Adventures of my Oven (Pinocchio) & ChipWhisperer



Colin O'Flynn



RECON Montreal - June 9, 2023. Presented by Colin O'Flynn.



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

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.

Pars

77 comments 😑

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





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



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.

Operation Command Data	Operation Mode	
10H	RAM Transfer	
20H	Flash Memory SUM	
30H	Product Information Read	\frown
40H	Flash Memory Chip Erase	
60H	Flash Memory Protect Set	12
		(\bigcirc)

 Table 14-7 Operation Command Data

	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		(\bigcirc)
	18th byte	-		ACK response to CHECKSUM value#2 Normal 10H Error 11H Communications error 18H
	19th byte	RAM storage start address 31 to 24 treal	June 9, 2023. Pres	ented by Colin O'Flynn.
	20th byte	PAM storage start address 22 to 16#4		

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 (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		R
	18th byte	-		ACK response to checksum value ^{#2} Normal 60H Error 61H Communications 68H
	1			ACK response to Brotest Set command

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

Important Take-Aways (for next part)

- 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" \rightarrow 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 Condition: New - Open box "New item in Good Condition" Last One / 1 sold Price: US \$280.00 **Buy another** Add to cart

Best Offer:

Si oven revd - TOSHIBA Integrated	Development Enviro	nment	4			-01	
File Edit View Project Build Debug	Tool Window Help	_ _ _ _					
						l. I	
	NISE IEL					1	
	7 57 EB						
				-			
							<u>_</u>
	Hived Mode						
	Start Address : Dxff	41e3	.				
	00ff41e3 PUS	HL XIZ					
	00ff41e4 PUS 00ff41e5 PUS	HL XIY HL XIX					
	00FF41e6 PUS	HL XHL HL XDE				le de la companya de	
	00ff41e8 PUS 00ff41e9 PUS	HL XBC HL XWA					
	00ff41ea CALI AAff41ee POP	L 0x0FF3C01 XWA					
	00ff41ef POP	L XBC I XDF					
	00FF41F1 POP						
•	00ff41f3 POP						
	00ff41f5 RET	I					1. A.
	00ff41f7 RET	I					htt:
	00ff41fa LDB	(0x300),0x60					
	00ff4204 LDL	XSP, 0x2FFF					
	00ff4209 LDL 00ff420e LDL	XDE,0X1000 XBC,0X2000					
	00FF4213 LDW 00FF4215 SRL					· · · · · · · · · · · · · · · · · · ·	
	ANFF4218 JR	Z,0X0FF4238 XHI_XDF			<u>•</u>	1	
Address(A): 0xffff00	I 2 4 €	▙ <u>┦▶ ≫ ♂</u> ₽ ★ ★			X 🔹 Current	02-0100:Conne 🔺	
0xffff00:00ff41f8 00ff41f	6 .AA 🦉	0x00fe885f: LDW	HL,IX (0v1208) HI	dc 8b	XWA : 0x 000000c I-1 XBC : 0x 00000210	02-0400:Restc 01-0D00:Anali	
0xffff10:00ff41f6 00ff41f	6 .AA	0x00fe8866: bit	0, (0x120A)	f1 Øa 12 c8	XDE :0x0000e70 I-1 XHI :0x0000001 I-1	01-1000:Down] 01-1400:Objec	
0xffff20:00ff41f6 00ff41f	8 .AA 6 b@ A	0x00fe886c: addw	hl, 11	db c8 0b 00	XIX :0x0000015 I-1	17-0103:SIMME 17-0103:SIMME	
0xffff30:00ff41f6 00ff41f	6 .AA	0x00fe8872: ???	0710000	00 30	XIZ :0x0000000 I-1	17-0103:SIMME 17-0103:SIMME	
0xffff40:00ff41f6 00ff41f	6 .AA				INTNEST: 0xe70 I-1	17-0103:SIMME 17-0103:SIMME	
0xffff50:00ff41f6 00ff41f	6 .AA				PC :0x00ffe04 I-1	17-0103:SIMME 17-0103:SIMME	
0xffff60:00ff40f8 00ff41f	6 .@A				IFF : 0x I-1	17-0103:SIMME 17-0103:SIMME	こしてい
0xffff70:00ff41f6 00ff412	4 .A\$A				RFP : 0x E-1	05-6400:Cannc 05-6400:Cannc	
0xffff80:00ff41f6 00ff41f	6 .AA				ZF : 0x E-1	05-6400:Cannc 05-6500:Cannc	
0xffff90:00ff41f6 00ff41f	6 .AA				UF : 0x E-1	05-6400:Cannc 05-6400:Cannc	
0xffffa0:00ff41f6 00ff41f	6 .AA 6 .AA				NH : UX CF : 0X E-1	05-6400:Cannc	
0xffffa8:00ff41e3 00ff41d 0xffffb0:00ffdf40 00ffe00	U .AA 0@						
0xffffb8:00ff41f6 00ff41f 0xffffc0:00ff41f6 00ff41f	6 .AA 6 .AA						
0xffffc8:00ff41f6 00ff41f 0xffffd0:00ff41f6 00ff41f	6 .AA 6 .AA =						
Beady	6 .AA 🖻			RECON Montrea	arrigune 9, 2023	uesented k	by Collin O'Flynn.
start Soven revd - TOSH	IBA 🕝 Desktop	Control Panel	C:\WINDOWS\syste			 \$ 100 8:03 PM 	

Windows XP?



14

Can you Read Back Bootloader?



Connected

Target CRC: ---

MCU: TMP91FY42 BOOT COM3, 38400 Baud

Segger "ToshLoad" can readback 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 $0 \times 0 FFF2D7$ 00fff2d5 LDB L,0x1 <-- L is flag, if set to 1 comparison failed 00fff2d7 LDW BC, $0 \times 0 C < -- 12$ bytes to compare 00fff2da LDL XIX, (0x0FFF00C) <-- Points to 0004FEF4 (PW) 00fff2df LDB RH1, 0x000fff2e2 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 < -- checksum00fff2f4 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.



ChipWhisperer-Husky Intro



Power Analysis?



Easy-Mode Level 1: Password Power Analysis





Difference Between Guessed Power Trace & Mean

Fault Injection?

FUNCTION	START:	RAM WRI	TE FUNC	CTION			
00fff2f5	CALR	0x0FFF	75F <-	Load	protection	n status	
00fff2f8	CPB	A,0x0E	FF < (Compare	protection	n status	
00fff2fb	JR	NZ, 0xC)FFF290	< Se	nd error i	f protection	enabled
00fff2fd	CALR	0x0FFE	72A2 <	- PW Ch	eck	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·



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)	C	<u> </u>
	4th byte	-		ACK response to CHECKSUM value ^{#2} Normal 20H Error x1H Communications error x8H
	5th byte	-	(7)	SUM (upper)
	6th byte	<u> </u>		SUM (lower)
	7th byte	-	\bigcirc	CHECKSUM value for 5th and 6th bytes
	8th byte	(Wait for the next operation command data)		

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

#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).

Fault Injection Setup / Demo

In [52]: ▶ reset target()

responsehex

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

response, responsehex = tx rx(b"\x86", 1, 1)

response, responsehex = tx rx($b'' \times 20''$, 4)

if responsehex[0] != 0x86:

raise IOError("Sync Error")

broken = False for glitch set

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.giitch.winch = 2000 #100 #1000	
<pre>for glitch_settings in gc.glitch_values(): scope.glitch.ext_offset = glitch_settings[0] for i in range(sample_size): reset_target()</pre>	
<pre>target.ser.flush() response, responsehex = tx_rx(b"\x86", 1, 1) if responsehex[0] != 0x86: raise IOError("Sync Error")</pre>	
<pre>scope.arm()</pre>	In [59]: ▶ known_pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44]
#Do glitch loop target.ser.write(b"\x10")	<pre>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)</pre>
<pre>ret = scope.capture()</pre>	
<pre>if ret: print('Timeout - no trigger') gc.add("reset")</pre>	<pre>In [60]: ▶ #Print the password (should match the known one) time.sleep(0.1)</pre>
<pre>#Device is slow to boot? reset_target()</pre>	<pre>data = rl.cmd_read(0x02FEF4, 12) ':'.join(hex(ord(char)) for char in data)</pre>
else:	Out[60]: '0xde:0xad:0xbe:0xef:0xca:0xfe:0xfa:0xce:0x11:0x22:0x33:0x44'
response = target.ser.read(1) response = [ord(i) for i in response]	In [12]. N #Read the full flash itself
<pre>if len(response) == 0: gc.add("reset") else:</pre>	#TMP91FW27UG in Single Boot Mode - flash is from 0x10000 to 0x30000 (starts @ 0x10000, length = 0x20000) flash = rl.cmd_read(0x10000, 0x20000)
<pre>if response[0] != 0x16:</pre>	In [13]: 🔰 len(flash)
<pre>#broken = True gc.add("success") print(response)</pre>	Out[13]: 131072
<pre>print(hex(response[0])) print(scope.glitch.ext offset)</pre>	In []: ▶ known_pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44]
<pre>print("\$", end="") if response[0] == 0x10: broken=True break</pre>	<pre>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)</pre>
#break	
else: gc.add("normal") if broken: break	
[16] 0×10	

8015

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.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))</pre>

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





2023. Presented by Colin O'Flynn.

Medium-Mode Level 2: Fault Injection

0x87 11710 🜲 [133] 0x85 11715 鷠 [17] 0x11 11750 **a** [16] 0x10 11755 • ▶ #known pw = [0xDE, 0xAD, 0xBE, 0xEF, 0xCA, 0xFE, 0xFA, 0xCE, 0x11, 0x22, 0x33, 0x44] In [59]: 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)



\$\$ → Samsung Parts Department



Did they have problems with returns?



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	80	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	òòðÂZ.
0070h:	00	ЗF	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	ЗA	14	00	3F	01	6E	0E	.òðÂ:?.n.
0090h:	F2	ЗA	14	00	00	00	F2	5C	12	00	00	03	68	26	F2	ЗA	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.òXòlòn
00C0h:	12	00	00	00	F2	F0	10	00	00	01	F2	58	14	00	00	00	òðòX
00D0h:	0E	8F	04	23	C2	6A	11	00	3F	0 A	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:	ЗF	05	66	06	F2	5C	12	00	00	03	CB	89	D8	12	D8	09	?.f.ò\ˉØ.Ø.
0100h:	09	00	F2	6A	11	00	32	F3	07	E8	E0	00	00	CB	89	D8	òj2ó.èàˉØ
0110h:	12	D8	09	09	00	F2	6B	11	00	32	F3	07	E8	E0	00	00	.Øòk2ó.èà
0120h:	CB	89	D8	12	D8	09	09	00	F2	68	11	00	32	F3	07	E8	E‰Ø.Øòh2ó.è
0130h:	E0	00	00	CB	89	D8	12	D8	09	09	00	F2	69	11	00	32	àE‰Ø.Øòi2
0140h:	F3	07	E8	E0	00	00	CB	89	D8	12	D8	09	09	00	F2	6C	ó.èàE‰Ø.Øòl
0150h:	11	00	32	F3	07	E8	E0	00	00	CB	89	D8	12	D8	09	09	2ó.èàE‰Ø.Ø
0160h:	00	F2	6D	11	00	32	F3	07	E8	E0	00	00	CB	89	D8	12	.óm2ó.éáE‰Ø.
0170h:	D8	09	09	00	F2	6E	11	00	32	F3	07	E8	E0	00	00	CB	Øòn2ó.èàE
0180h:	89	D8	12	D8	09	09	00	F2	6F	11	00	32	F3	07	E8	E0	‰Ø.Øòo2ó.èà
0190h:	02	00	00	CB	89	D8	12	F2	98	12	00	32	F3	07	E8	E0	E‰Ø.ŏ~2ŏ.ė́a
01A0h:	00	00	CB	89	D8	12	F2	34	14	00	32	F3	07	E8	EO	00	E‰Ø.ó42ó.éā.
01B0h:	00	CB	89	D8	12	D8	80	F2	FE	13	00	32	F3	07	E8	E0	.E‰Ø.Ø€op2o.ea
OICON:	02	00	00	C2	64	11	00	3F	00	6E	12	F2	A6	12	00	00	Ad?.n.o¦
	00	FZ	A/	12	00	00	00	FZ	A8	12	00	00	00	CB	08	66	.030EØT
01EUN:	04	CB	09	0E	14	FZ	00	12	00	00	00	FZ	DA	12	00	00	.EUN.OU
01F011	00	12	08	12	00	00	00	68	00	FZ	08	12	00	00	00	F2	.onn.ovo
020011	DC	12	00	00	00	FZ	Eð	13	00	10	10	UE	FZ	00	11	00	U0e0T
021011.	10	00	F2	04	11	00	10	00	F2	1E	12	50	10	00	F2	90	
022011.	T2	66	10	00	F2	90	12	50	64	10	F2	DE	12	00	50	00	0@0^
0230H.	12	00	00	00	52	10	12	F2	04	12	00	U2 E2	00	12	F2	80	٥١٥u٥u
024011	12	50	54	11	F2	18	12	00	D2	20	11	F2	02	12	00	50	ÀT ÀO À
0260h	30	11	00	00	00	02	00	00	10	50	42	EE	55	62	E2	05	.01000
0270h	12	00	00	00	E2	40	12	00	00	00	F2	42	12	02	00	00_	à àc
0280h	F2	20	12	00	00	00	C2	EQ	12	00	26	00	66	06	E2	EA	à Âg 7 f àú
02000	12	00	00	01	E2	40	14	00	00	00	51	22	12	00	02	00	à@ à2



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).

Reverse Engineering Tools



GHIDRA

	State-of-the-art binary code analysis tools												
hex-rays		Products >	Solutions	Partners	Shop	Support >	Company >						
	IDA Pro	Philosophy	Disassembler	Debugger	Learn & Su	pport	Buy a license						
				IDA The best-of- malware and	Pro breed binary cr alyst and cybers	ode analysis to security profes	ol, an indispensable item in the						

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 (assemb) 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-readule.

man	<u> </u>											
0413980			09150800	00110000	00120830		00(3300)	**:::***	01138880	00130030	80118081	
0413484		00110000	00110000	00110000	001200N		00130000		00130001	00130803	80110080	
0013000			04150801	00110000	00120000	00110000	00133000	**:::***	00130081	00130830	00110001	
0013000	1 00310000	00138081	09130800	00110000	00130830	8111000	00(3304)	10130804	00138981	00130800	80318061	
0413080	-	0053000	00110803	00518006	NET CORNE	BUILDORD	00533083	P0124800	ACC 38985	0011080	80518085	
0012000	00110000	00530003	00110000	00110000	00120000	BREEDONG (00533000	00228900	00538980	00130696	00510001	. 00
0013000	1 00330000	0013000	09150801	00218000	00120990	80110001	00(33000	00120000	01138881	00130000	#0118081	
PE13081	rs +++++	00132000	08130800	00518085	00520930	-	00533999		****	04130636	#0518085	
00123000	0 00210000	00530003	00110000	00518003	00520000	00110001	00533000	84558866	00530000	00130803	89518085	
0012000	1 00220000	10153000	09120800	00110001	00220000	essees	00120002	01228908	101231025	0112010	00120001	
0000121	100430001	49114465	00110000	000000	80110000	0003000	0033000	NUMBER	10110001	00118000	90002150	
		00139081		00110001	00120830	-			44138884	0013080	85118081	
04905.00		-		00110003	NUMBER	RELINGED	-	MULTING	00120001	00130000	#01100HD	
								12000	DECOMP			4
-								100				
-	Inc. Mark				- 00	1.000			6400	104110		1.
	Lab Amer	******						100		(shrute)	Auril 10	
-			w Le	and the second	an), 15	Tant annu	10.1		-			
	"untie"								_		NK + 2/	
										(attrated	Aurol Blu	
										1 strate	Auril 10	
			inter all							strate	Auril 10	



AutoSave	● off 📙 🖌) →	~	2	Search							Colir	n O'Flynn	•	(ž	-		×
File Home	e Insert Drav	w Page Layout Formulas	Data Review	/iew Help	Acrobat	Table Des	sign Query	/							모 Comn	nents 🛛	3 Share	v
Paste	Calibri B I U 、	$ \begin{array}{c c} & & \\ \hline \\ \hline$	= <u></u> ≫ ~ e = = ± ± ±	Wrap Text	nter ~	ieneral \$~%	(√ 00. 00. 00. 00. 00. 00.	Condition Formatting	al Formata y Table ~	s Cell Styles ~	Insert	Delete For	rmat ×	Sort &	Find & Select ~	Analyze Data		
Clipboard F	ž.	Font 🛛	Alignment		L2	Numbe	r L		Styles			Cells		Editing		Analysis		~
F9931 V	$i \times \sqrt{f_x}$																	~
(Column2)	Column2	Column 4	Caluman	Calum	Column					K			N	0	D	0		
9926 0xte9681	d8 12	EXTZW WA	▼ Columns	- Columi	• Column •	G	П		J	ĸ	L	IVI	IN	0	٢	Q	H	<u> </u>
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 routi	ne?														
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									புβITB Ox	3,(0x209)					
9933 0xfe969b	66 Ob	JR Z,0x0FE96A8										JR Z,HE	ERE					
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 (0)	(1120),0x0					
9937 0xfe96a7	0e	RET									HERE	BITB Ox	2,(0x209)					
9938 0xfe96a8	f1 09 02 ca	BITB 0x2,(0x209)										JR Z,HE	RE2					
9939 Oxfe96ac	66 Ob	JR Z,0x0FE96B9										NOP						
9940 0xfe96ae	00	NOP										LDB A,(0x208)					
9941 0xfe96af	c1 08 02 21	LDB A,(0x208)										LDB (0)	(1120),0x0					
9942 0xfe96b3	f2 00 11 00 41	LDB (0x1100),A									HERE2	BITB Ox	4,(0x209)					
9943 0xfe96b8	0e	RET										JR Z,HE	ERE3					
9944 0xfe96b9	f1 09 02 cc	BITB 0x4,(0x209)										NOP						
9945 0xfe96bd	66 Ob	JR Z,0x0FE96CA										LDB A,(0x208)					
9946 0xfe96bf	00	NOP										LDB (0)	(1120),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	4					
9949 0xfe96c9	0e	RET										EXTZW E	зс					
9950 0xfe96ca	c2 20 11 00 21	LDB A,(0x1120)	What is 1120??									LDAL X	DE,0x1110					
9951 0xfe96cf	c9 8b	LDB C,A										LDB A,(0x208)					
9952 0xfe96d1	d9 12	EXTZW BC	BC has (0x1120)	data								LDB (XI	DE+BC),A					
9953 0xfe96d3	f2 10 11 00 32	LDAL XDE,0x1110										CPB (0)	(1120),0x0C					
9954 0xfe96d8	c1 08 02 21	LDB A,(0x208)	RX Byte									RET UL	E					
9955 0xfe96dc	f3 07 e8 e4 41	LDB (XDE+BC),A	Load byte here?									LDB (0)	(1120),0x0					
9956 0xfe96e1	c2 20 11 00 3f	00 CPB (0x1120),0x0										CALR 0	OFE956F					
9957 0xfe96e7	6e 10	JR NZ,0x0FE96F9	Fail I guess?				0	x1110				RET						-
< >	DE92-02439	F FW Disassembly Sheet	RECON	Iontreal	June 9,	2023. 1	Presente	d by Co	in D 'Flyr	nn.						_	42	Þ

🕁 Display Settings 🔠 🗉 🖳 – — — + 100%

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.



OK, Just I

In [7]: ▶ bl = t]
#bl.cmd
bl.cmd
Bl.cmd
Read: n



\$\$\$ → Samsung Parts Department



RECON Montreal - June 9, 2023. Presented by Colin O'Flynn.

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...



RECON Montreal - June 9, 2023. Presented by Colin O'Flynn.

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

Checksums Differ!

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

...Add the Serial Monitor

AutoSave 🤅	• off) 🗄 りー		ven REVD.xlsx $ \mathbf{v} $			∕⊃ Searc	h								C	olin O'Fly	nn co	\Leftrightarrow	(3°	-		\times
File Home	Insert Draw I	Page Layout Fo	ormulas Data	Review	View	Help A	crobat	Table I	Design	Query	(₽ Com	ments	🖻 Shar	e ~
Paste ✓ ✓ ✓	Calibri ~) <u>11</u> ~ A^ A`] ~ <u>A</u> ~ <u>A</u> ~) & ~ = =	<mark>き</mark> 》Wrap	o Text je & Center	~ \$	eneral 5 ~ %	9 €00	ب 00.	Conditional Formatting ~	Format as Table ~	Cell Styles ~	Insert	Delete	Format	∑ ~ ↓ ~ ∳ ~	A Z Sort & F Filter ~ S	-ind & elect ~	Analyze Data		
Clipboard 🕞	Font		Гъ	Alignm	nent		Гъ	Nur	nber	٦		Styles			Cells			Editing		Analysis		~
SECURITY W	VARNING External Dat	a Connections have	e been disabled	Enable	e Content																	×
H124388 ~	$: \times \checkmark f_x$ R	< Interrupt																				~
ın2	- Column3	- Colur	mn4		-	Column5		-	Column	16	- Colum	า7		- Col	umn 🗸	Colun	n 🚽	1		J	К	
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 Int	errup	t				
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 E	0 FF	CALL 0	KOFFEO	50									
124396 08	ff	SWI	0x7						58		POPL	XWA										
124397 09	ff	SWI	0x7						59		POPL	ХВС										
124398 0a	ff	SWI	0x7						5a		POPL	XDE										
124399 Ob	ff	SWI	0x7						5b		POPL	XHL										
< >	DE92-02439D FW	/ Disassembly	Sheet1	+							:	-	_	_	_	_	_	_	_	D		•
Ready 🔀 Accessib	pility: Investigate													📮 Dis	play Setti	ngs E		1 2			+ 1	23%

Slight risk of overwriting something else important....

RECON Montreal - June 9, 2023. Presented by Colin O'Flynn.

Examples of Global Variables

0x1248 = Top Temp in F 0x120a = Heater "ON" Flag



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





Patched Display Logic



RECON Montreal - June 9, 2023. Presented by Colin O'Flynn.

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			P 💱 🖉 💱 🖉 X								
SUBW WA,(0x12A8)			0x00fe885f: LDW 0x00fe8861: LDW	HL,IX (0x12A8), HL	dc 8b f2 a8 12 00 53						
CPW WA,0x2		From fe885f, insert:	0x00fe8866: bit	0, (0x120A)	f1 0a 12 c8						
JR ULE,0x0FE8894	Tested	LDW HL, IX	0x00fe886a: jr 0x00fe886c: addw	z, 0xfe8800 hl, 11	00 44 db c8 9b 99						
CPW IX,(0x12A8)		LDW (0x12A8),HL	0x00fe8870: jr 0x00fe8872: 222	0xfe88b0	68 3e						
JR ULE,0x0FE888D	Tested	BIT 1, (0x120A)	58551200121								
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 her<="" jump="" td="" to=""><td>e from fe885f</td><td>db c8 0b 00</td><td></td></patch>	e from fe885f	db c8 0b 00								
LDW (0x12A8),HL			68 3e								
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







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

RECON Montreal - June 9, 2023. Presented by Colin O'Flynn.

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.

Playing with Your Own Oven



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.
- <u>Lots</u> 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@bluenoser.me Hellsite: @colinoflynn