperer-Husky Intro



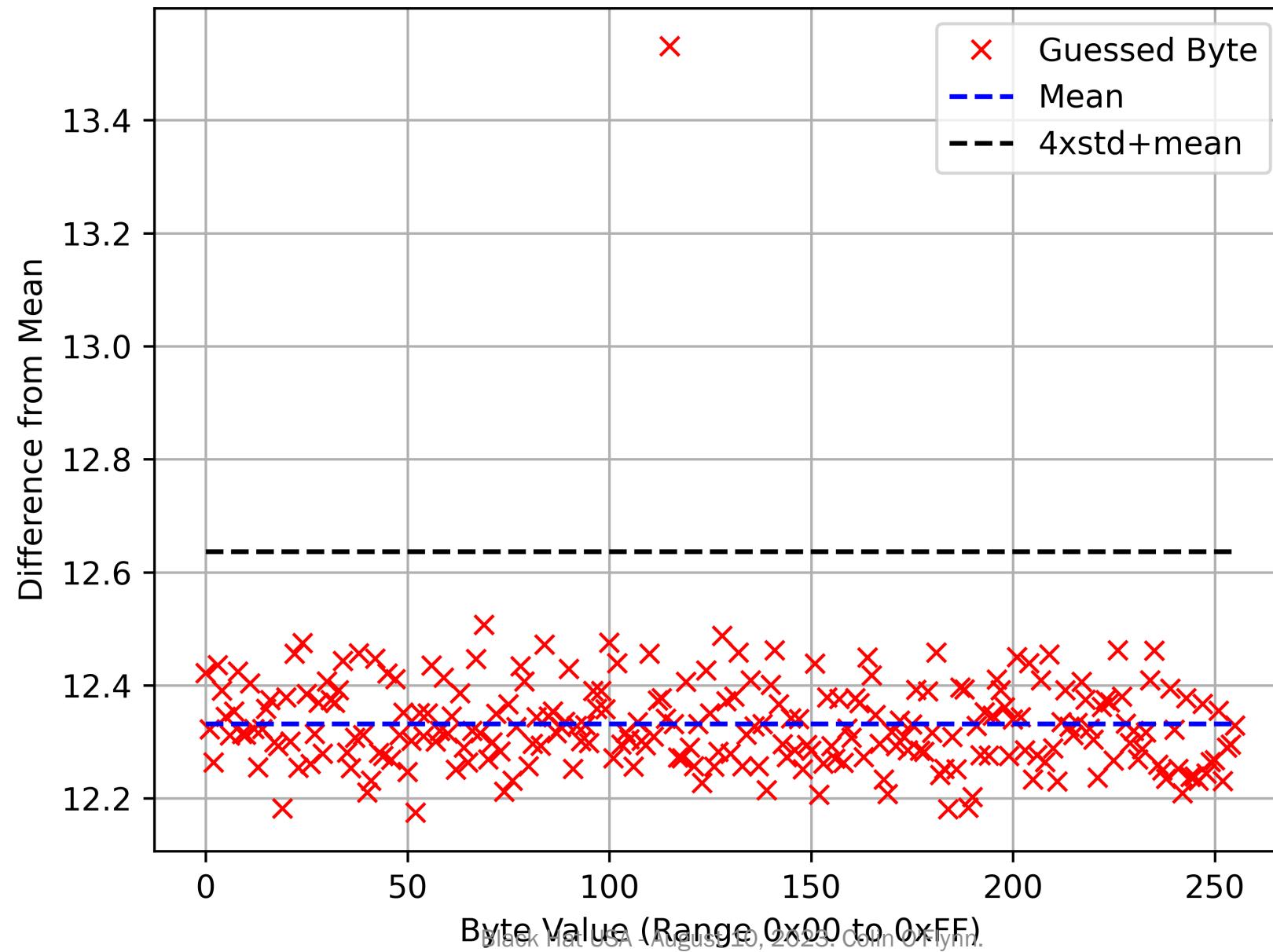
Power Analysis?



Easy-Mode Level 1: Password Power Analysis



Difference Between Guessed Power Trace & Mean



Fault Injection?

FUNCTION START: RAM WRITE FUNCTION

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

SE AB FF

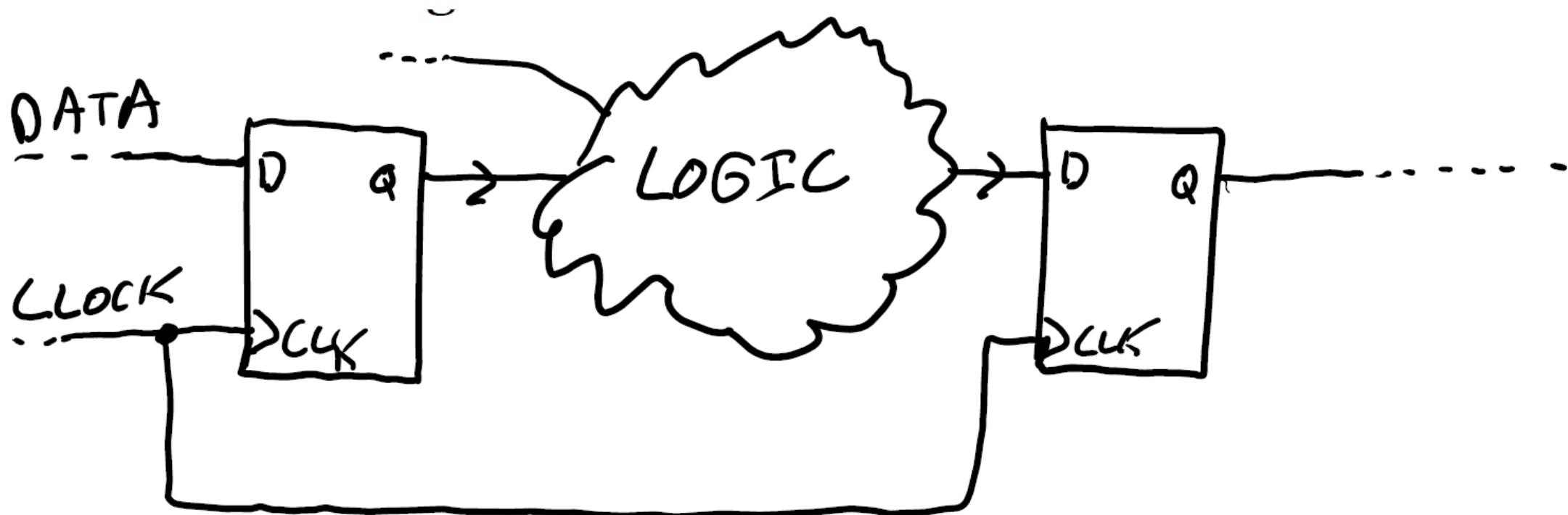
Fetch

CARBON TAX REFORM

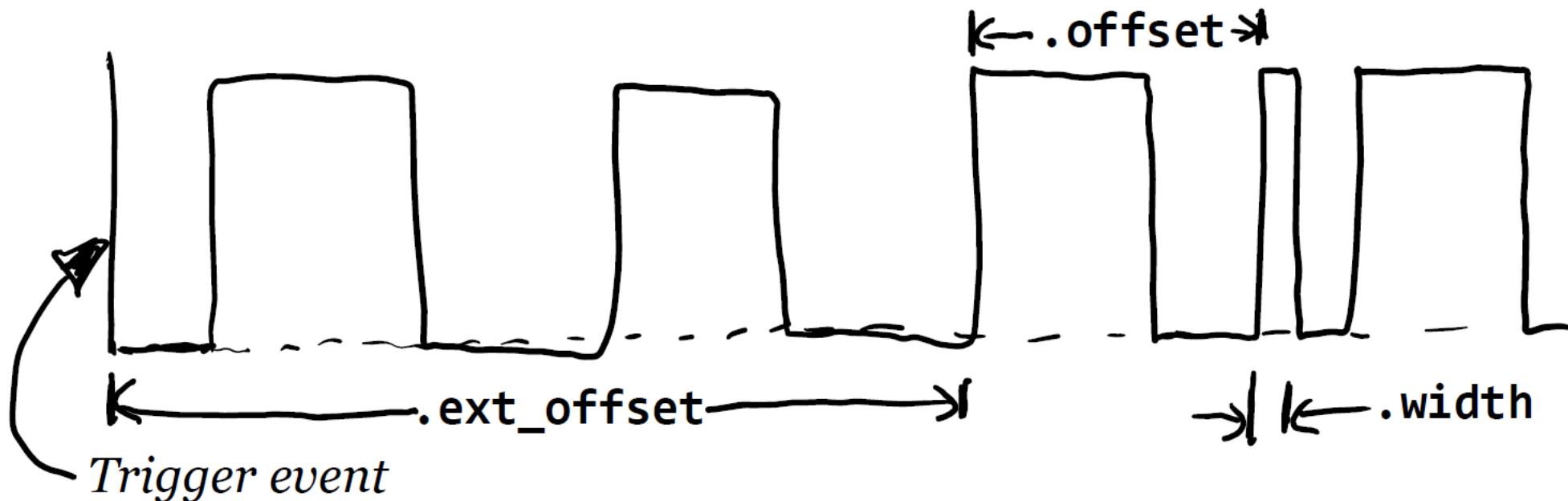
Decode

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	UART Baud rate setting 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")

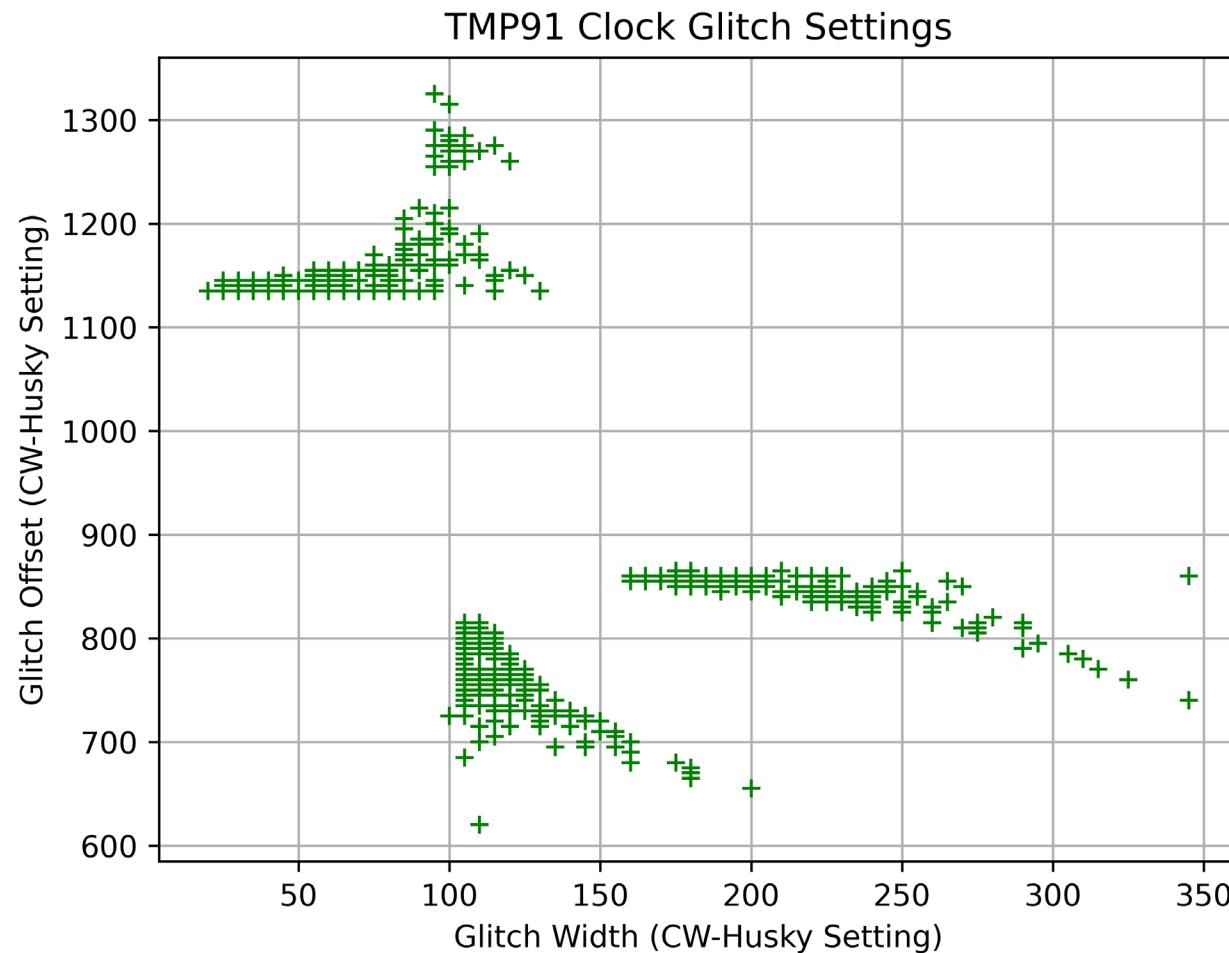
        #Device is slow to boot?
        reset_target()

    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.gitch.width = 2000 #100 #1000

for glitch_settings in gc.glitch_values():
    scope.gitch.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.gitch.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

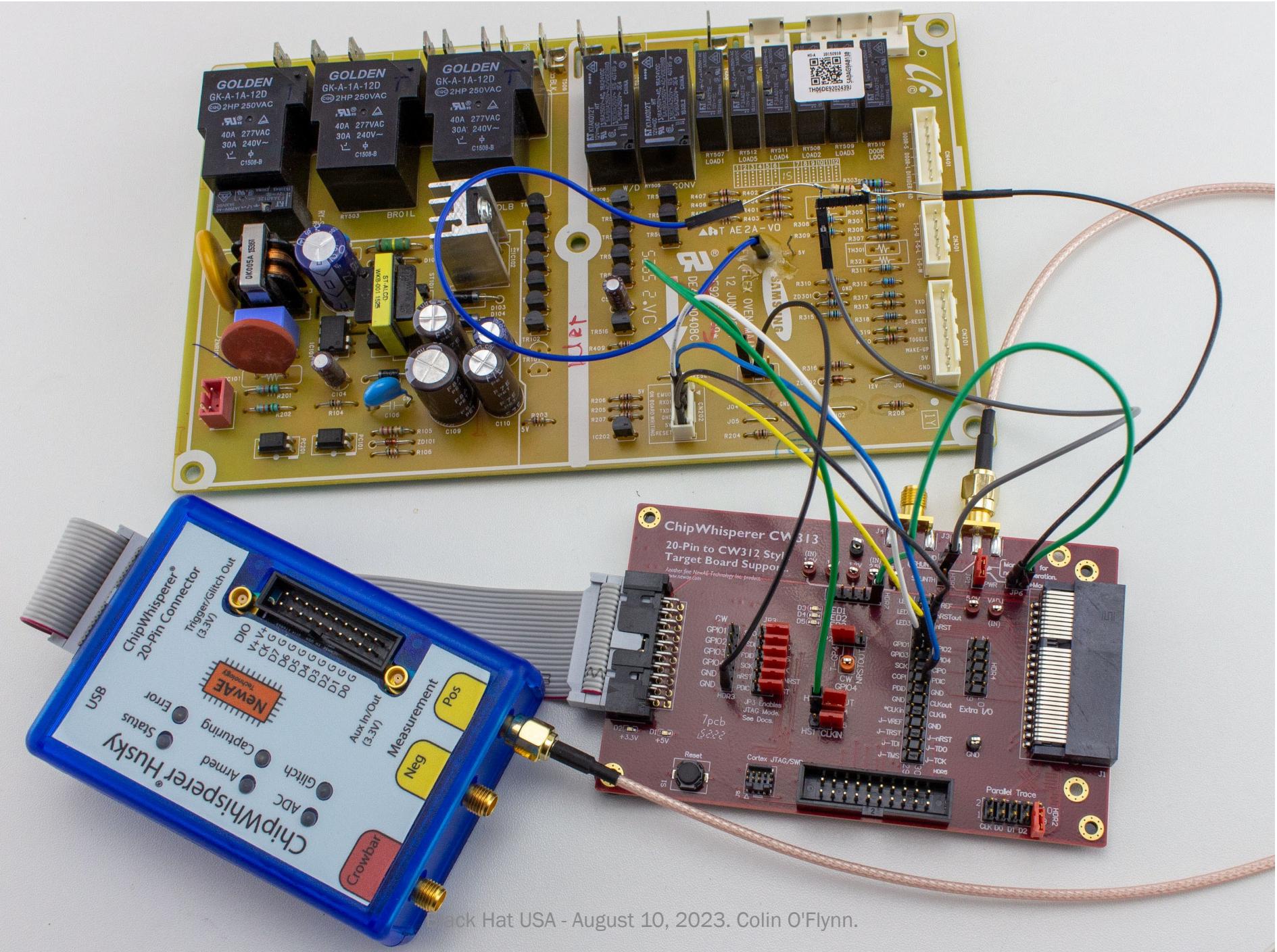
In [ ]: known_pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44]

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.





Medium-Mode Level 1: Power Analysis

In [18]:

```
%matplotlib notebook
import matplotlib.pyplot 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%02x" % 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

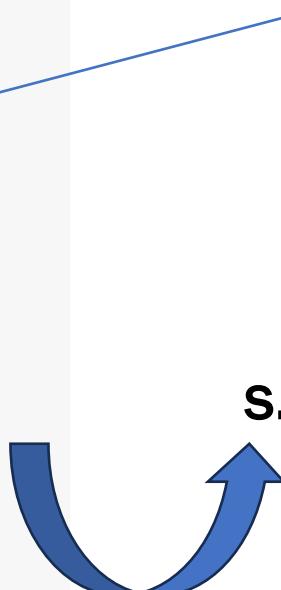
    diff = np.sum(abs(trace - trace1[start:end]))
    diffs.append(diff)
    print("%x %f" %(i, diff))

    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
[1]
```

```
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

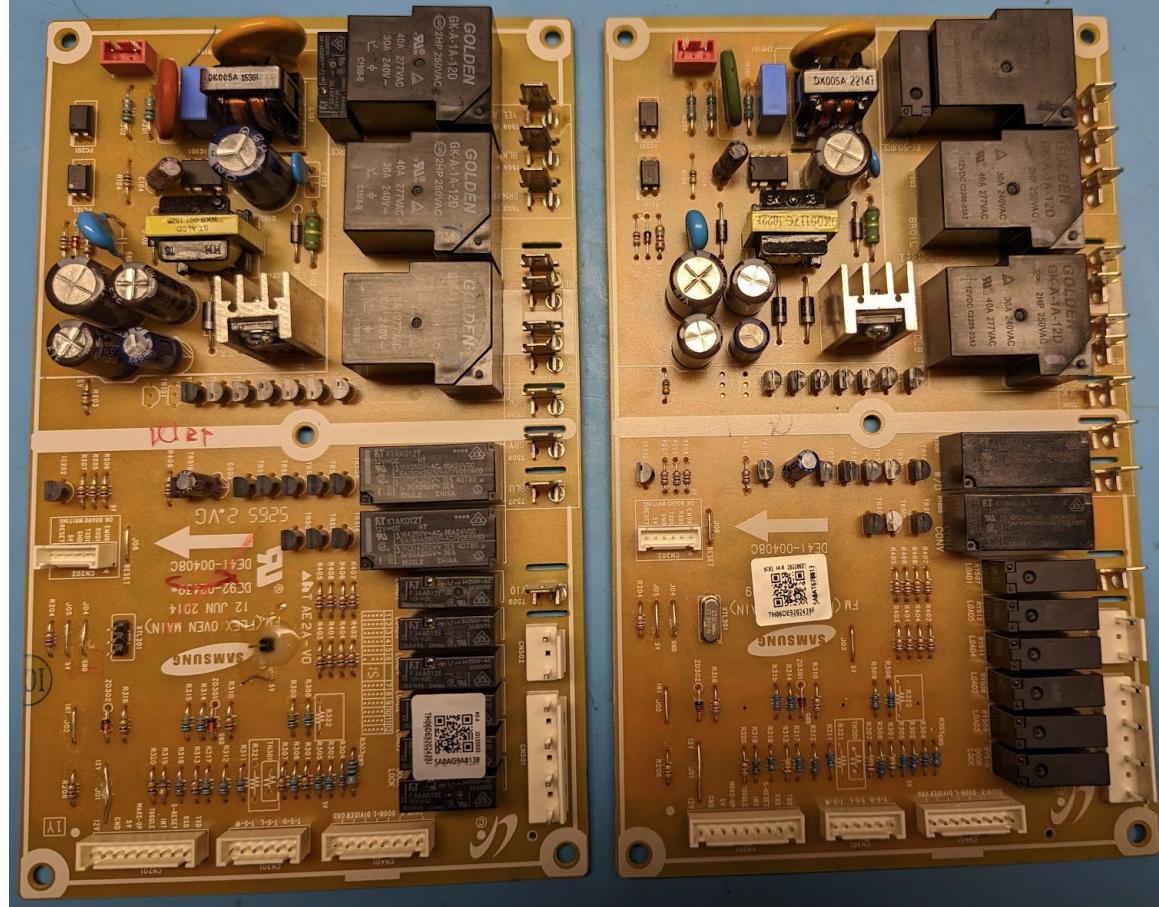
```
Out[12]: 'yyyyyyyyyyyyyyyyyyyy  
yyyyyyyyyyyyyyyyyyyyyyyy  
yyyyyyyy'
```

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

Read: n
Write:



\$\$ → Samsung Parts Department



Did they have problems with returns?

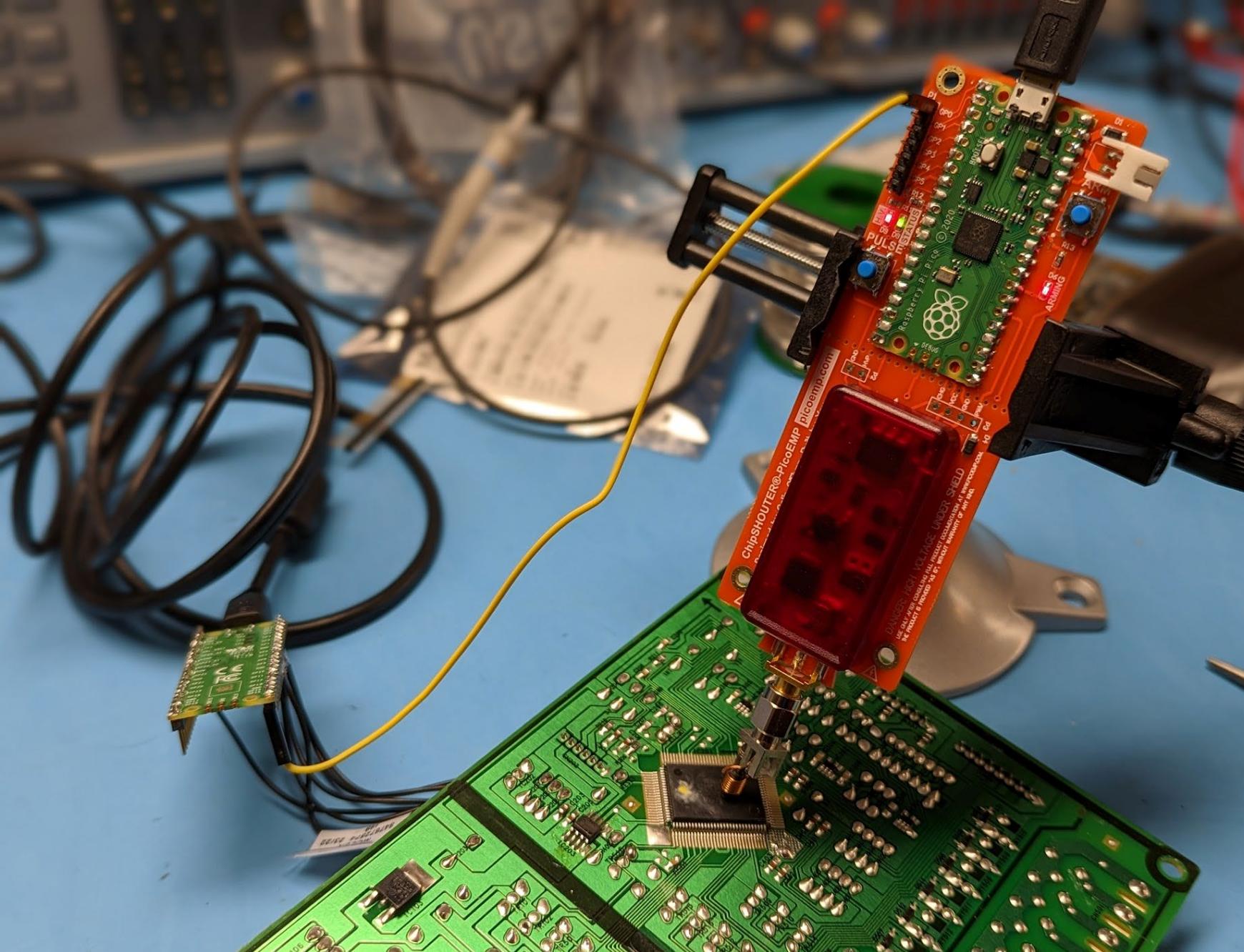
Or were they just
needed on the
boards!!



0000h:	C2 DA 13 00	3F 01 B0 F6	C2 F8 13 00	3F 00 B0 FE	ÀÚ...?..ºöÂø..?..ºþ
0010h:	C2 C6 12 00	3F 00 6E 08	C2 64 11 00	3F 00 B0 FE	ÀÈ..?..n.Ãd..?..ºþ
0020h:	C2 4E 13 00	3F 00 B0 FE	C2 1A 13 00	3F 00 B0 FE	ÀÑ..?..ºþÂ...?..ºþ
0030h:	C2 68 11 00	3F 00 B0 FE	C2 71 11 00	3F 00 B0 FE	Àh...?..ºþÂq..?..ºþ
0040h:	C2 7A 11 00	3F 00 B0 FE	D2 BC 12 00	3F 90 01 6B	Àz...?..ºþò¼..?..k
0050h:	09 D2 BE 12 00	3F 90 01 63	13 F2 A2 12 00	00 03	.º%..?..c.ºc....
0060h:	F2 20 13 00	00 01 F2 F0	10 00 00 08	0E C2 5A 12	ò ...òð.....Àz.
0070h:	00 3F 01 6E	13 F2 A2 12 00	00 03 F2 20 13 00	00	.?..n.ºc....ò ...
0080h:	02 F2 F0 10	00 08 0E	C2 3A 14 00	3F 01 6E 0E	.òð.....À:..?..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 00ò\....ÀX...?
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 00òð....òX....
00D0h:	0E 8F 04 23	C2 6A 11 00	3F 0A 66 08	C2 6A 11 00#Àj...?..f.Àj..
00E0h:	3F 0B 6E 16	C2 6B 11 00	3F 0F 6E 0E	C2 5C 12 00	?..n.Ãk..?..n.À\..
00F0h:	3F 05 66 06	F2 5C 12 00	00 03 CB 89	D8 12 D8 09	?..f.ò\....ÈºØ.Ø.
0100h:	09 00 F2 6A 11 00	32 F3 07 E8 E0	00 00 CB 89 D8	..òj..Zò.èa..ÈºØ	
0110h:	12 D8 09 09 00	F2 6B 11 00	32 F3 07 E8 E0	00 00	.Ø...òk..Zò.èa..
0120h:	CB 89 D8 12 D8 09 09 00	F2 68 11 00	32 F3 07 E8	ÈºØ.Ø...òh..Zò.è	
0130h:	E0 00 00 CB 89 D8 12 D8 09 09 00	F2 69 11 00	32	à..ÈºØ.Ø...òi..2	
0140h:	F3 07 E8 E0 00 00 CB 89 D8 12 D8 09 09 00	F2 6C	ò.èà..ÈºØ.Ø...òl		
0150h:	11 00 32 F3 07 E8 E0 00 00 CB 89 D8 12 D8 09 09	00	..Zò.èà..ÈºØ.Ø..		
0160h:	00 F2 6D 11 00 32 F3 07 E8 E0 00 00 CB 89 D8 12	00	òm..Zò.èà..ÈºØ.		
0170h:	D8 09 09 00 F2 6E 11 00 32 F3 07 E8 E0 00 00 CB	00	Ø...òn..Zò.èà..È		
0180h:	89 D8 12 D8 09 09 00 F2 6F 11 00 32 F3 07 E8 E0	00	%Ø.Ø...òo..Zò.èà		
0190h:	02 00 00 CB 89 D8 12 F2 98 12 00 32 F3 07 E8 E0	00	...ÈºØ.ò~..Zò.èà		
01A0h:	00 00 CB 89 D8 12 F2 34 14 00 32 F3 07 E8 E0 00	00	..ÈºØ.ò4..Zò.èà.		
01B0h:	00 CB 89 D8 12 D8 80 F2 FE 13 00 32 F3 07 E8 E0	00	.ÈºØ.Øèþ..Zò.èà		
01C0h:	02 00 00 C2 64 11 00 3F 00 6E 12 F2 A6 12 00 00	00	...Àd..?..n.ò!..		
01D0h:	00 F2 A7 12 00 00 00 F2 A8 12 00 00 00 CB D8 66	00	.ò§....ò~..ÈØf		
01E0h:	04 CB D9 6E 14 F2 D6 12 00 00 00 F2 DA 12 00 00	00	.ÈÙn.òò...òÙ...		
01F0h:	00 F2 68 12 00 00 00 68 0C F2 D8 12 00 00 00 F2	00	òh....h.òØ....ò		
0200h:	DC 12 00 00 00 F2 E8 13 00 00 00 0E F2 66 11 00	00	Ü....òè....òf..		
0210h:	00 00 F2 64 11 00 00 00 F2 1E 12 00 00 00 F2 96	00	..òd....ò....ò-		
0220h:	12 00 00 00 F2 9C 12 00 00 00 F2 5E 12 00 00 00	00òæ....ò^....		
0230h:	F2 66 12 00 02 00 00 F2 64 12 00 02 00 00 F2 8C	00	òf....òd....ò£		
0240h:	12 00 00 00 F2 18 12 00 02 00 00 F2 C0 12 00 00	00ò....ò....òÀ...		
0250h:	00 F2 54 11 00 02 00 00 F2 38 11 00 02 00 00 F2	00	.òT....ò8....ò		
0260h:	3C 14 00 00 00 0B 00 00 1D EF A3 FE EF 62 F2 9E	<.....i£þibòž			
0270h:	12 00 00 00 F2 A0 12 00 00 00 F2 A2 12 00 00 00	00ò....òc....		
0280h:	F2 20 13 00 00 00 C2 F8 13 00 3F 00 66 06 F2 FA	òÀø..?..f.òú		
0290h:	12 00 00 01 F2 40 14 00 00 00 F2 32 13 00 02 00	00	òø....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!

chipshouter-picoemp Public

Edit Pins Unwatch 17

main 1 branch 1 tag

Go to file Add file Code

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

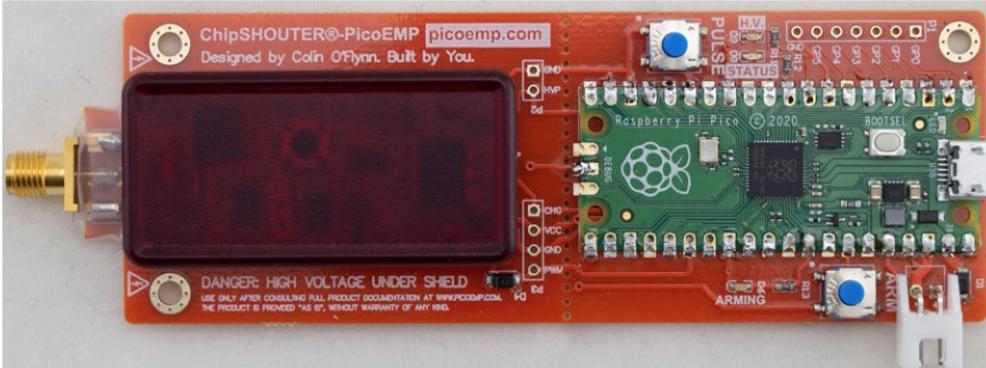
.gitignore add configurable duty_frac and pulse_time; add single character seria... last year

README.md Add one reference 2 months ago

README.md

ChipSHOUTER-PicoEMP

License CC BY-SA 3.0



CC-BA-SA 3.0 License!

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

Reverse Engineering Tools



The screenshot shows the official website for HexRays' IDA Pro. The top navigation bar includes links for Products, Solutions, Partners, Shop, Support, and Company. A prominent "Buy a license" button is visible. The main content area features a large image of the IDA Pro interface, which displays assembly code, memory dump, and debugger panes. A callout box highlights the "A powerful disassembler and a versatile debugger" feature. Below the interface, a detailed description of IDA Pro's capabilities is provided.

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.



File Home Insert Draw Page Layout Formulas Data Review View Help Acrobat Table Design Query

Clipboard Font Alignment Number Styles Cells Editing Analysis

Paste Calibri 11 A⁺ A⁻ Wrap Text General \$ % , .00 .00 Conditional Formatting Merge & Center Format as Table Cell Styles Insert Delete Format Sort & Filter Find & Select Analyze Data

	Column2	Column3	Column4	Column5	Column6	G	H	I	J	K	L	M	N	O	P	Q	R
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													
9933	0xfe969b	66 0b	JR Z,0x0FE96A8														
9934	0xfe969d	00	NOP														
9935	0xfe969e	c1 08 02 21	LDB A,(0x208)														
9936	0xfe96a2	f2 00 11 00 41	LDB (0x1100),A														
9937	0xfe96a7	0e	RET														
9938	0xfe96a8	f1 09 02 ca	BITB 0x2,(0x209)														
9939	0xfe96ac	66 0b	JR Z,0x0FE96B9														
9940	0xfe96ae	00	NOP														
9941	0xfe96af	c1 08 02 21	LDB A,(0x208)														
9942	0xfe96b3	f2 00 11 00 41	LDB (0x1100),A														
9943	0xfe96b8	0e	RET														
9944	0xfe96b9	f1 09 02 cc	BITB 0x4,(0x209)														
9945	0xfe96bd	66 0b	JR Z,0x0FE96CA														
9946	0xfe96bf	00	NOP														
9947	0xfe96c0	c1 08 02 21	LDB A,(0x208)														
9948	0xfe96c4	f2 00 11 00 41	LDB (0x1100),A														
9949	0xfe96c9	0e	RET														
9950	0xfe96ca	c2 20 11 00 21	LDB A,(0x1120)	What is 1120??													
9951	0xfe96cf	c9 8b	LDB C,A														
9952	0xfe96d1	d9 12	EXTZW BC	BC has (0x1120) data													
9953	0xfe96d3	f2 10 11 00 32	LDAL XDE,0x1110														
9954	0xfe96d8	c1 08 02 21	LDB A,(0x208)	RX Byte													
9955	0xfe96dc	f3 07 e8 e4 41	LDB (XDE+BC),A	Load byte here?													
9956	0xfe96e1	c2 20 11 00 3f 00	CPB (0x1120),0x0														
9957	0xfe96e7	6e 10	JR NZ,0x0FE96F9	Fail I guess?													

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.



DE

02439E

OK, Just F



In [11]: ➜ resp = rcp.cmd_read

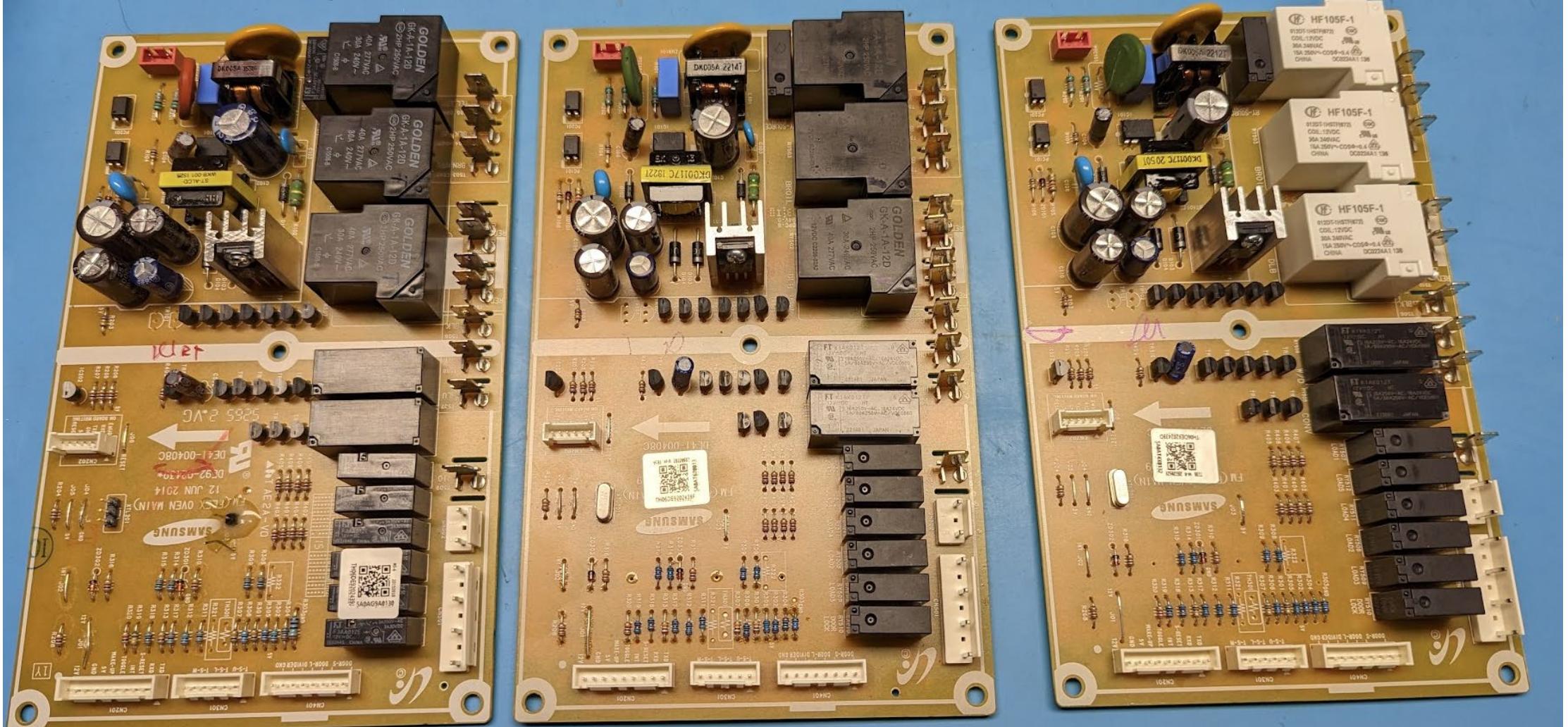
In [12]: ➤ resp

```
out[12]: 'yyyyyyyyyyyyyyyyyyyy  
yyyyyyyyyyyyyyyyyyyyyyyy  
yyyyyyyy'
```

```
In [7]: bl = t1  
#bl.cmd  
#bl.cmd  
bl.cmd_
```

Read: ✓
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...



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!

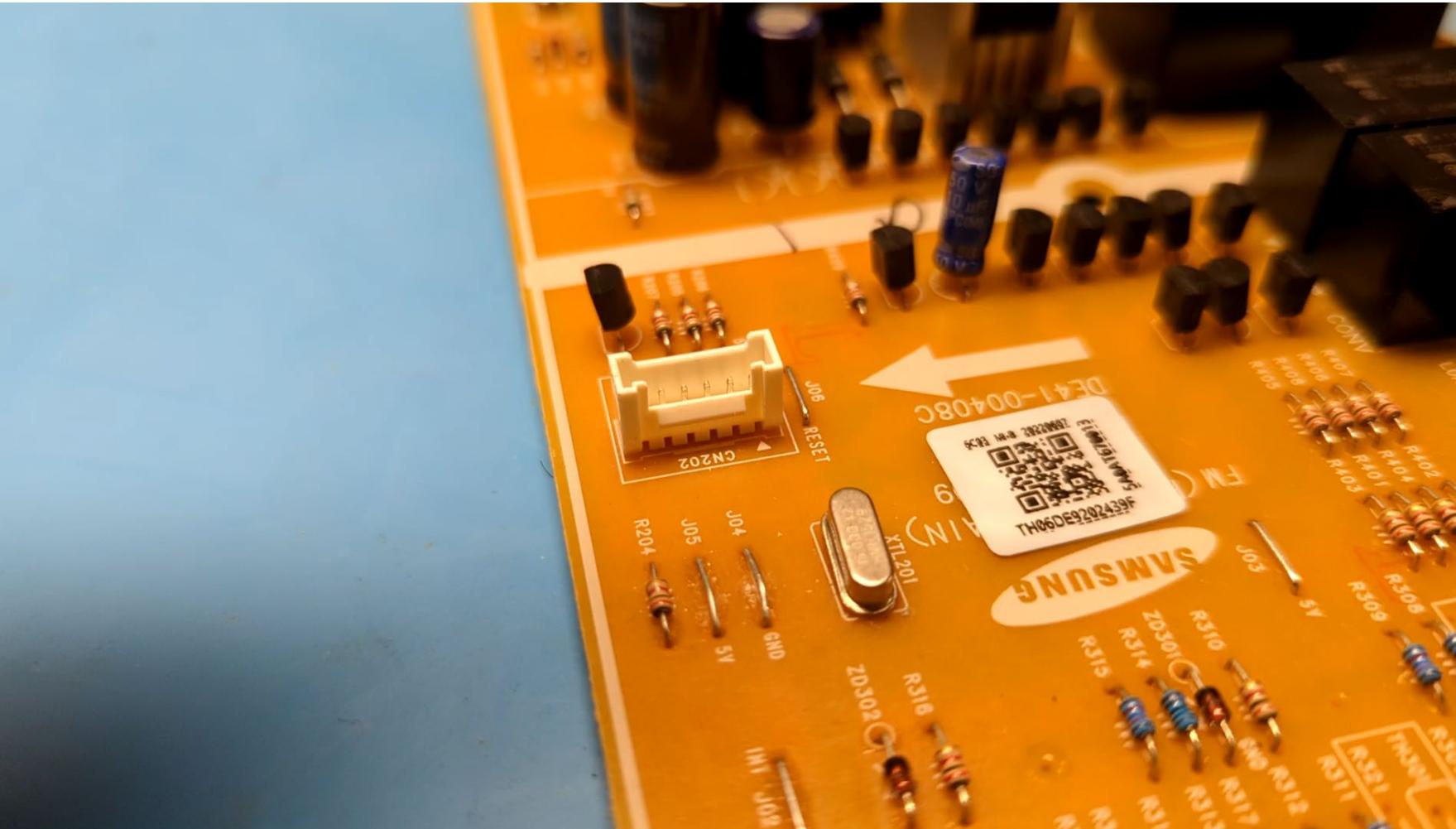
..Add the Serial Monitor

The screenshot shows a Microsoft Excel spreadsheet titled "RE Oven REVD.xlsx". The table has columns labeled "in2", "Column3", "Column4", "Column5", "Column6", "Column7", and "Column". The data consists of memory dump and assembly code. A new row, "RX Interrupt", is being added at the bottom of the table.

in2	Column3	Column4	Column5	Column6	Column7	Column
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
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]: #addrss = [0x1200, 0x1342, 0x110E, 0x11F6, 0x11F8, 0x11E4, 0x11FA, 0x110A]
#addrss = [0x1113, 0x110a, 0x1213, 0x1117, 0x10c0, 0x11ec, 0x1216, 0x11ea, 0x11ee, 0x11f0, 0x1120, 0x1248]
addrss = [0x12A8, 0x1248, 0x122A, 0x1232]
addr_data_list = [[] for _ in addrss]

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

while True:
    time.sleep(1)

    for i,addr in enumerate(addrss):
        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)"%(addrss[i], old, new, new))
        else:
            print("%04x: %2x (%d)"%(addrss[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)
```

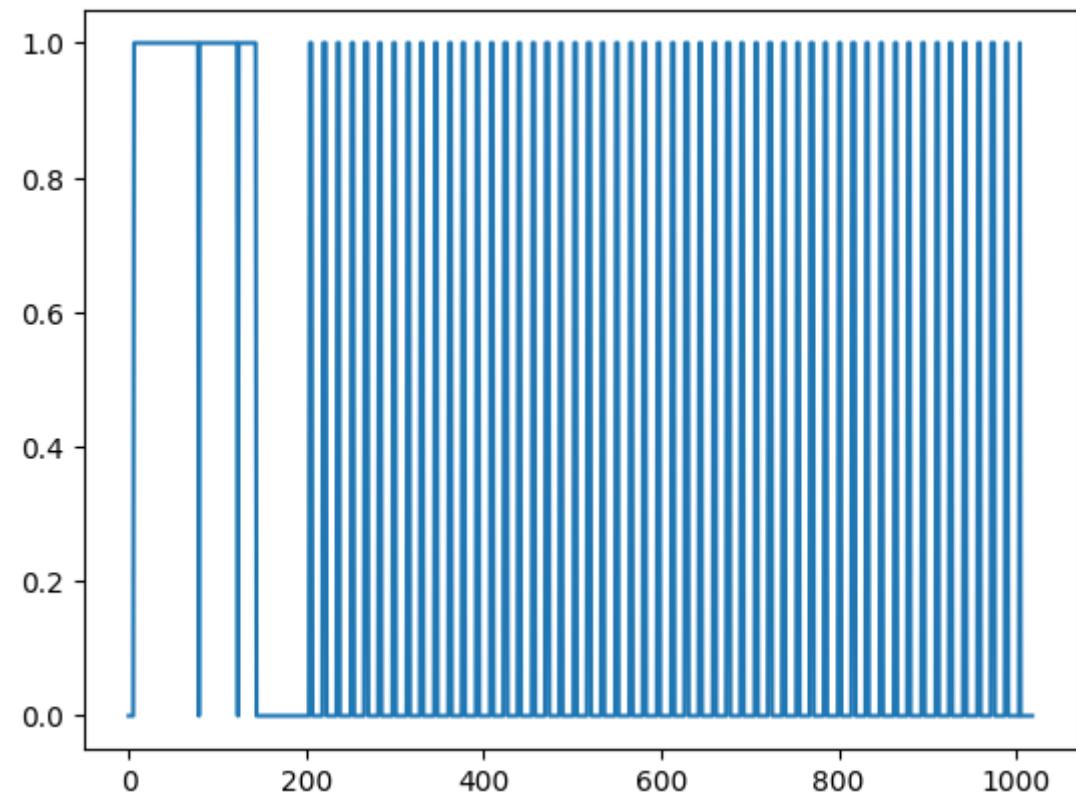
The screenshot shows a Microsoft Excel spreadsheet with several tabs visible at the top. The active tab is 'Column3'. The table contains assembly code in columns 3 and 4, and various memory addresses and values in columns 5 through 8. A 'Find and Replace' dialog box is open on the right side of the screen, with the search term '0x124A' entered in the 'Find what:' field. The 'Look in:' dropdown is set to 'Formulas'.

Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column15	Column16	Column17	Column18	Column19	Column20	Column21	Column22	Column23	Column24	Column25	Column26	Column27	Column28	Column29	Column30	Column31	Column32	Column33	Column34	Column35	Column36	Column37	Column38	Column39	Column40	Column41	Column42	Column43	Column44	Column45	Column46	Column47	Column48	Column49	Column50	Column51	Column52	Column53	Column54	Column55	Column56	Column57	Column58	Column59	Column60	Column61	Column62	Column63	Column64	Column65	Column66	Column67	Column68	Column69	Column70	Column71	Column72	Column73	Column74	Column75	Column76	Column77	Column78	Column79	Column80	Column81	Column82	Column83	Column84	Column85	Column86	Column87	Column88	Column89	Column90	Column91	Column92	Column93	Column94	Column95	Column96	Column97	Column98	Column99	Column100	Column101	Column102	Column103	Column104	Column105	Column106	Column107	Column108	Column109	Column110	Column111	Column112	Column113	Column114	Column115	Column116	Column117	Column118	Column119	Column120	Column121	Column122	Column123	Column124	Column125	Column126	Column127	Column128	Column129	Column130	Column131	Column132	Column133	Column134	Column135	Column136	Column137	Column138	Column139	Column140	Column141	Column142	Column143	Column144	Column145	Column146	Column147	Column148	Column149	Column150	Column151	Column152	Column153	Column154	Column155	Column156	Column157	Column158	Column159	Column160	Column161	Column162	Column163	Column164	Column165	Column166	Column167	Column168	Column169	Column170	Column171	Column172	Column173	Column174	Column175	Column176	Column177	Column178	Column179	Column180	Column181	Column182	Column183	Column184	Column185	Column186	Column187	Column188	Column189	Column190	Column191	Column192	Column193	Column194	Column195	Column196	Column197	Column198	Column199	Column200	Column201	Column202	Column203	Column204	Column205	Column206	Column207	Column208	Column209	Column210	Column211	Column212	Column213	Column214	Column215	Column216	Column217	Column218	Column219	Column220	Column221	Column222	Column223	Column224	Column225	Column226	Column227	Column228	Column229	Column230	Column231	Column232	Column233	Column234	Column235	Column236	Column237	Column238	Column239	Column240	Column241	Column242	Column243	Column244	Column245	Column246	Column247	Column248	Column249	Column250	Column251	Column252	Column253	Column254	Column255	Column256	Column257	Column258	Column259	Column260	Column261	Column262	Column263	Column264	Column265	Column266	Column267	Column268	Column269	Column270	Column271	Column272	Column273	Column274	Column275	Column276	Column277	Column278	Column279	Column280	Column281	Column282	Column283	Column284	Column285	Column286	Column287	Column288	Column289	Column290	Column291	Column292	Column293	Column294	Column295	Column296	Column297	Column298	Column299	Column300	Column301	Column302	Column303	Column304	Column305	Column306	Column307	Column308	Column309	Column310	Column311	Column312	Column313	Column314	Column315	Column316	Column317	Column318	Column319	Column320	Column321	Column322	Column323	Column324	Column325	Column326	Column327	Column328	Column329	Column330	Column331	Column332	Column333	Column334	Column335	Column336	Column337	Column338	Column339	Column340	Column341	Column342	Column343	Column344	Column345	Column346	Column347	Column348	Column349	Column350	Column351	Column352	Column353	Column354	Column355	Column356	Column357	Column358	Column359	Column360	Column361	Column362	Column363	Column364	Column365	Column366	Column367	Column368	Column369	Column370	Column371	Column372	Column373	Column374	Column375	Column376	Column377	Column378	Column379	Column380	Column381	Column382	Column383	Column384	Column385	Column386	Column387	Column388	Column389	Column390	Column391	Column392	Column393	Column394	Column395	Column396	Column397	Column398	Column399	Column400	Column401	Column402	Column403	Column404	Column405	Column406	Column407	Column408	Column409	Column410	Column411	Column412	Column413	Column414	Column415	Column416	Column417	Column418	Column419	Column420	Column421	Column422	Column423	Column424	Column425	Column426	Column427	Column428	Column429	Column430	Column431	Column432	Column433	Column434	Column435	Column436	Column437	Column438	Column439	Column440	Column441	Column442	Column443	Column444	Column445	Column446	Column447	Column448	Column449	Column450	Column451	Column452	Column453	Column454	Column455	Column456	Column457	Column458	Column459	Column460	Column461	Column462	Column463	Column464	Column465	Column466	Column467	Column468	Column469	Column470	Column471	Column472	Column473	Column474	Column475	Column476	Column477	Column478	Column479	Column480	Column481	Column482	Column483	Column484	Column485	Column486	Column487	Column488	Column489	Column490	Column491	Column492	Column493	Column494	Column495	Column496	Column497	Column498	Column499	Column500	Column501	Column502	Column503	Column504	Column505	Column506	Column507	Column508	Column509	Column510	Column511	Column512	Column513	Column514	Column515	Column516	Column517	Column518	Column519	Column520	Column521	Column522	Column523	Column524	Column525	Column526	Column527	Column528	Column529	Column530	Column531	Column532	Column533	Column534	Column535	Column536	Column537	Column538	Column539	Column540	Column541	Column542	Column543	Column544	Column545	Column546	Column547	Column548	Column549	Column550	Column551	Column552	Column553	Column554	Column555	Column556	Column557	Column558	Column559	Column560	Column561	Column562	Column563	Column564	Column565	Column566	Column567	Column568	Column569	Column570	Column571	Column572	Column573	Column574	Column575	Column576	Column577	Column578	Column579	Column580	Column581	Column582	Column583	Column584	Column585	Column586	Column587	Column588	Column589	Column590	Column591	Column592	Column593	Column594	Column595	Column596	Column597	Column598	Column599	Column600	Column601	Column602	Column603	Column604	Column605	Column606	Column607	Column608	Column609	Column610	Column611	Column612	Column613	Column614	Column615	Column616	Column617	Column618	Column619	Column620	Column621	Column622	Column623	Column624	Column625	Column626	Column627	Column628	Column629	Column630	Column631	Column632	Column633	Column634	Column635	Column636	Column637	Column638	Column639	Column640	Column641	Column642	Column643	Column644	Column645	Column646	Column647	Column648	Column649	Column650	Column651	Column652	Column653	Column654	Column655	Column656	Column657	Column658	Column659	Column660	Column661	Column662	Column663	Column664	Column665	Column666	Column667	Column668	Column669	Column670	Column671	Column672	Column673	Column674	Column675	Column676	Column677	Column678	Column679	Column680	Column681	Column682	Column683	Column684	Column685	Column686	Column687	Column688	Column689	Column690	Column691	Column692	Column693	Column694	Column695	Column696	Column697	Column698	Column699	Column700	Column701	Column702	Column703	Column704	Column705	Column706	Column707	Column708	Column709	Column710	Column711	Column712	Column713	Column714	Column715	Column716	Column717	Column718	Column719	Column720	Column721	Column722	Column723	Column724	Column725	Column726	Column727	Column728	Column729	Column730	Column731	Column732	Column733	Column734	Column735	Column736	Column737	Column738	Column739	Column740	Column741	Column742	Column743	Column744	Column745	Column746	Column747	Column748	Column749	Column750	Column751	Column752	Column753	Column754	Column755	Column756	Column757	Column758	Column759	Column760	Column761	Column762	Column763	Column764	Column765	Column766	Column767	Column768	Column769	Column770	Column771	Column772	Column773	Column774	Column775	Column776	Column777	Column778	Column779	Column780	Column781	Column782	Column783	Column784	Column785	Column786	Column787	Column788	Column789	Column790	Column791	Column792	Column793	Column794	Column795	Column796	Column797	Column798	Column799	Column800	Column801	Column802	Column803	Column804	Column805	Column806	Column807	Column808	Column809	Column810	Column811	Column812	Column813	Column814	Column815	Column816	Column817	Column818	Column819	Column820	Column821	Column822	Column823	Column824	Column825	Column826	Column827	Column828	Column829	Column830	Column831	Column832	Column833	Column834	Column835	Column836	Column837	Column838	Column839	Column840	Column841	Column842	Column843	Column844	Column845	Column846	Column847	Column848	Column849	Column850	Column851	Column852	Column853	Column854	Column855	Column856	Column857	Column858	Column859	Column860	Column861	Column862	Column863	Column864	Column865	Column866	Column867	Column868	Column869	Column870	Column871	Column872	Column873	Column874	Column875	Column876	Column877	Column878	Column879	Column880	Column881	Column882	Column883	Column884	Column885	Column886	Column887	Column888	Column889	Column890	Column891	Column892	Column893	Column894	Column895	Column896	Column897	Column898	Column899	Column900	Column901	Column902	Column903	Column904	Column905	Column906	Column907	Column908	Column909	Column910	Column911	Column912	Column913	Column914	Column915	Column916	Column917	Column918	Column919	Column920	Column921	Column922	Column923	Column924	Column925	Column926	Column927	Column928	Column929	Column930	Column931	Column932	Column933	Column934	Column935	Column936	Column937	Column938	Column939	Column940	Column941	Column942	Column943	Column944	Column945	Column946	Column947	Column948	Column949	Column950	Column951	Column952	Column953	Column954	Column955	Column956	Column957	Column958	Column959	Column960	Column961	Column962	Column963	Column964	Column965	Column966	Column967	Column968	Column969	Column970	Column971	Column972	Column973	Column974	Column975	Column976	Column977	Column978	Column979	Column980	Column981	Column982	Column983	Column984	Column985	Column986	Column987	Column988	Column989	Column990	Column991	Column992	Column993	Column994	Column995	Column996	Column997	Column998	Column999	Column1000	Column1001	Column1002	Column1003	Column1004	Column1005	Column1006	Column1007	Column1008	Column1009	Column1010	Column1011	Column1012	Column1013	Column1014	Column1015	Column1016	Column1017	Column1018	Column1019	Column1020	Column1021	Column1022	Column1023	Column1024	Column1025	Column1026	Column1027	Column1028	Column1029	Column1030	Column1031	Column1032	Column1033	Column1034	Column1035	Column1036	Column1037	Column1038	Column1039	Column1040	Column1041	Column1042	Column1043	Column1044	Column1045	Column1046	Column1047	Column1048	Column1049	Column1050	Column1051	Column1052	Column1053	Column1054	Column1055	Column1056	Column1057	Column1058	Column1059	Column1060	Column1061	Column1062	Column1063	Column1064	Column1065	Column1066	Column1067	Column1068	Column1069	Column1070	Column1071	Column1072	Column1073	Column1074	Column1075	Column1076	Column1077	Column1078	Column1079	Column1080	Column1081	Column1082	Column1083	Column1084	Column1085	Column1086	Column1087	Column1088	Column1089	Column1090	Column1091	Column1092	Column1093	Column1094	Column1095	Column1096	Column1097	Column1098	Column1099	Column1100	Column1101	Column1102	Column1103	Column1104	Column1105	Column1106	Column1107	Column1108	Column1109	Column1110	Column1111	Column1112	Column1113	Column1114	Column1115	Column1116	Column1117	Column1118	Column1119	Column1120	Column1121	Column1122	Column1123	Column1124	Column1125	Column1

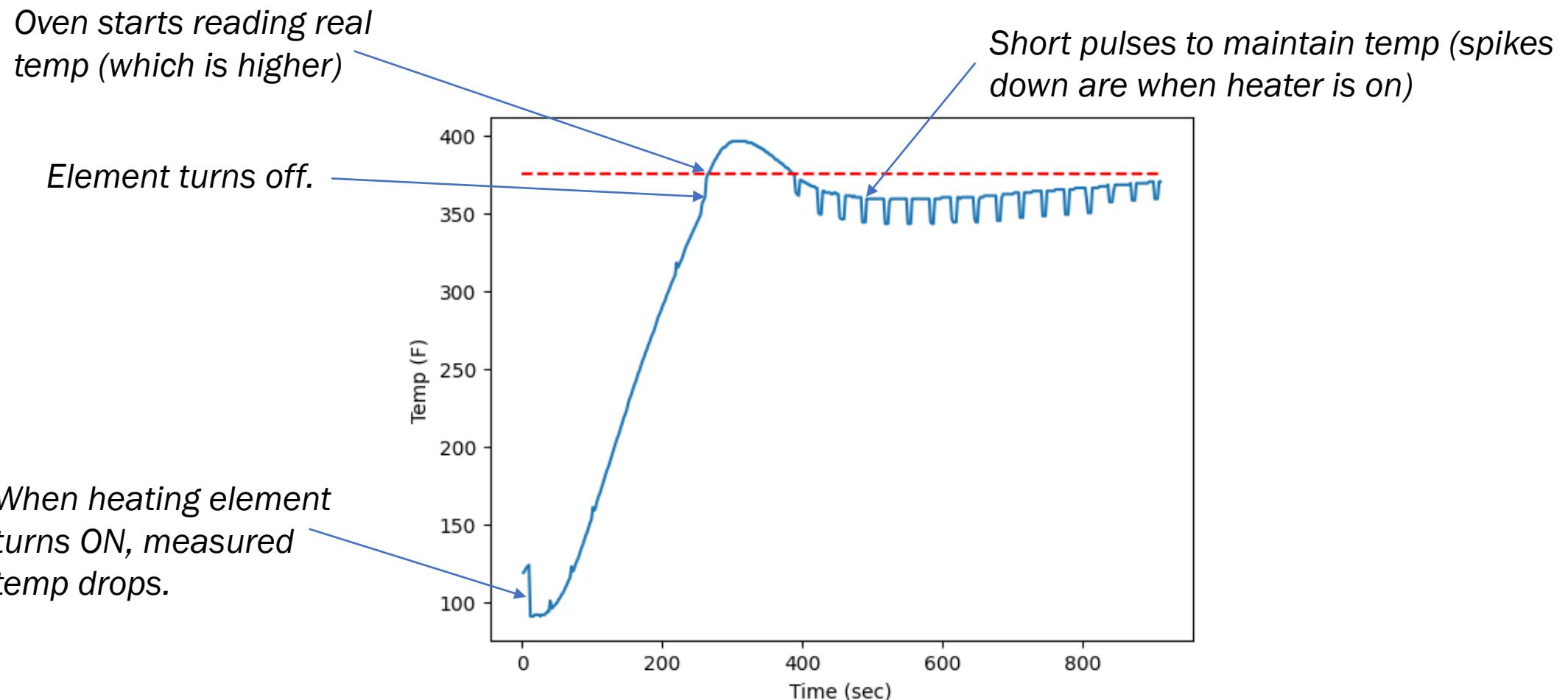
Examples of Global Variables

0x1248 = Top Temp in F

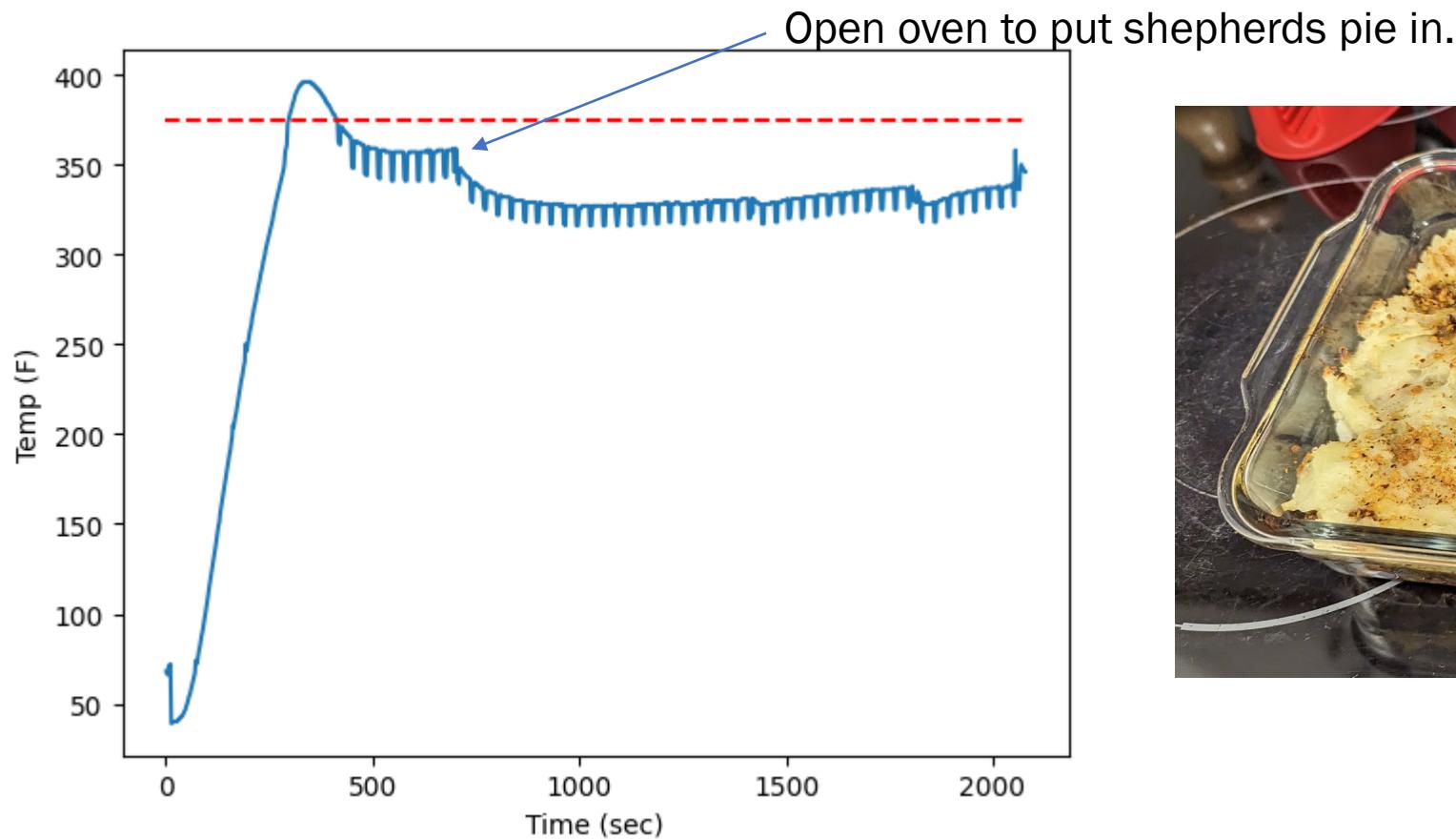
0x120a = Heater “ON” Flag



Set 375F, Cold Start, No Load



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



Observed Display Logic During Pre-Heat

if temp < 150F:

display(150F)

elif t

else:

display(temp)

old_temp = temp

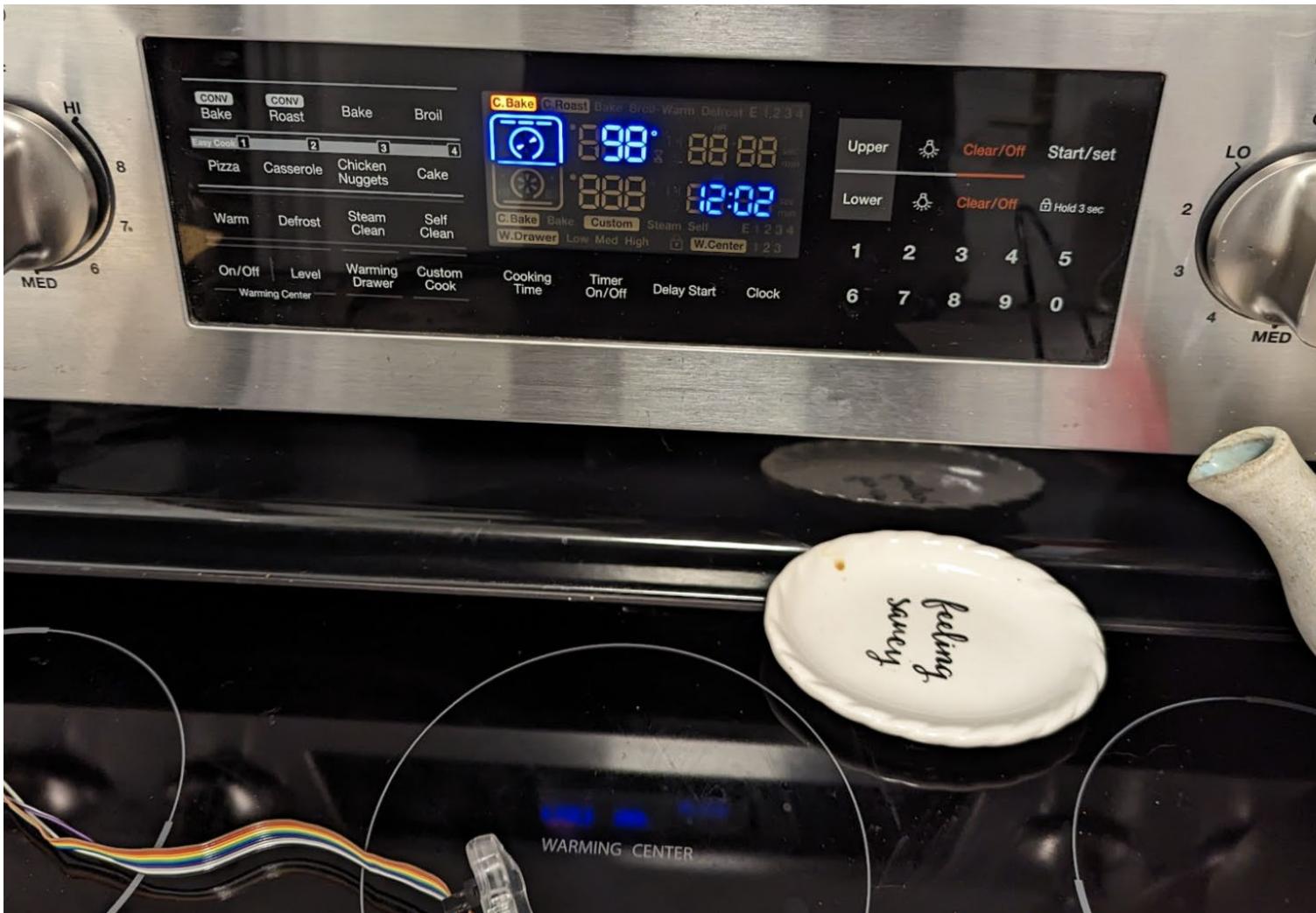
9214	UXfe8858	68 03	JK	UXUF-E885D		
9215	0xfe885a	33 96 00	LDW	HL,0x96	0x96 = 150F	
9216	0xfe885d	db f4	CPW	IX,HL		
9217	0xfe885f	67 4a	JR	C,0x0FE88AB	Tested	#patch(0xfe885f, "68") #-- Displays 150F on
9218	0xfe8861	d2 a8 12 00 fb	CPW	(0x12A8),HL		
9219	0xfe8866	67 3c	JR	C,0x0FE88A4	Tested	#patch(0xfe8866, "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 $\text{temp} > \text{set_temp}$, 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:

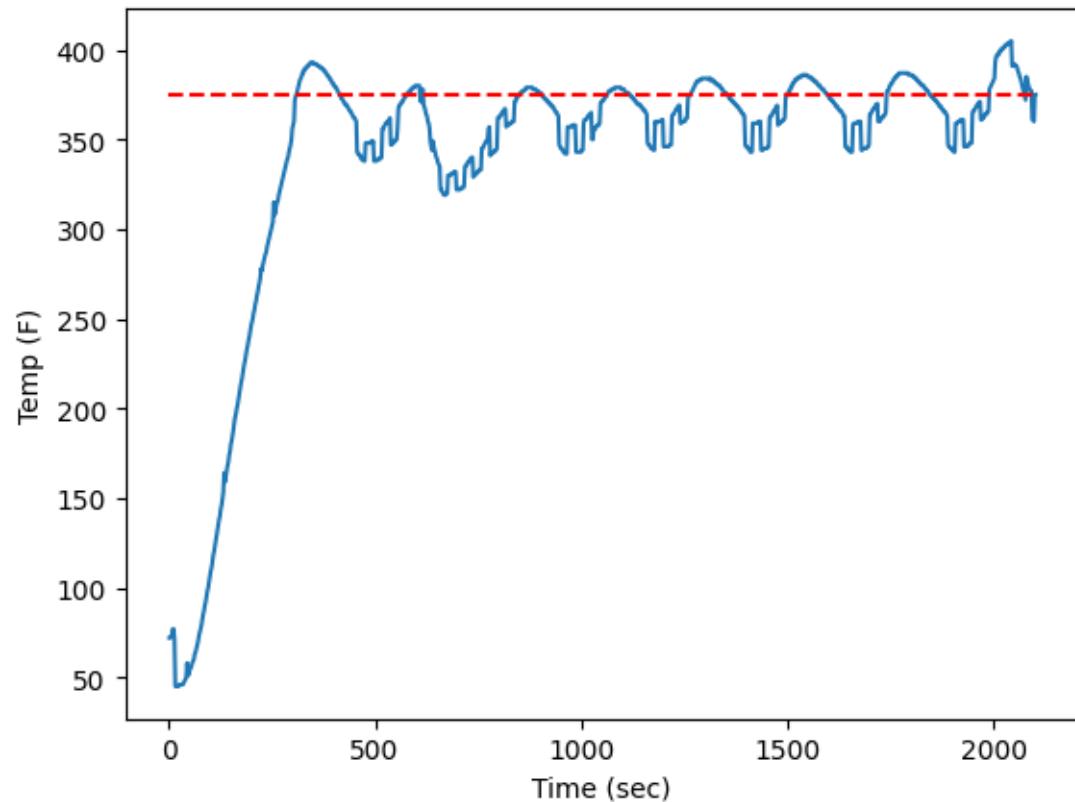
heater(off)

display(temp)

Op	OpName	OpDesc	OpType	OpValue
JR	NC,0x0FE889D	Tested		
LDW	WA,IX			
SUBW	WA,(0x12A8)			
CPW	WA,0x2	From fe885f, insert:		
JR	ULE,0x0FE8894	Tested	LDW HL, IX	dc 8b
CPW	IX,(0x12A8)		LDW (0x12A8),HL	f2 a8 12 00 53
JR	ULE,0x0FE889D	Tested	BIT 1, (0x120A)	f1 0a 12 c8
BITB	0x6,(0x11CC)		JR NZ, 0x0FE88B0	66 44
JR	Z,0x0FE88D	Tested	ADDW HL, 11	db c8 0b 00
INCW	0x2,(0x12A8)		JR 0x0FE88B0	68 3e
LDW	HL,(0x12A8)			
JR	0x0FE88B0		<-Patch to jump here from fe885f	
LDW	HL,IX			
LDW	(0x12A8),HL			
JR	0x0FE88B0			

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>

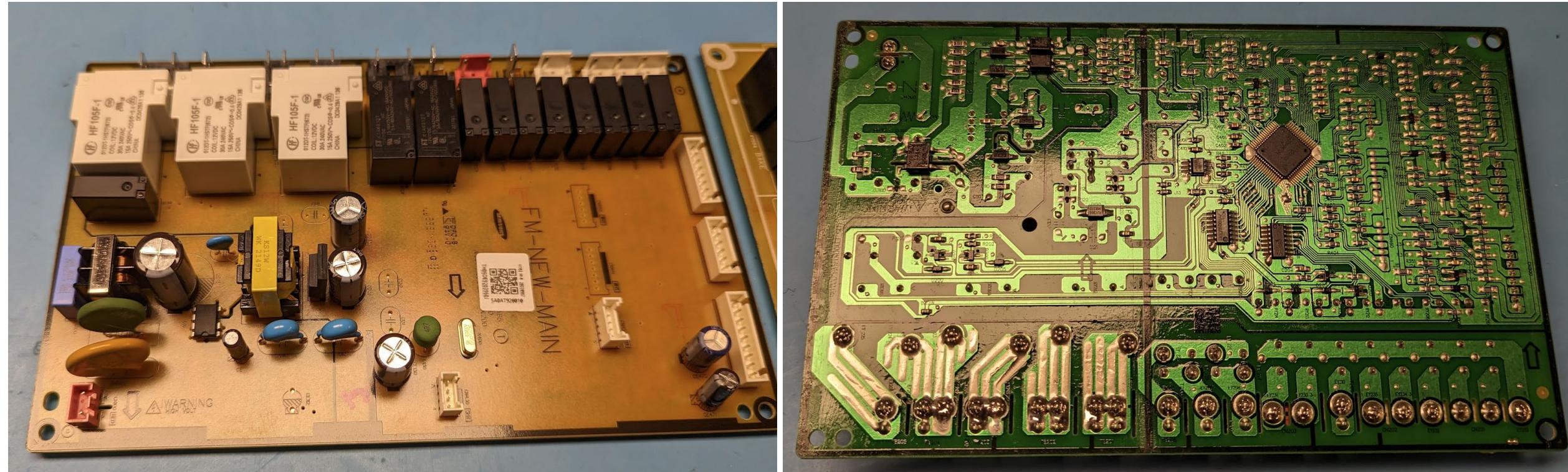
Known Bugs



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:

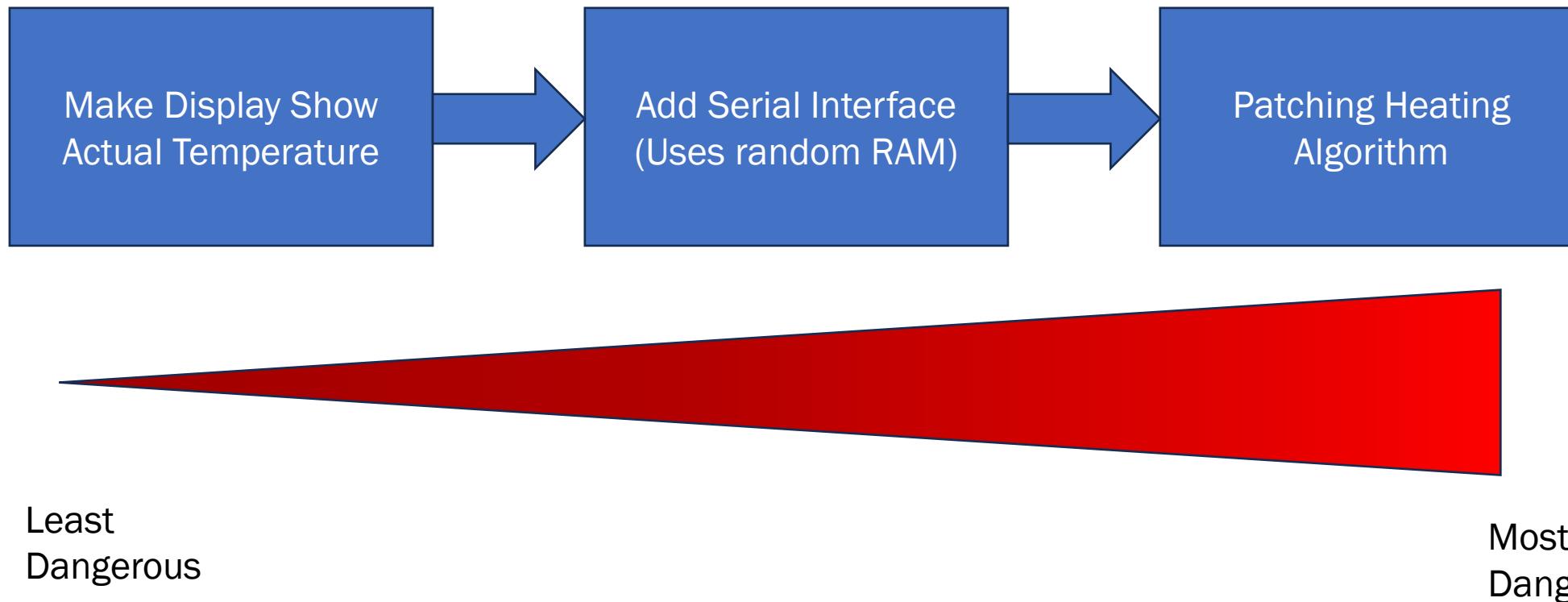


- “Newer” ovens based on R5F100LEAFB#V0 (RL78/G13)
- No protection (can read-out with debugger)
- Supported Ghidra plugin!

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 μ Art, <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



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 receipts & 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

