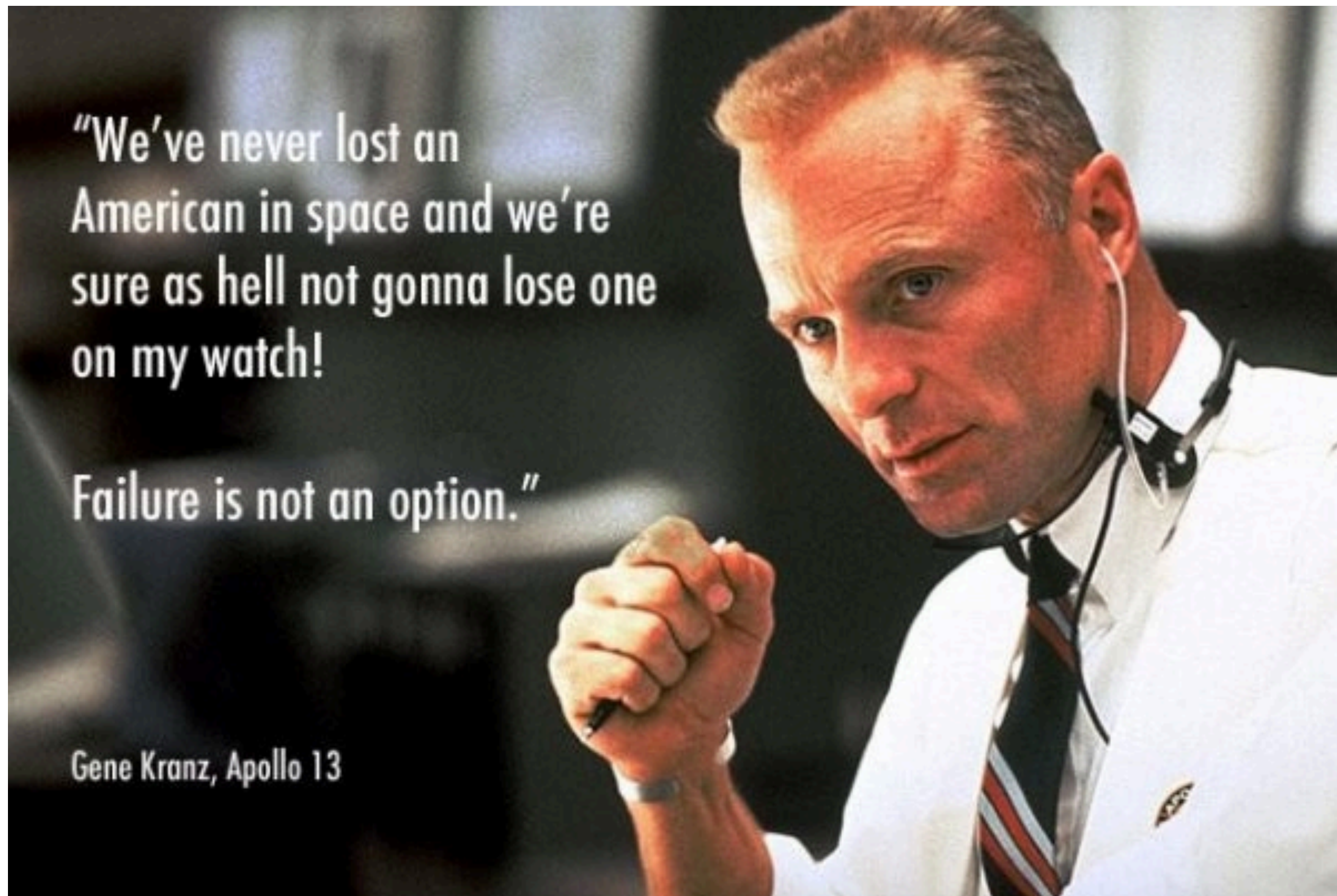


Failure Is Not an Option

**A RELIABLE PROCESS TO EXPLOIT STM32F2/F4
MICROCONTROLLERS**

Joe Grand + Lennert Wouters
REcon 2026, Montreal, Canada

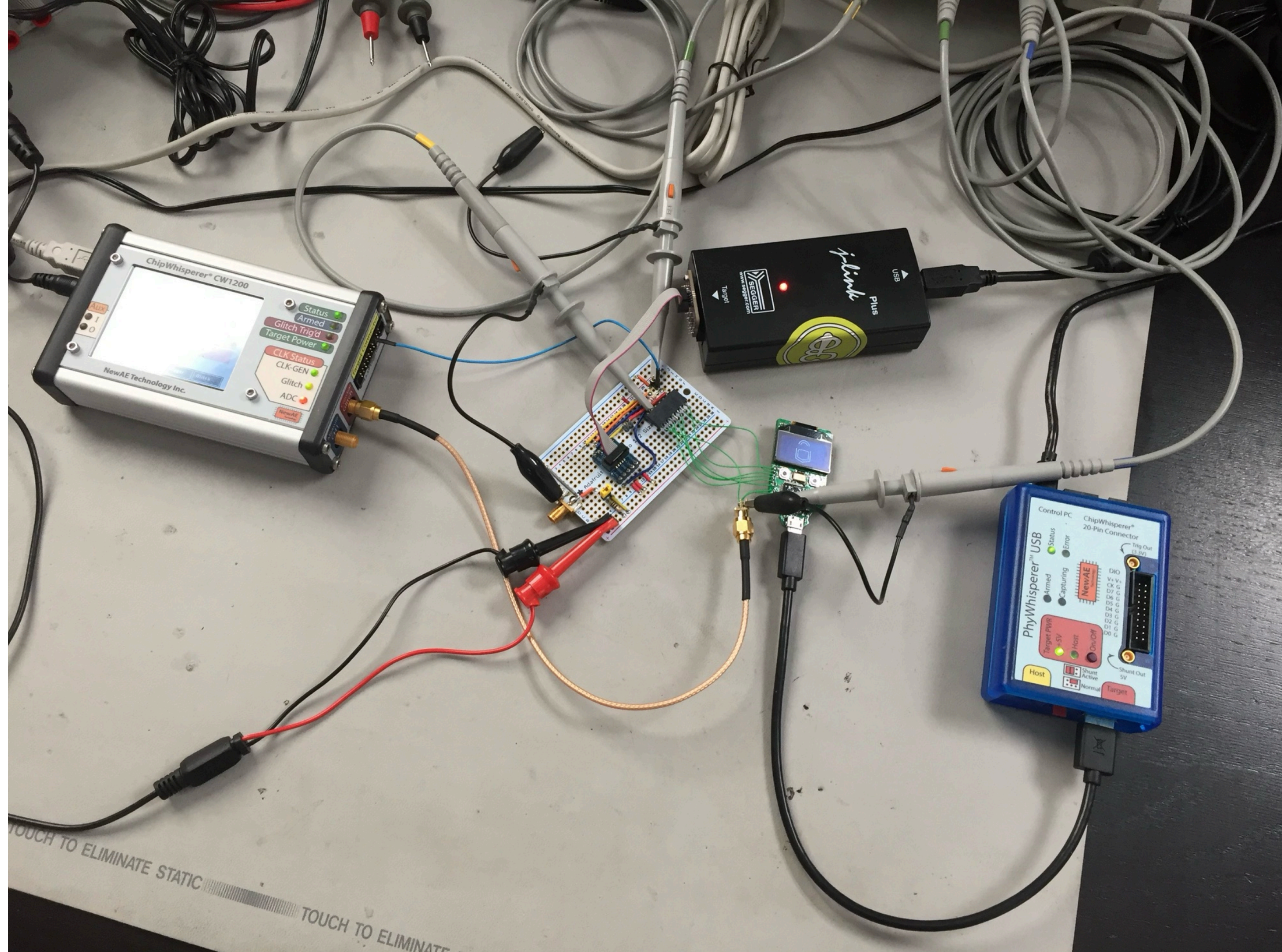


** Unproven that he actually said this

IT ALL STARTED WITH AN E-MAIL

"I'm reaching out because of an issue I have with my trezor wallet. In 2017, I bought some crypto and haven't touched the wallet since. When I was moving earlier this year, I think that I accidentally threw out the recovery seed."

-Dan





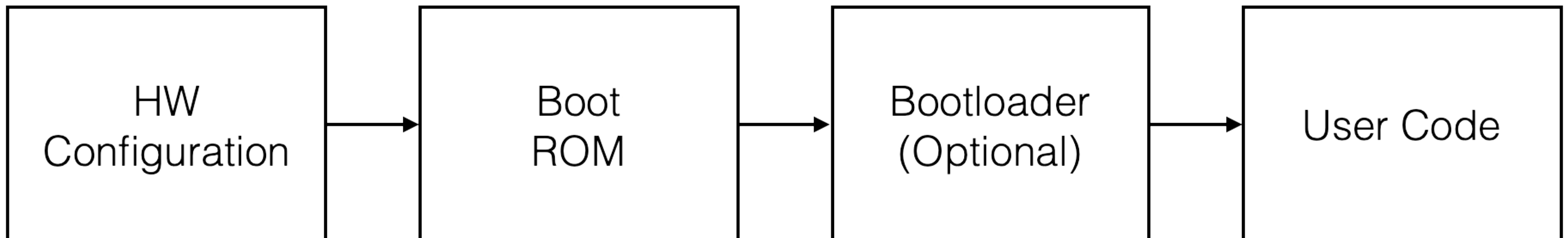
LENNERT AND JOE MEET

"I'm a bit late to the party, but I started playing a bit with FI on the STM32 series to bypass RDP."

–Lennert, January 4, 2023

MICROCONTROLLER (MCU) SECURITY 101

- MCUs have internal memories that may contain secrets (e.g. firmware / IP or key material)
- Vendor-specific implementations try to prevent adversaries from reading those secrets
 - May require fuse/register setting, password, challenge/response
 - Reduce access (allow subset of functionality)
 - "Permanently" disable access
- Configured/checked during chip boot process

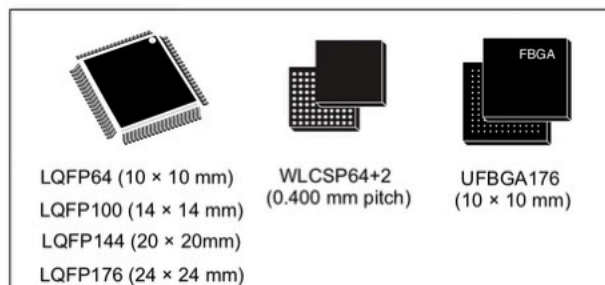


Arm®-based 32-bit MCU, 150 DMIPs, up to 1 MB Flash/128+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces and camera

Datasheet - production data

Features

- Core: Arm® 32-bit Cortex®-M3 CPU (120 MHz max) with Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution performance from Flash memory, MPU, 150 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1)
- Memories
 - Up to 1 Mbyte of Flash memory
 - 512 bytes of OTP memory
 - Up to 128 + 4 Kbytes of SRAM
 - Flexible static memory controller that supports Compact Flash, SRAM, PSRAM, NOR and NAND memories
 - LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
 - From 1.8 to 3.6 V application supply + I/Os
 - POR, PDR, PVD and BOR
 - 4 to 26 MHz crystal oscillator
 - Internal 16 MHz factory-trimmed RC
 - 32 kHz oscillator for RTC with calibration
 - Internal 32 kHz RC with calibration
- Low-power modes
 - Sleep, Stop and Standby modes
 - V_{BAT} supply for RTC, 20 × 32 bit backup registers, and optional 4 Kbytes backup SRAM
- 3 × 12-bit, 0.5 μs ADCs with up to 24 channels and up to 6 MSPS in triple interleaved mode
- 2 × 12-bit D/A converters
- General-purpose DMA: 16-stream controller with centralized FIFOs and burst support
- Up to 17 timers
 - Up to twelve 16-bit and two 32-bit timers, up to 120 MHz, each with up to four IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Debug mode: Serial wire debug (SWD), JTAG, and Cortex®-M3 Embedded Trace Macrocell™
- Up to 140 I/O ports with interrupt capability:
 - Up to 136 fast I/Os up to 60 MHz
 - Up to 138 5 V-tolerant I/Os
- Up to 15 communication interfaces
 - Up to three I²C interfaces (SMBus/PMBus)
 - Up to four USARTs and two UARTs (7.5 Mbit/s, ISO 7816 interface, LIN, IrDA, modem control)
 - Up to three SPIs (30 Mbit/s), two with muxed I²S to achieve audio class accuracy via audio PLL or external PLL
 - 2 × CAN interfaces (2.0B Active)
 - SDIO interface
- Advanced connectivity
 - USB 2.0 full-speed device/host/OTG controller with on-chip PHY
 - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
 - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII
- 8- to 14-bit parallel camera interface (48 Mbyte/s max.)
- CRC calculation unit
- 96-bit unique ID

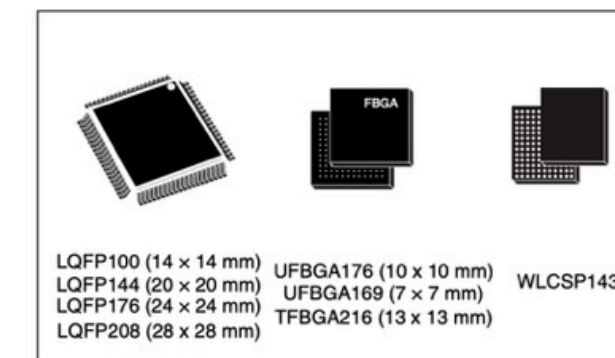


32b Arm® Cortex®-M4 MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 com. interfaces, camera & LCD-TFT

Datasheet - production data

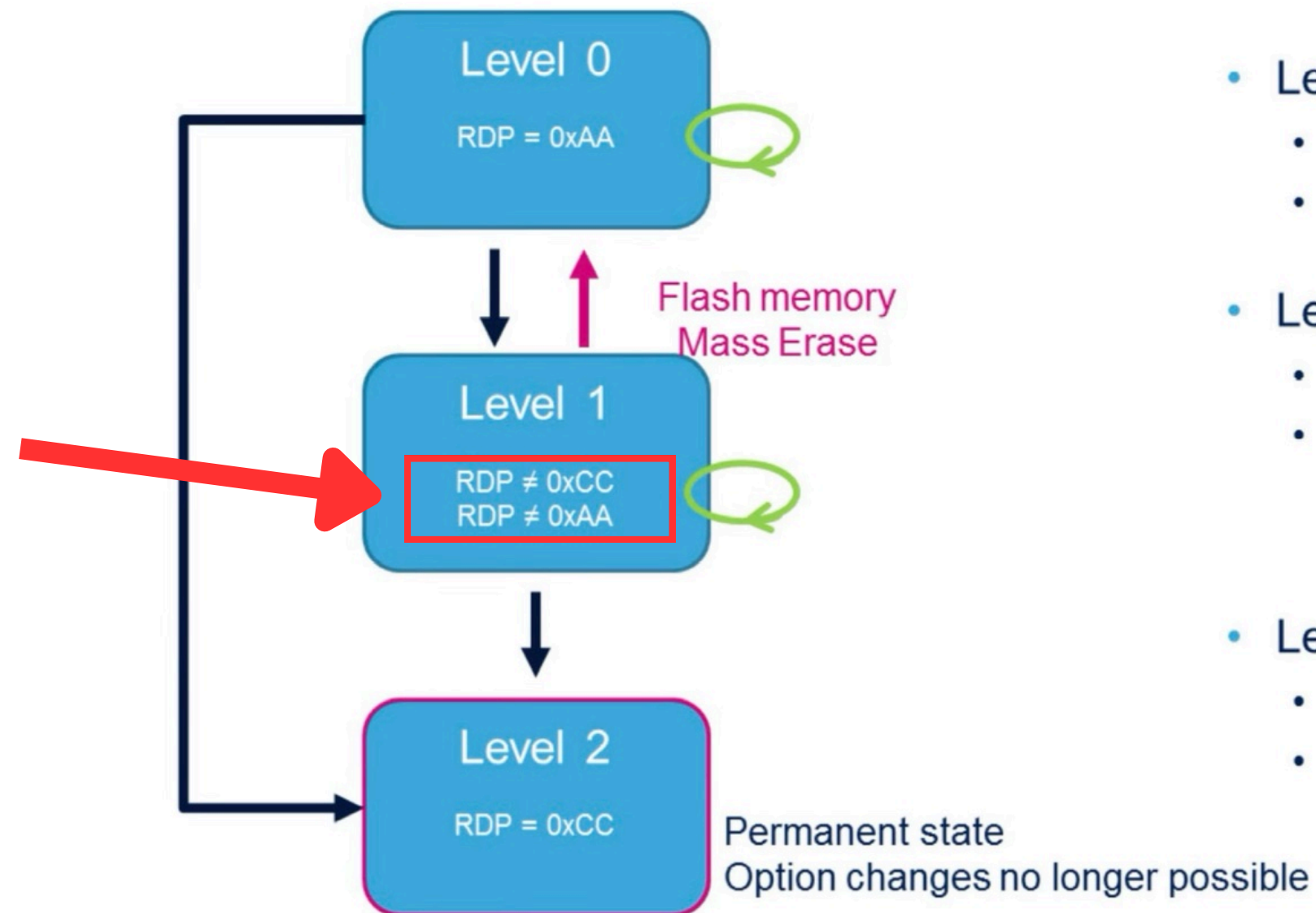
Features

- Core: Arm® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution from Flash memory, frequency up to 180 MHz, MPU, 225 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
 - Up to 2 MB of Flash memory organized into two banks allowing read-while-write
 - Up to 256+4 KB of SRAM including 64-KB of CCM (core coupled memory) data RAM
 - Flexible external memory controller with up to 32-bit data bus: SRAM, PSRAM, SDRAM/LPDDR SDRAM, Compact Flash/NOR/NAND memories
- LCD parallel interface, 8080/6800 modes
- LCD-TFT controller with fully programmable resolution (total width up to 4096 pixels, total height up to 2048 lines and pixel clock up to 83 MHz)
- Chrom-ART Accelerator™ for enhanced graphic content creation (DMA2D)
- Clock, reset and supply management
 - 1.7 V to 3.6 V application supply and I/Os
 - POR, PDR, PVD and BOR
 - 4-to-26 MHz crystal oscillator
 - Internal 16 MHz factory-trimmed RC (1% accuracy)
 - 32 kHz oscillator for RTC with calibration
 - Internal 32 kHz RC with calibration
- Low power
 - Sleep, Stop and Standby modes
 - V_{BAT} supply for RTC, 20×32 bit backup registers + optional 4 KB backup SRAM
- 3×12-bit, 2.4 MSPS ADC: up to 24 channels and 7.2 MSPS in triple interleaved mode
- 2×12-bit D/A converters
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support
- Up to 17 timers: up to twelve 16-bit and two 32-bit timers up to 180 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Debug mode
 - SWD & JTAG interfaces
 - Cortex-M4 Trace Macrocell™
- Up to 168 I/O ports with interrupt capability
 - Up to 164 fast I/Os up to 90 MHz
 - Up to 166 5 V-tolerant I/Os
- Up to 21 communication interfaces
 - Up to 3 × I²C interfaces (SMBus/PMBus)
 - Up to 4 USARTs/4 UARTs (11.25 Mbit/s, ISO7816 interface, LIN, IrDA, modem control)
 - Up to 6 SPIs (45 Mbits/s), 2 with muxed full-duplex I²S for audio class accuracy via internal audio PLL or external clock
 - 1 × SAI (serial audio interface)
 - 2 × CAN (2.0B Active) and SDIO interface
- Advanced connectivity
 - USB 2.0 full-speed device/host/OTG controller with on-chip PHY
 - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
 - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII
- 8- to 14-bit parallel camera interface up to 54 Mbytes/s
- True random number generator
- CRC calculation unit
- RTC: subsecond accuracy, hardware calendar
- 96-bit unique ID



RDP transition scheme

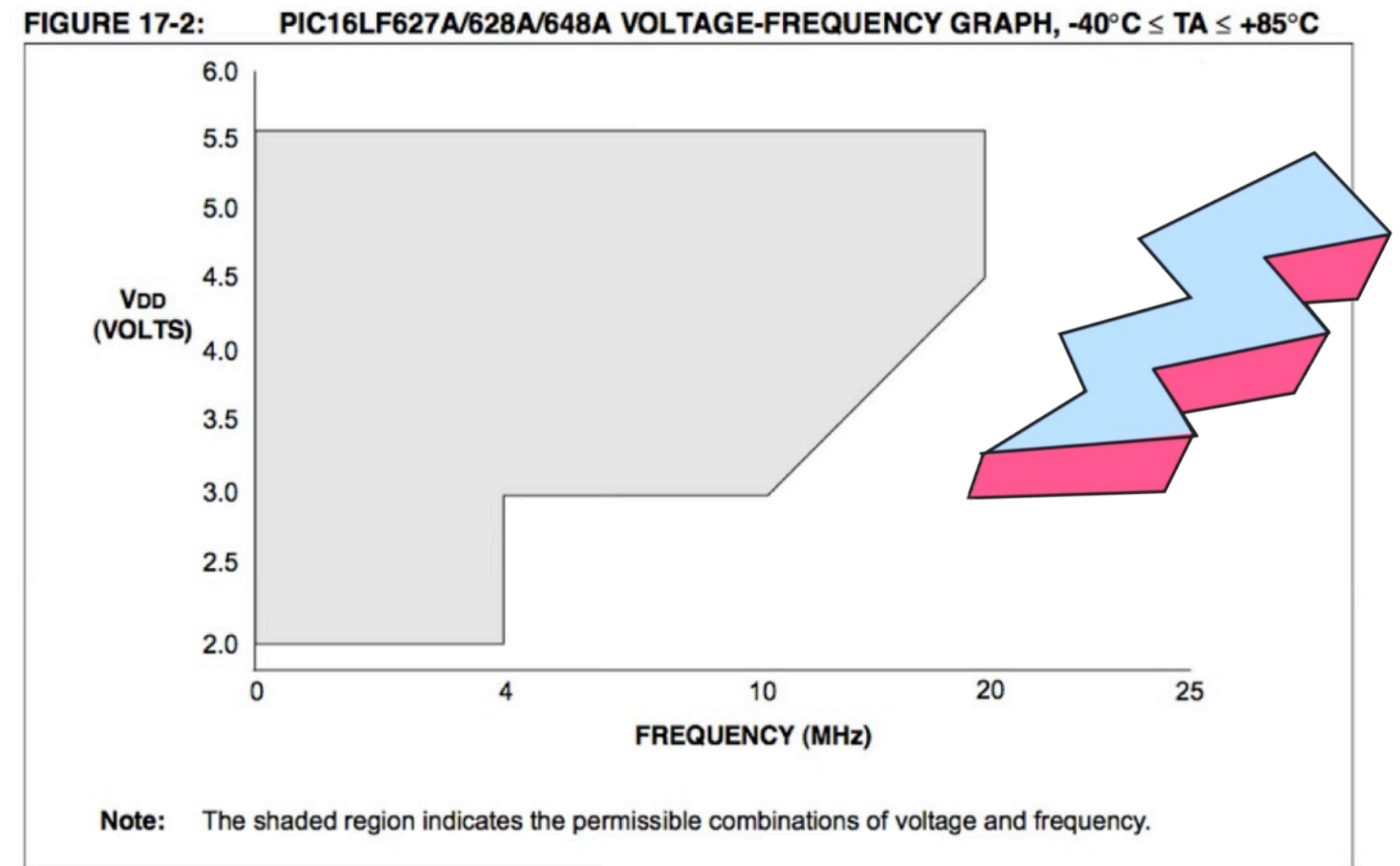
10



- Level 0
 - Option byte mods are allowed
 - Can transition to Level 1 or Level 2
- Level 1
 - Option byte mods are allowed.
 - Can transition to Level 0 or Level 2
 - Level 0 → Mass erase of user Flash memory, backup regs and SRAM2
- Level 2
 - Option bytes are frozen
 - No transition possible

FAULT INJECTION

- Devices have defined operating parameters
- What happens if you intentionally & momentarily operate outside the acceptable range?
- Could possibly cause a fault in the device
 - System reset/halt
 - Change in software decision
 - Skip an instruction
 - Affect branching
 - Computational fault
 - Instruction decoding errors
 - Malformed data read/write



STM32 BOOT MODES

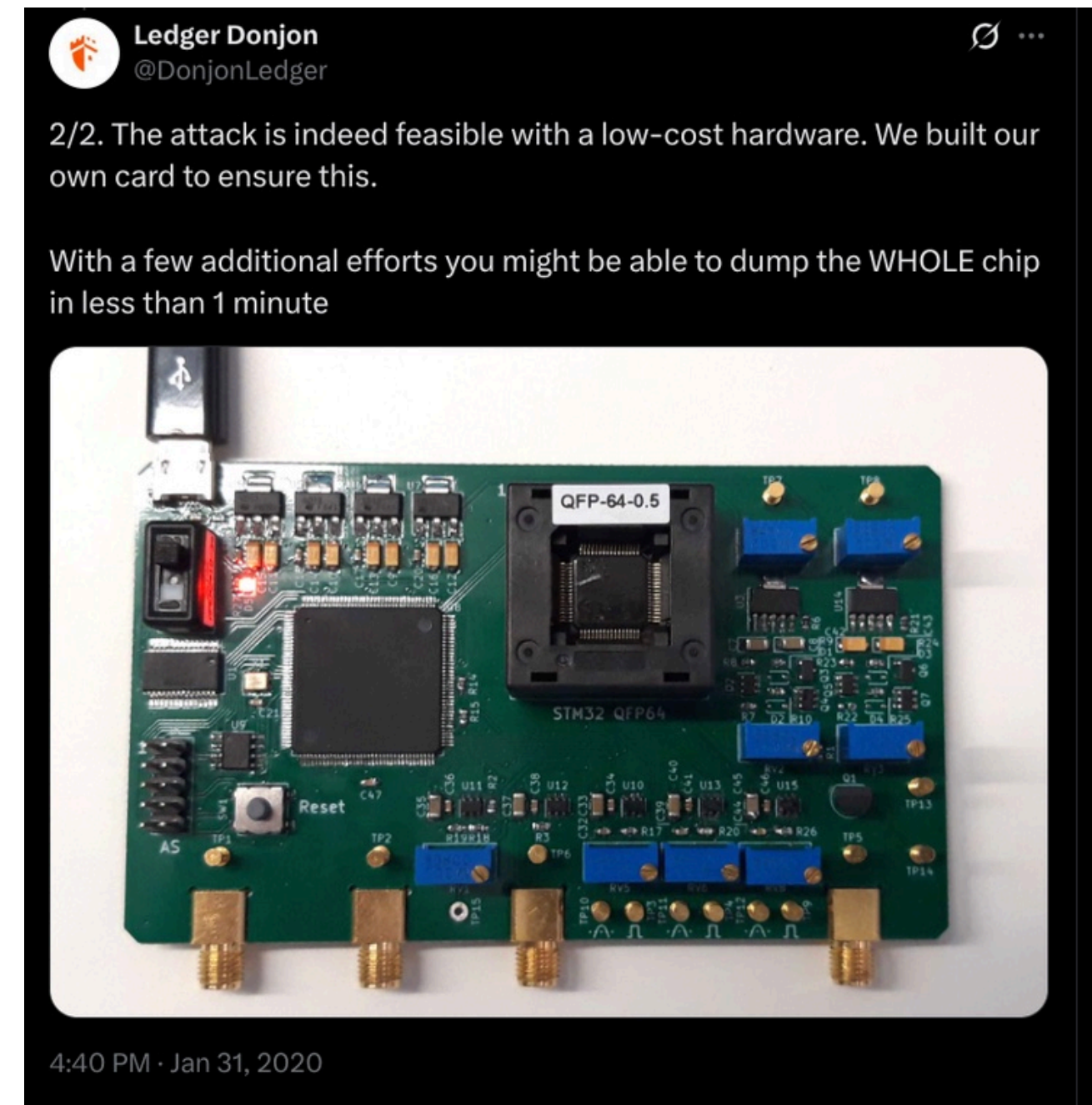
- [RM0033 STM32F205xx Reference Manual](#)
- The main flash memory boot mode is the only available option if the chip is in RDP2

Table 4. Boot modes

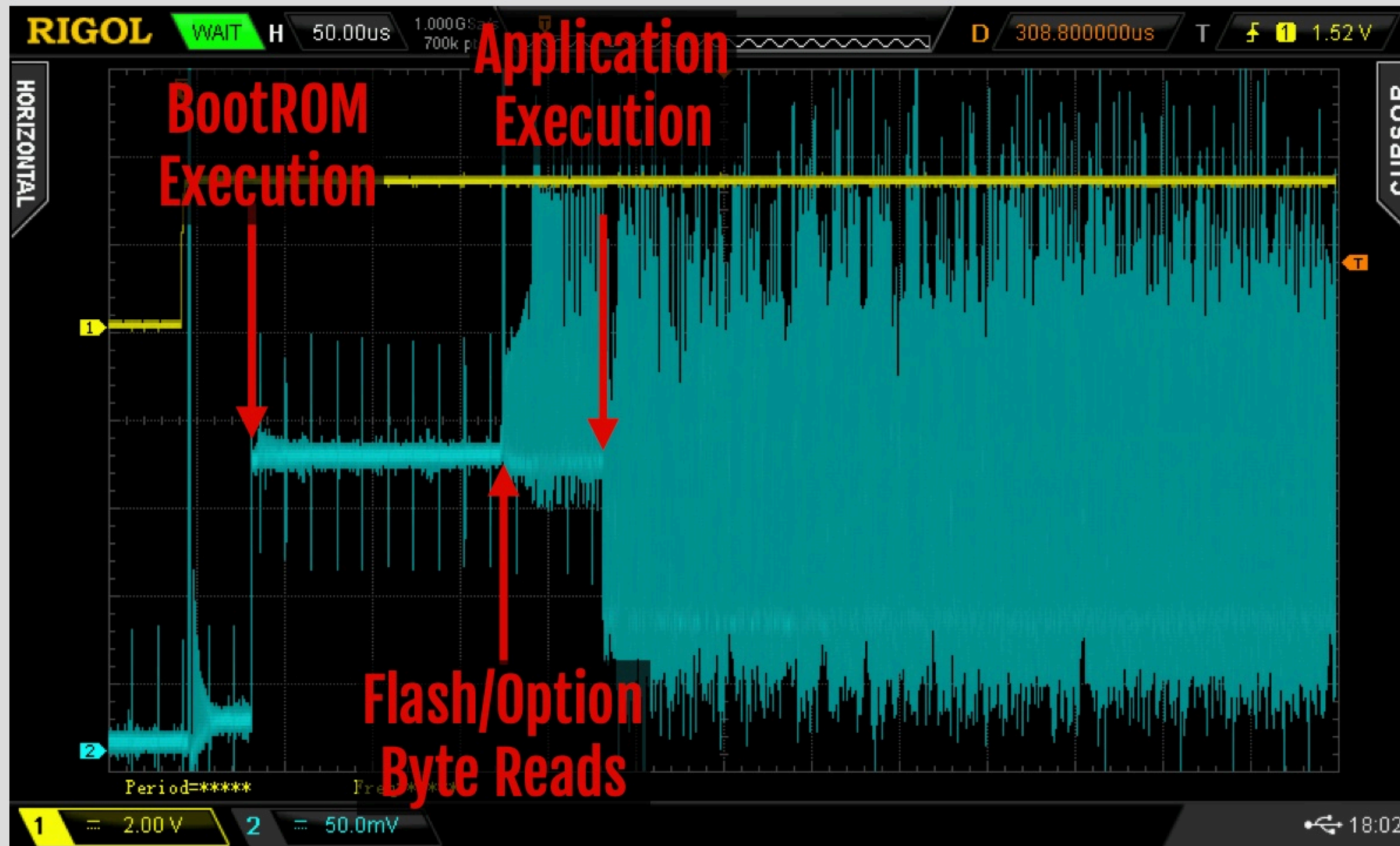
Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

EXISTING ATTACK TECHNIQUES

- RDP Downgrade
 - [wallet.fail](#) / [chip.fail](#) (2018/2019)
- Bootloader Read Memory Glitch (STM32F1)
 - [Shaping the Glitch](#), Bozzato, et al. (2019)
 - [Prog.World](#) (2020)
- Combined
 - [Kraken Security Labs](#) (2019)
 - [Wrongbaud](#) (Downgrade + Bootloader Go) (2022)
- Other
 - [ROM bootloader USB DFU bug](#), Goodspeed (2012)
 - [Ledger unknown attack technique](#) (2020)
 - [FPB \(Flash Patch and Breakpoint\)](#), Obermaier, et al. (2020)
 - [Flash erase suppression](#), Schink, et al. (2024)



Power consumption after reset (200μs)



PROBLEMS W/ EXISTING TECHNIQUES

- Can lead to device corruption or permanent loss of data
- We weren't successful in reproducing some attacks
- Some attacks did not have any public details
- Previous research does not cover a large enough sample size to account for device variability when performing fault injection on multiple chips

PROBLEM STATEMENT

We're dealing with real-world devices potentially containing millions of dollars of cryptocurrency. Therefore, we require a repeatable technique with a next to zero failure rate that works reliably across STM32 microcontrollers (protected by RDP level 2) extracted from different devices.

WISH LIST

- Complete Flash extraction from an RDP2-secured MCU
- Near-zero risk of unintentional corruption or data loss
- Reliable and repeatable
- Limit the number and type of faults
- Use accessible/available hardware
- No semi-invasive or invasive attacks

EMFI

CNC 3018 Pro Max

ChipSHOUTER

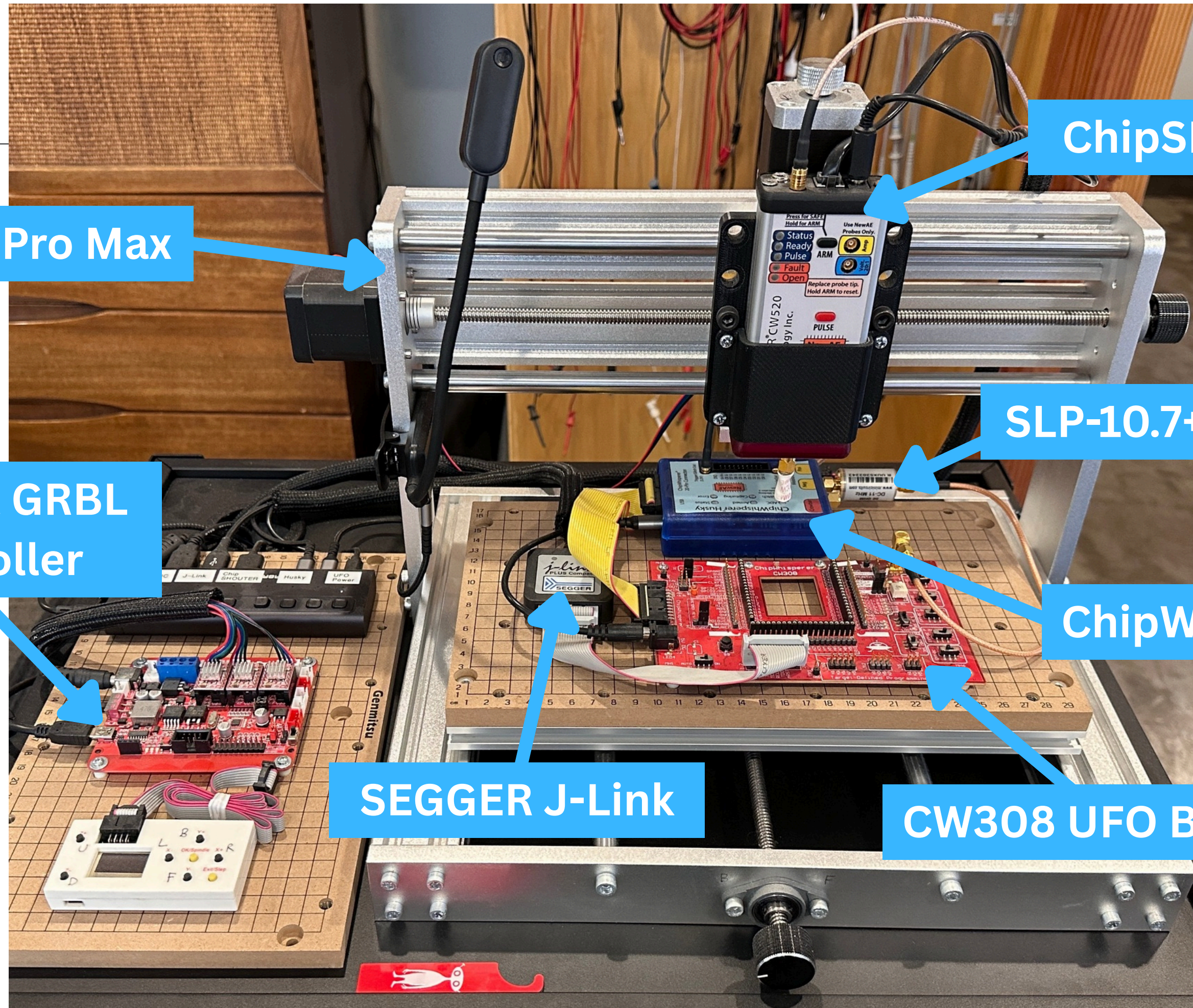
Doesbot GRBL
Controller

SLP-10.7+ Low Pass Filter

ChipWhisperer Husky

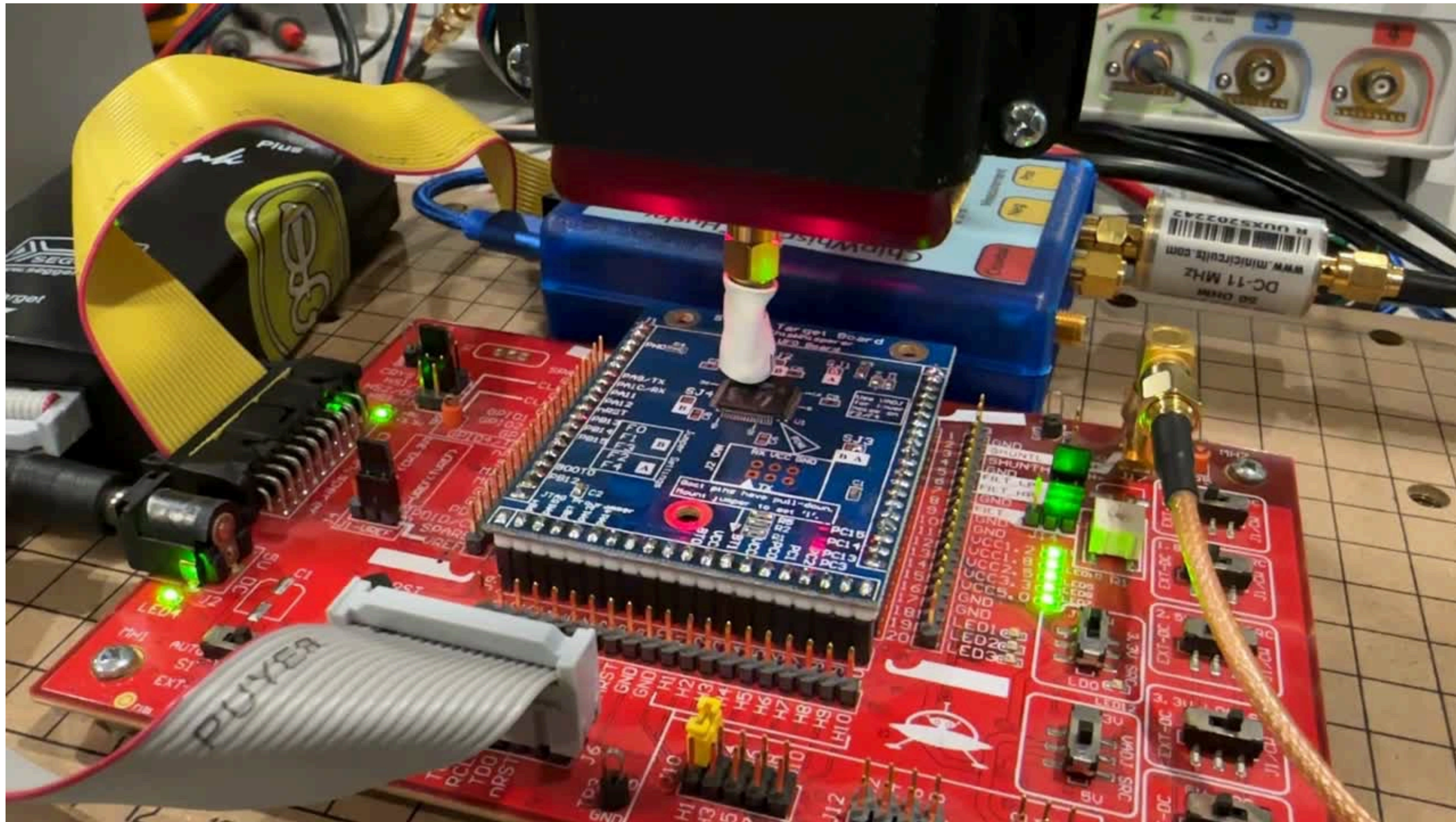
SEGGER J-Link

CW308 UFO Board



AUTOMATED SCAN

- Identify vulnerable areas of the chip



DEVICE CHARACTERIZATION

- Goal is to determine parameters across target device
 - Probe type, probe position, glitch settings (voltage, offset, trigger)
- Different parameters can cause different results
 - No change, reset, option byte corruption, permanent downgrade, Flash wipe/erase
- Use test code first to ensure device is vulnerable, then increase to specific attacks
- Can be manual or automated positioning
- Visualize results w/ plots/graphs

CHARACTERIZATION TEST CODE

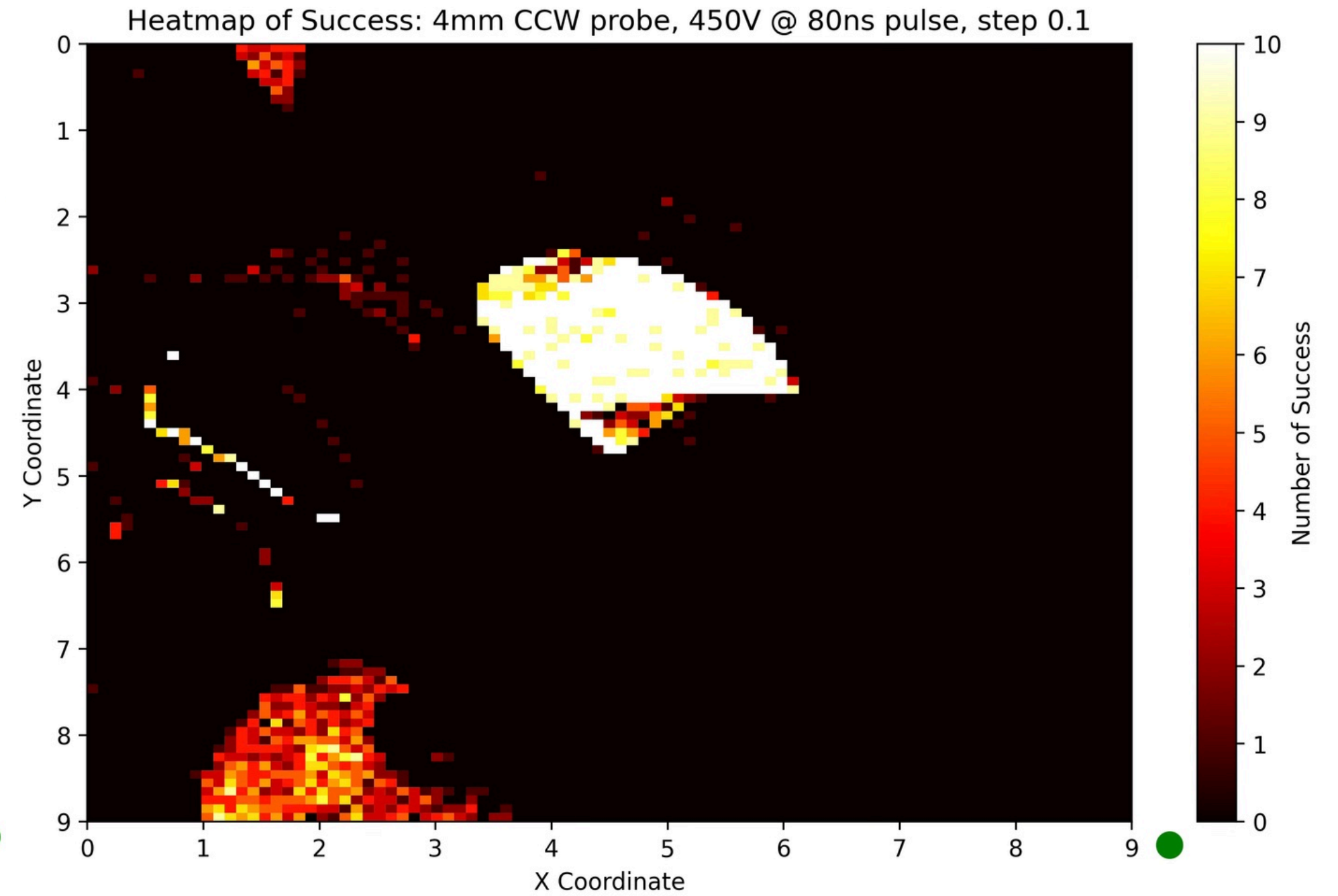
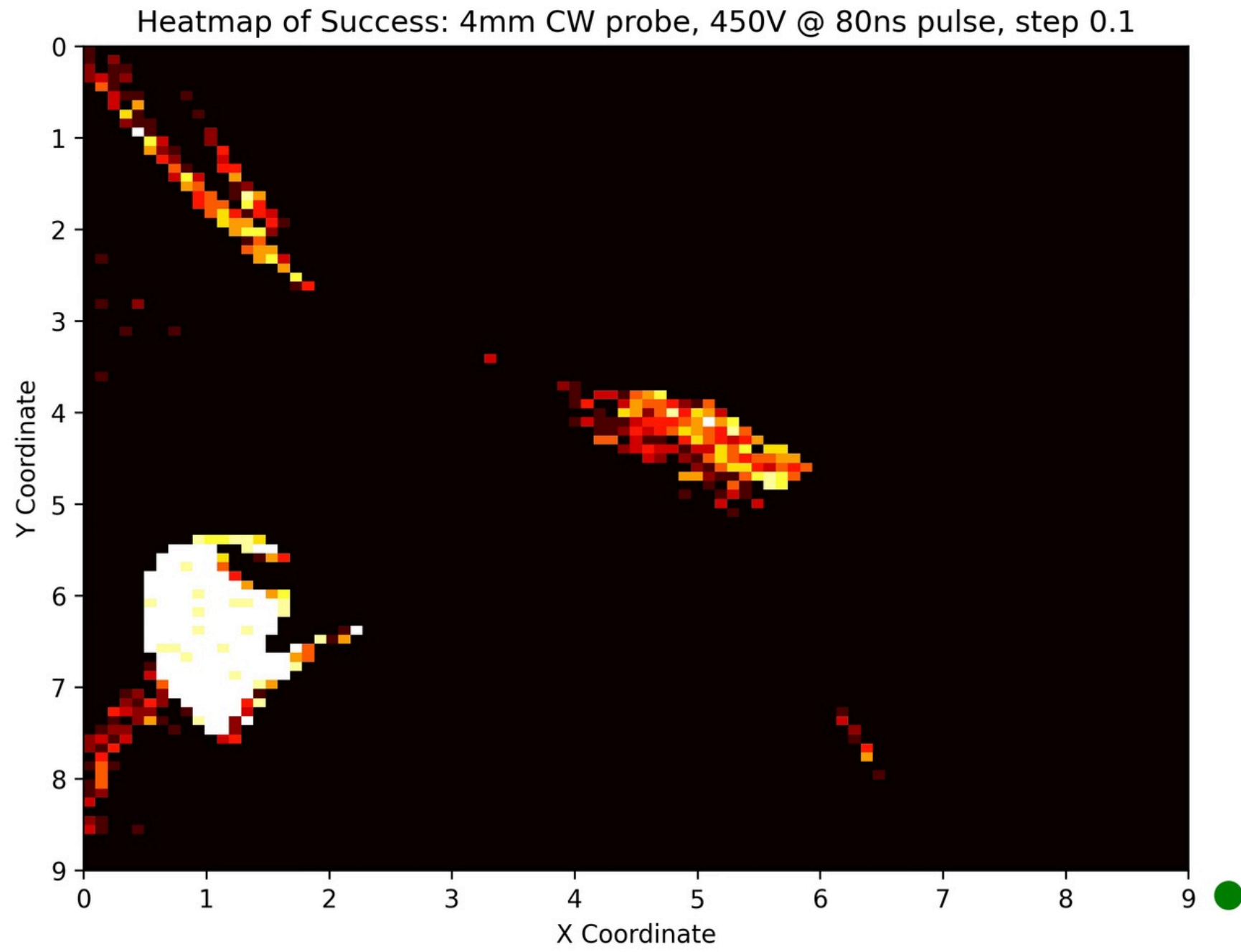
- Start-up string
- Nested for() loop
- Read arbitrary data (register, memory, etc.)
 - Many things can go wrong during glitching, being able to read memory helps to debug

```
void glitch(void)
{
    unsigned int k = 0;
    //Declared volatile to avoid optimizing away loop.
    //This also adds lots of SRAM access
    volatile uint16_t i, j;
    volatile unsigned int cnt;
    cnt = 0;

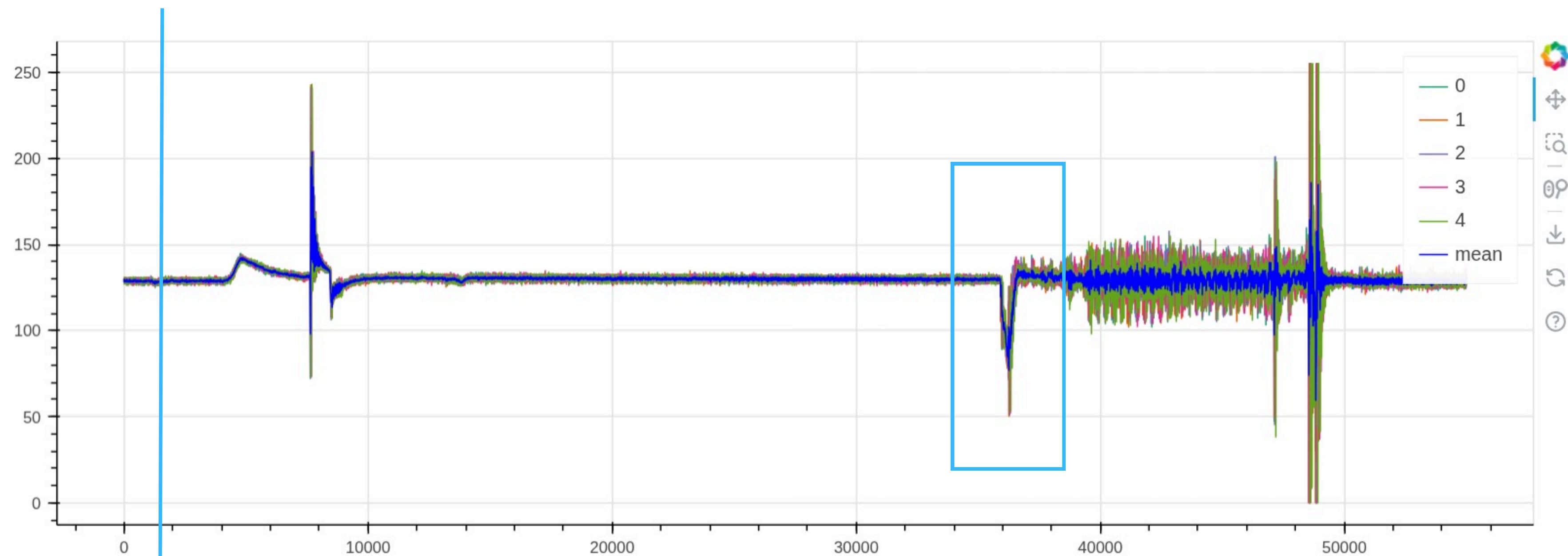
    HAL_GPIO_WritePin(triggerpin_GPIO_Port, triggerpin_Pin, GPIO_PIN_SET);
    for(i=0; i<50; i++){
        for(j=0; j<50; j++){
            cnt++;
        }
    }
    HAL_GPIO_WritePin(triggerpin_GPIO_Port, triggerpin_Pin, GPIO_PIN_RESET);

    printf("%u %u %u %u\n", cnt, i, j, k++);
}
```


F2 CHARACTERIZATION: TEST CODE

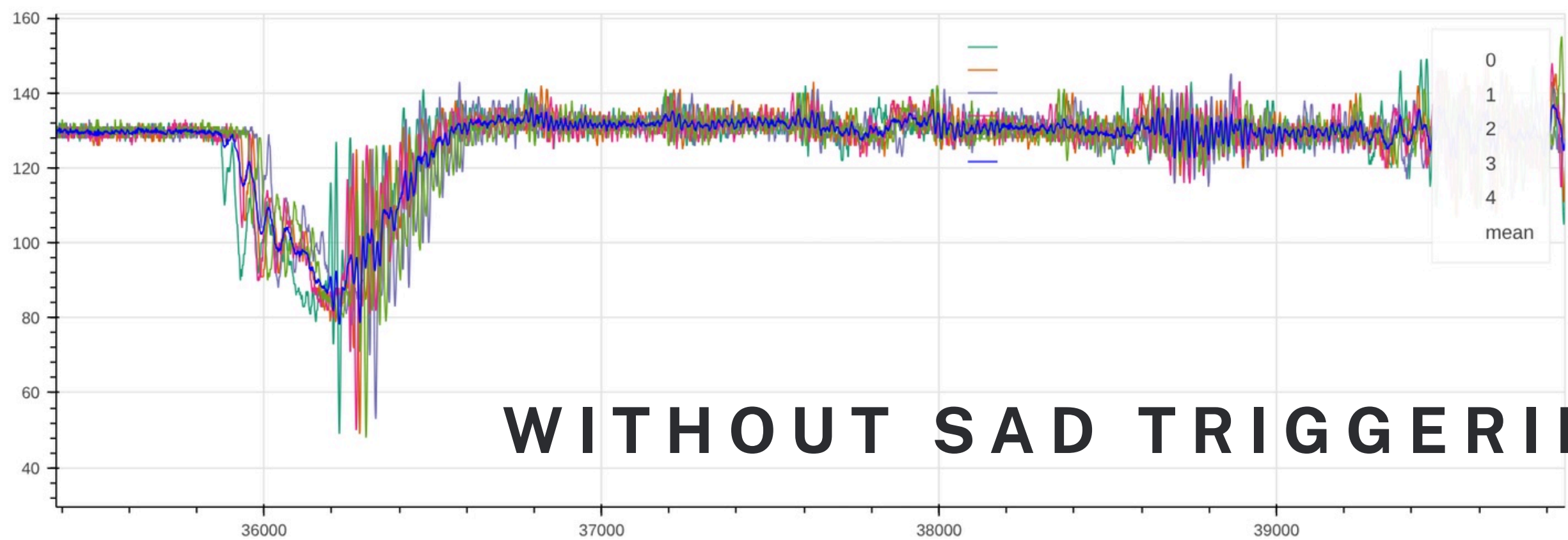


F2 RDP DOWNGRADE: POWER ANALYSIS

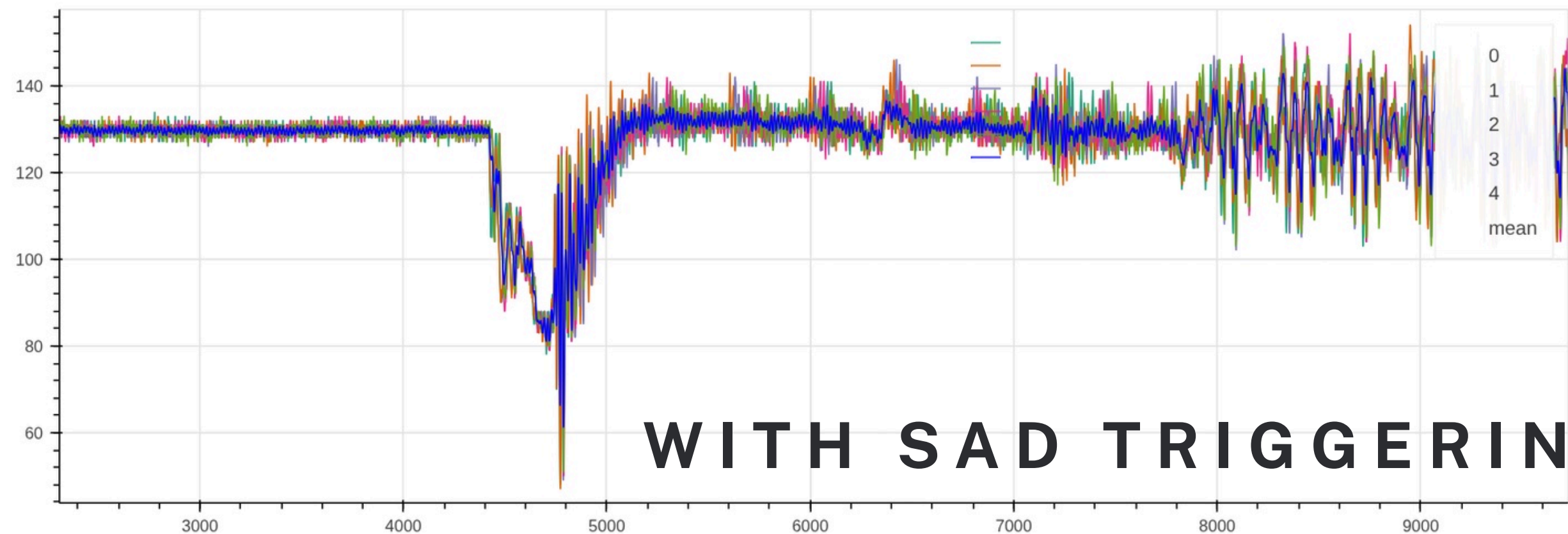


NRST TRIGGER

F2 RDP DOWNGRADE: POWER ANALYSIS



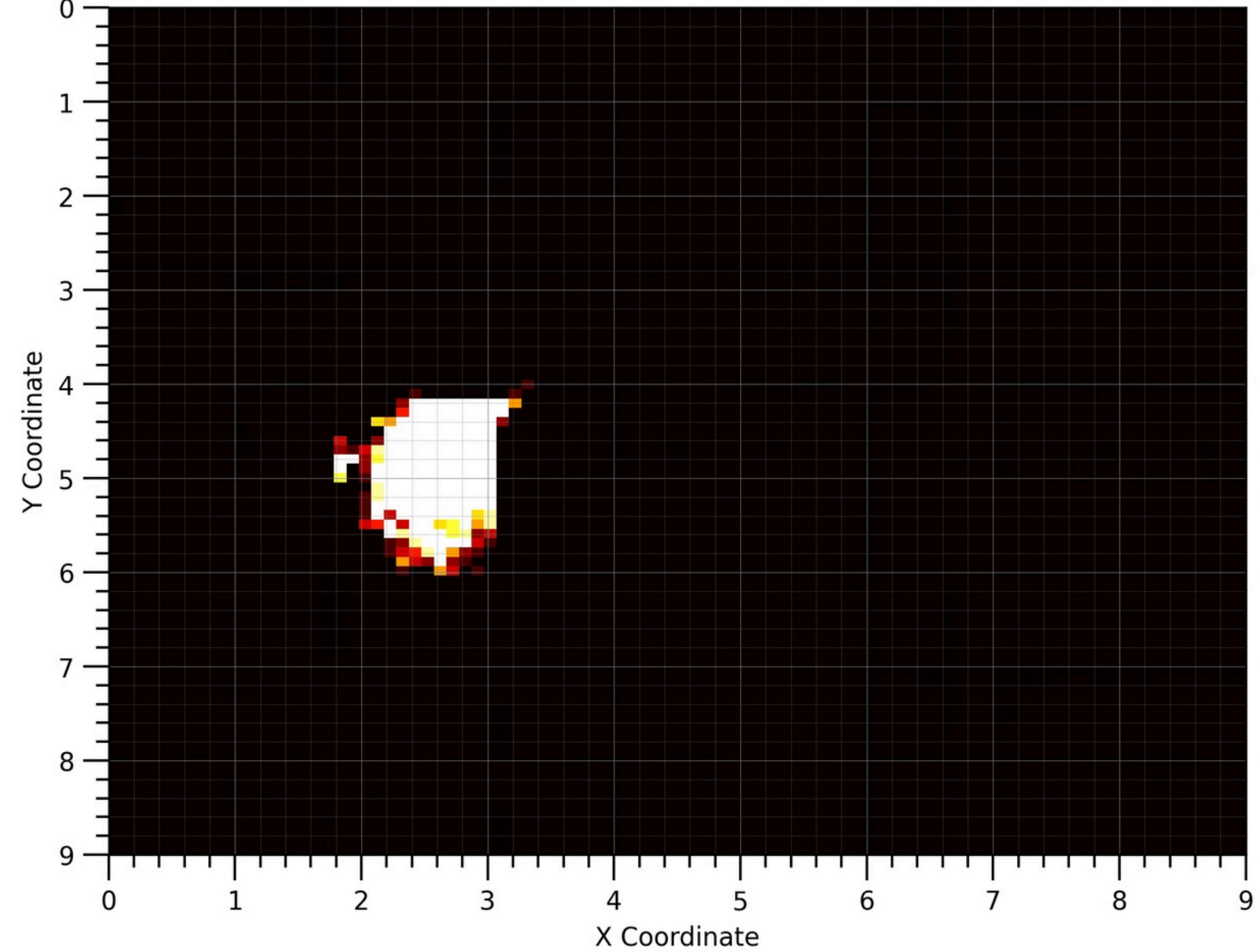
WITHOUT SAD TRIGGERING



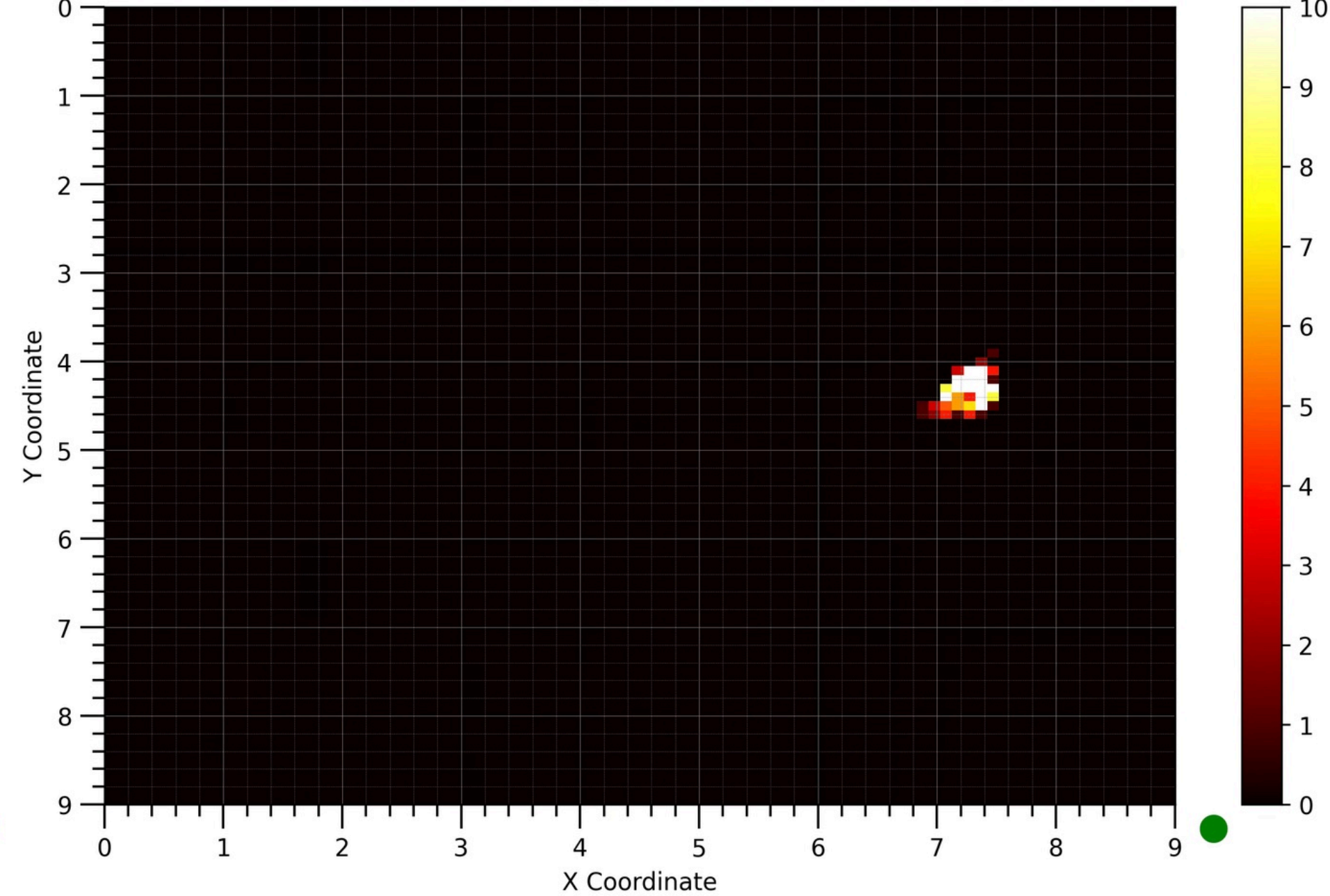
WITH SAD TRIGGERING

F2 RDP DOWNGRADE: POSITION (CW)

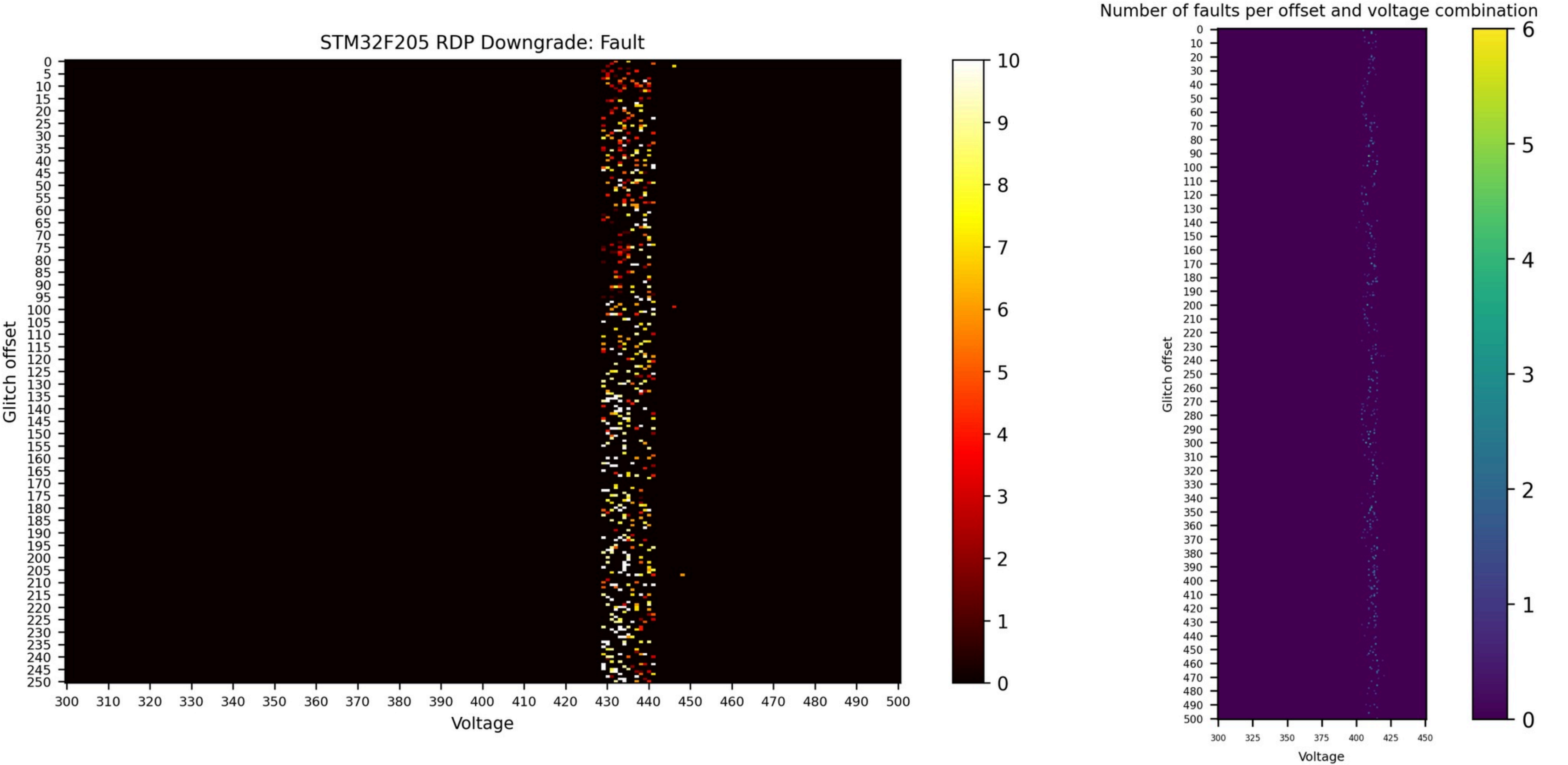
STM32F205 RDP BL Downgrade: Fault, 4mm CW, step 0.1



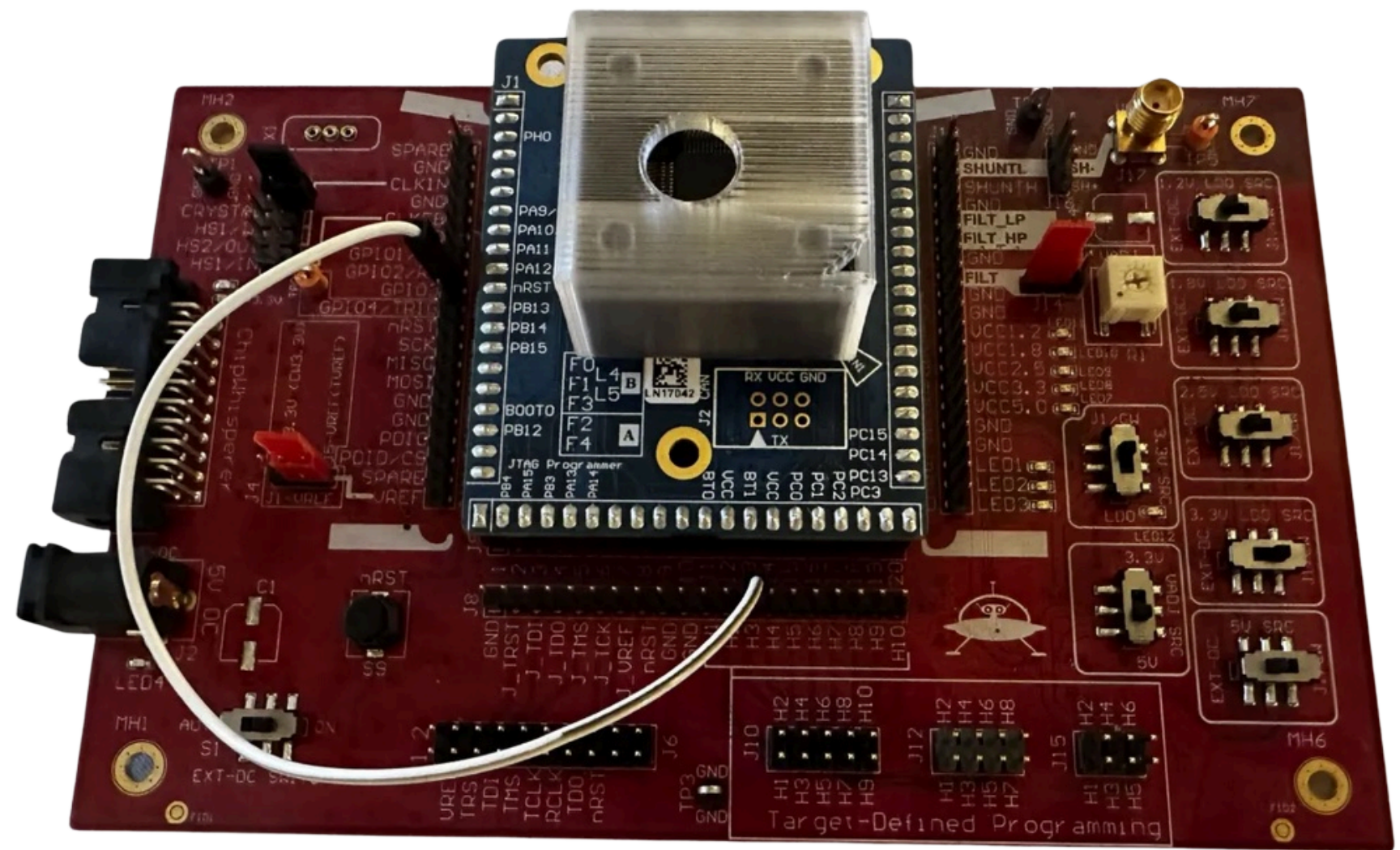
STM32F205 RDP BL Downgrade: Fault, 4mm CCW, Step 0.1



F2 RDP DOWNGRADE: OFFSET INDEPENDENT



EMFI ALIGNMENT HATS



Worked for ~5 samples...
...then things got weird!

F2 RDP DOWNGRADE: PROBE POSITIONING

- Different devices resulted in different probe location for RDP downgrade
 - Based on fab location, packaging location, other manufacturing parameters?

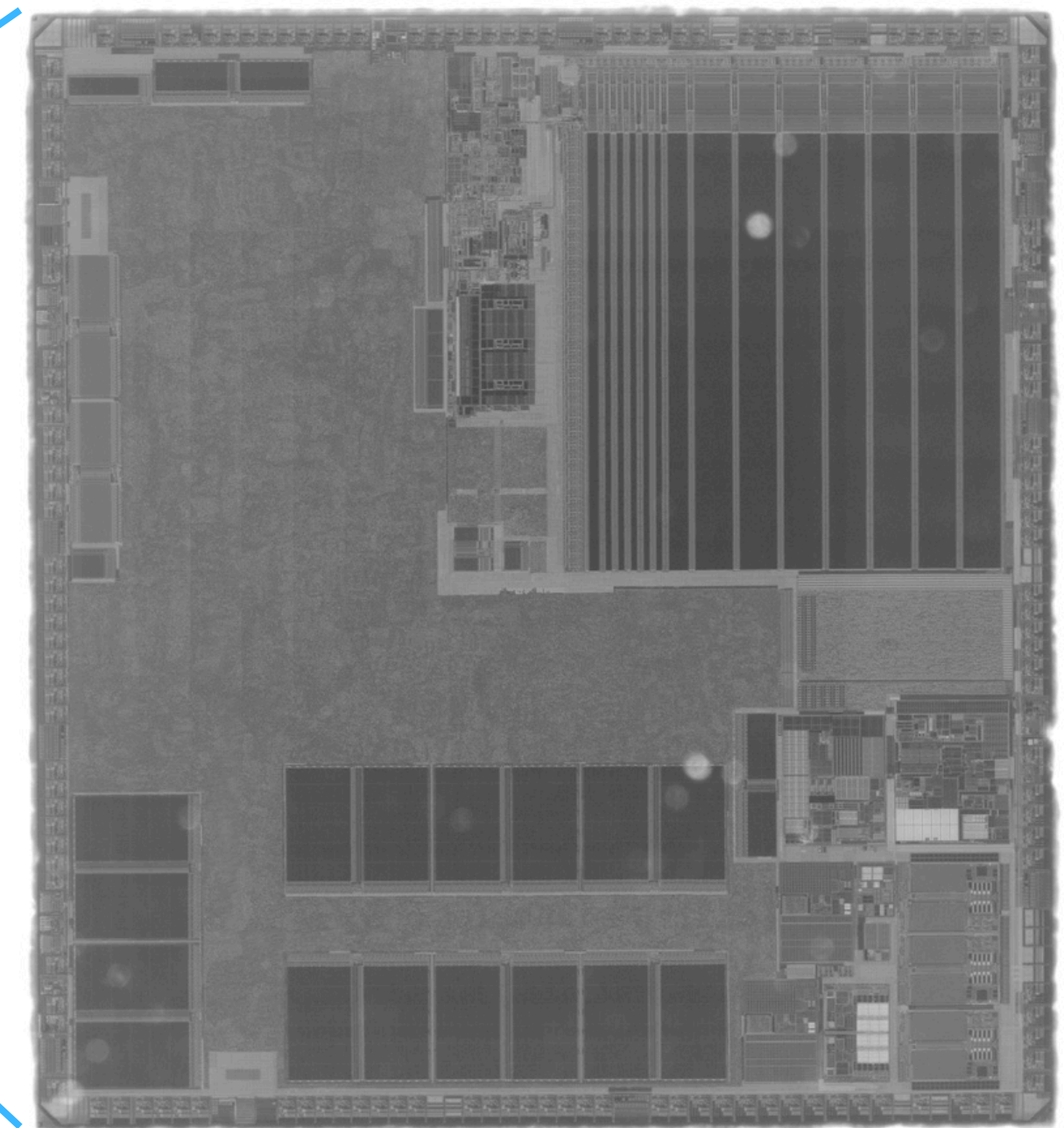
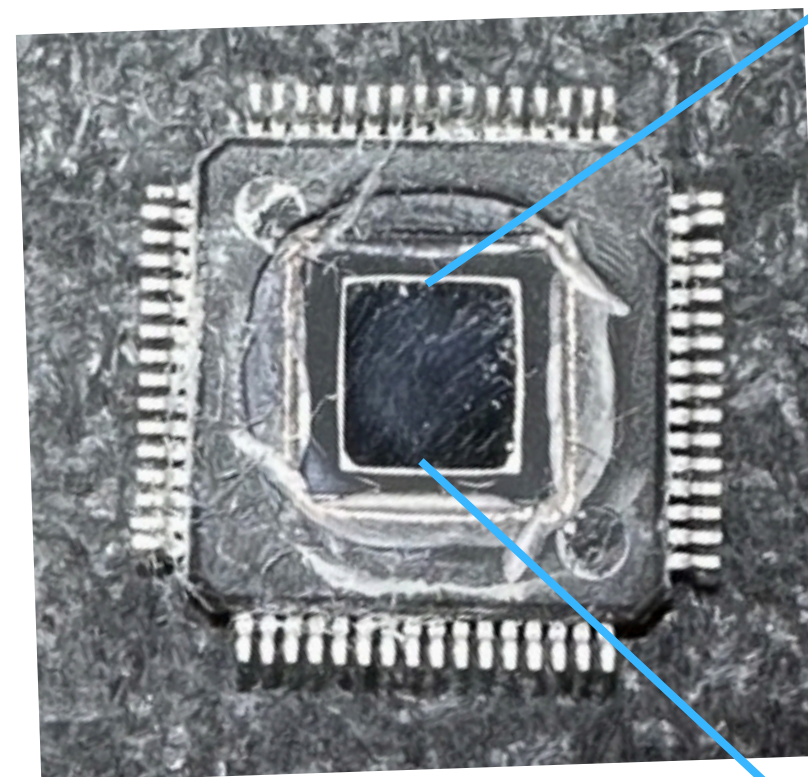
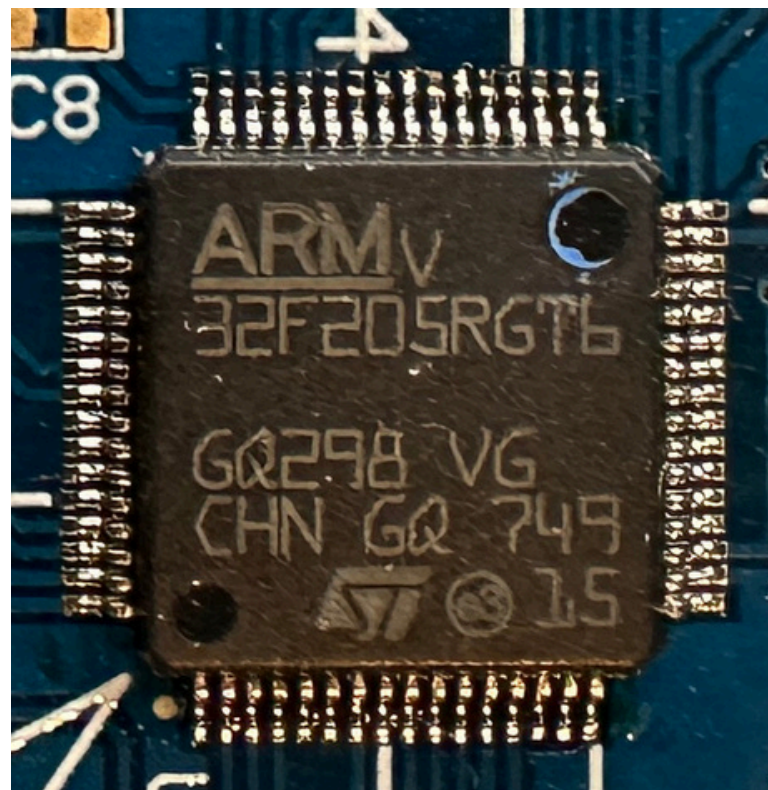
125%

View

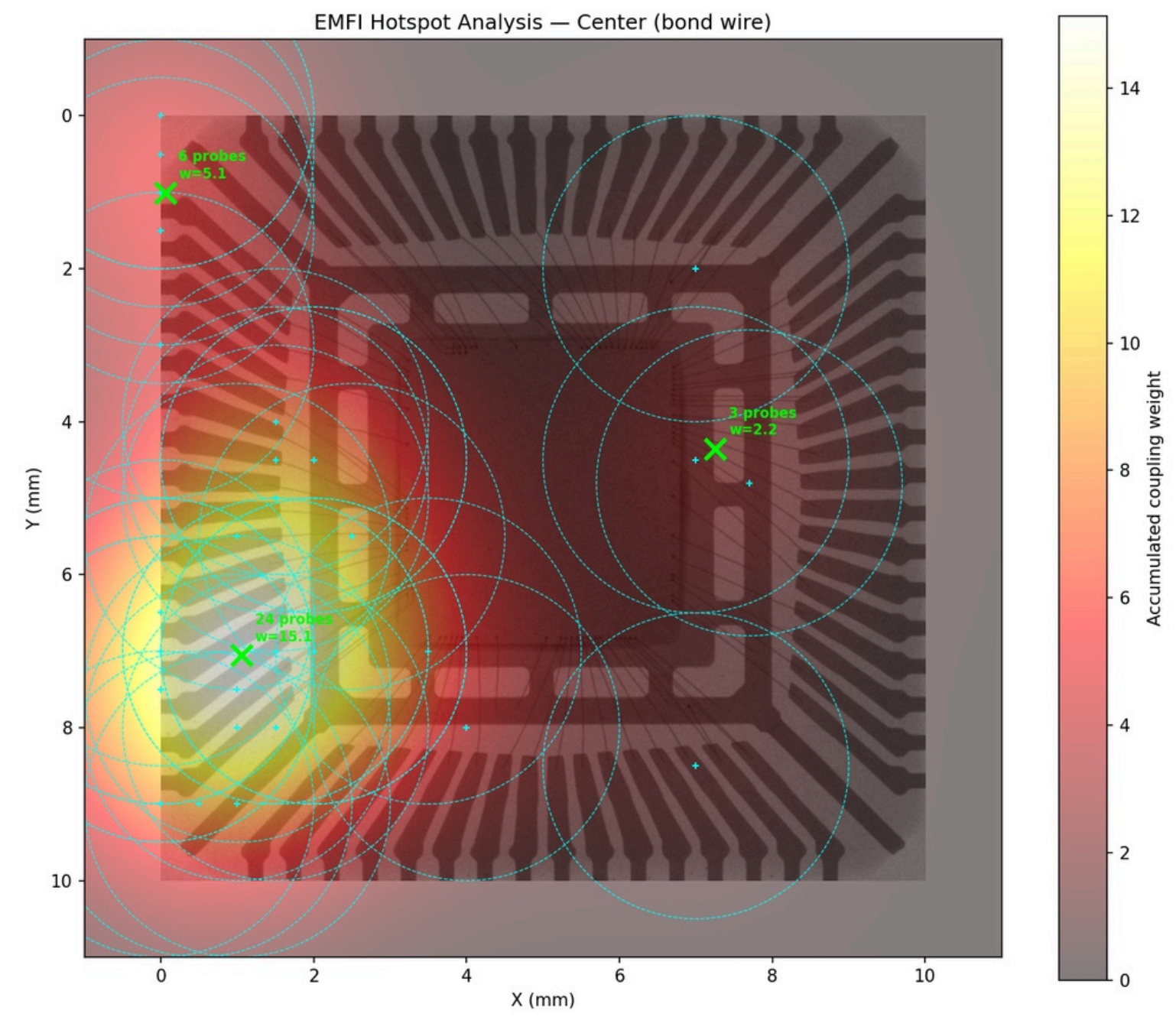
Zoom

CHIP PACKAGING

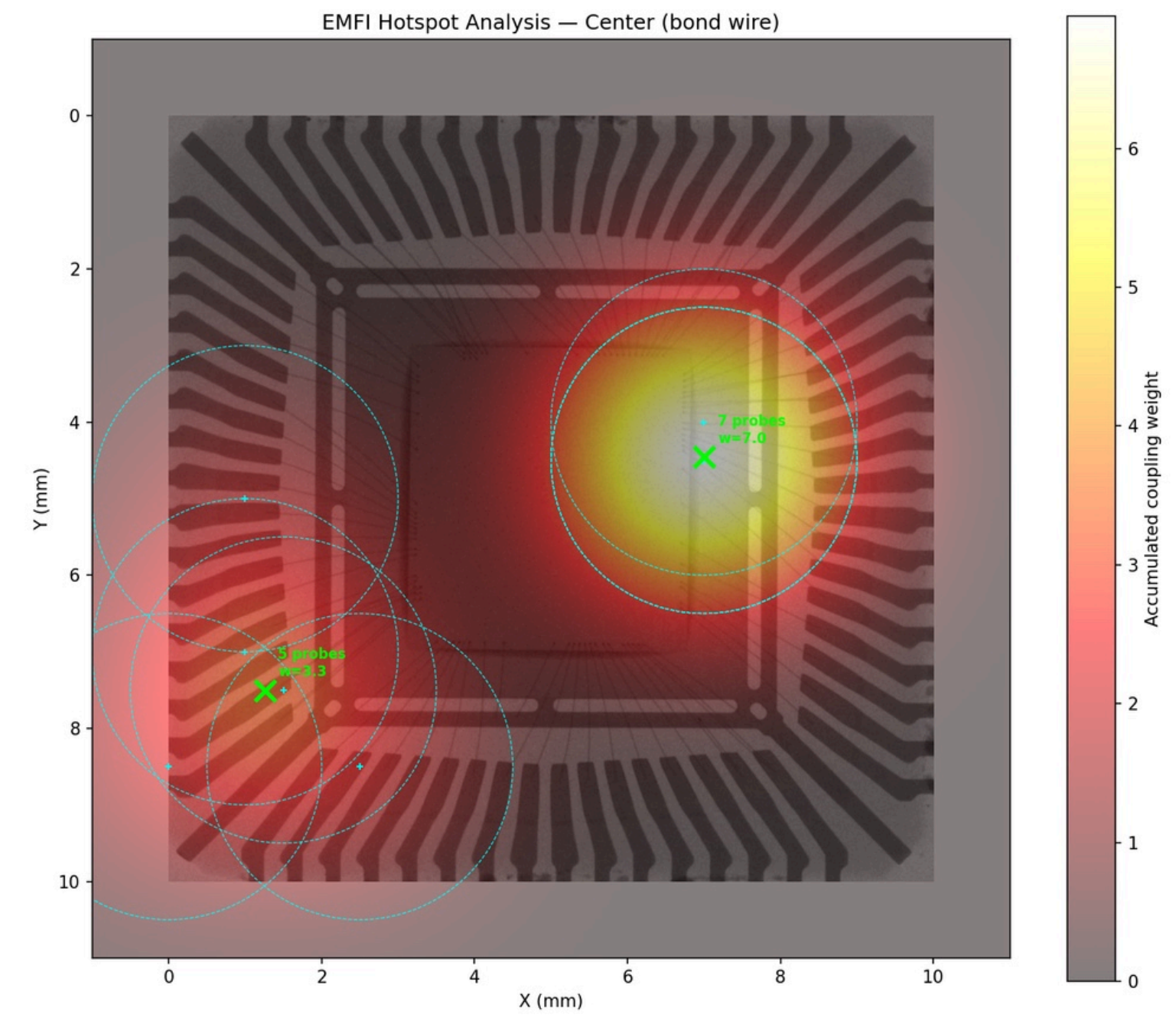
- EMFI can be more localized than VFI, but often couples to bond wires
- For many package types, the actual chip/die is much smaller than the package



X-RAY + EMFI LOCATION

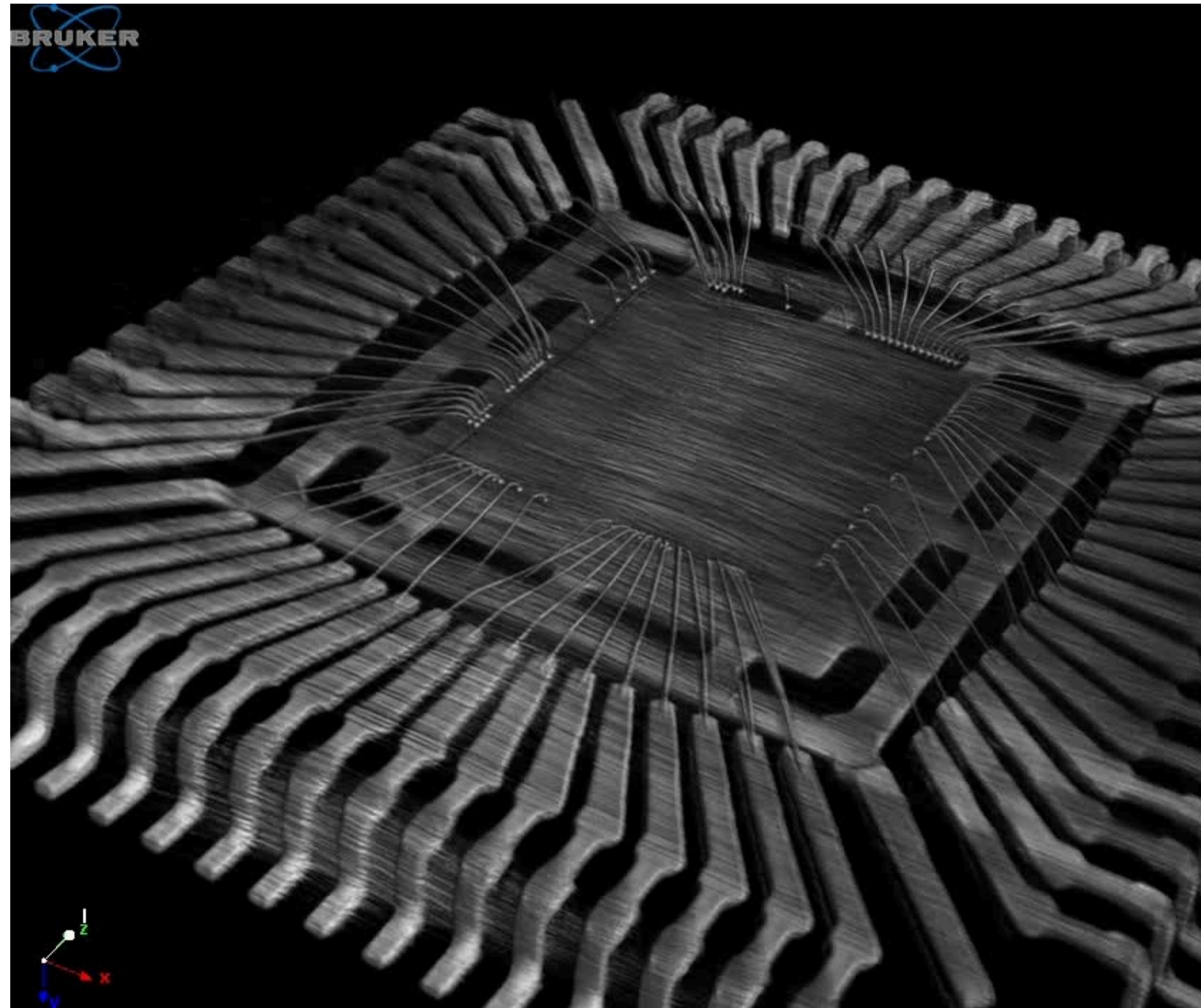


PHL (Phillipines)

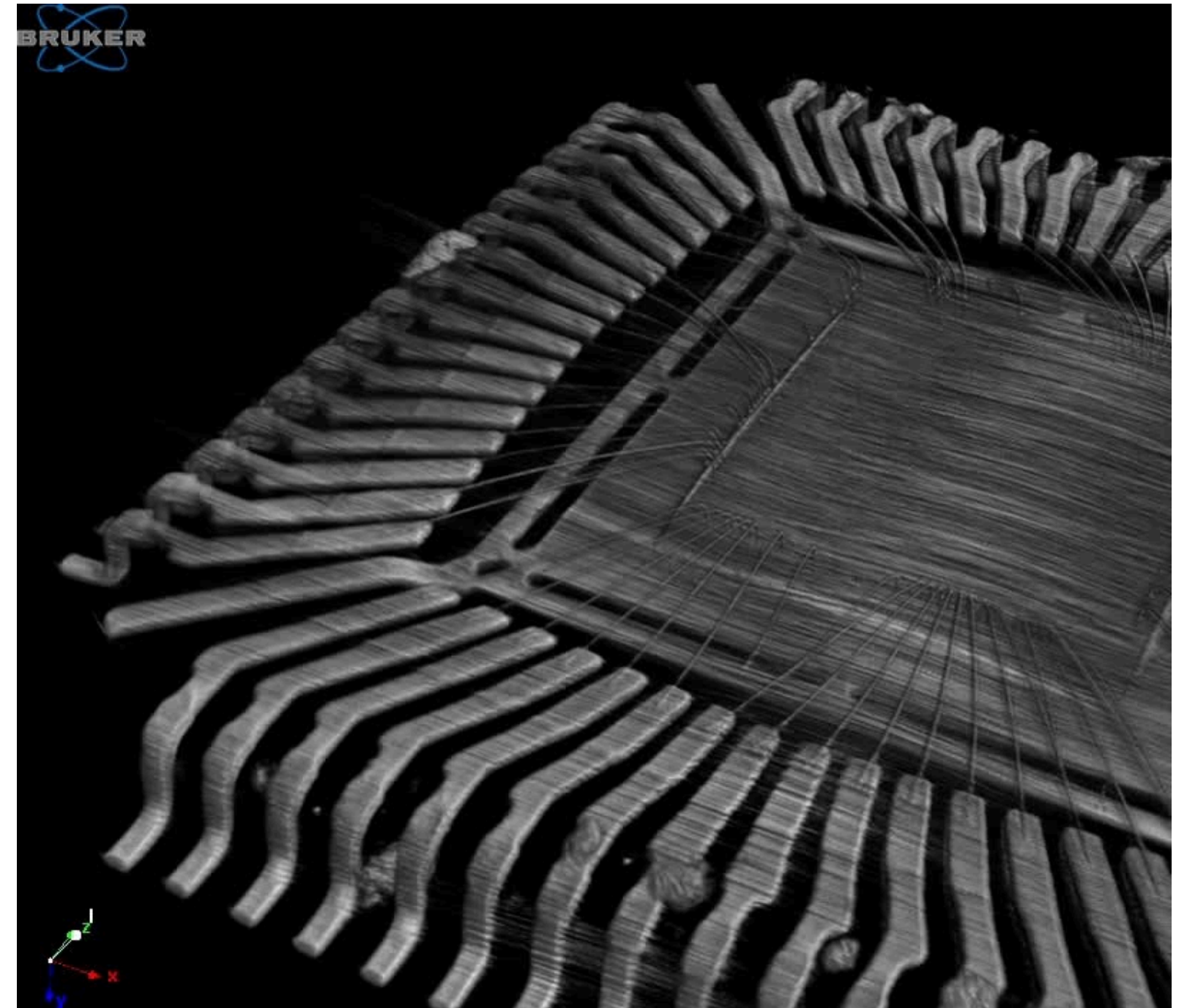


CHN (China)

WIREBOND ANALYSIS (CT)

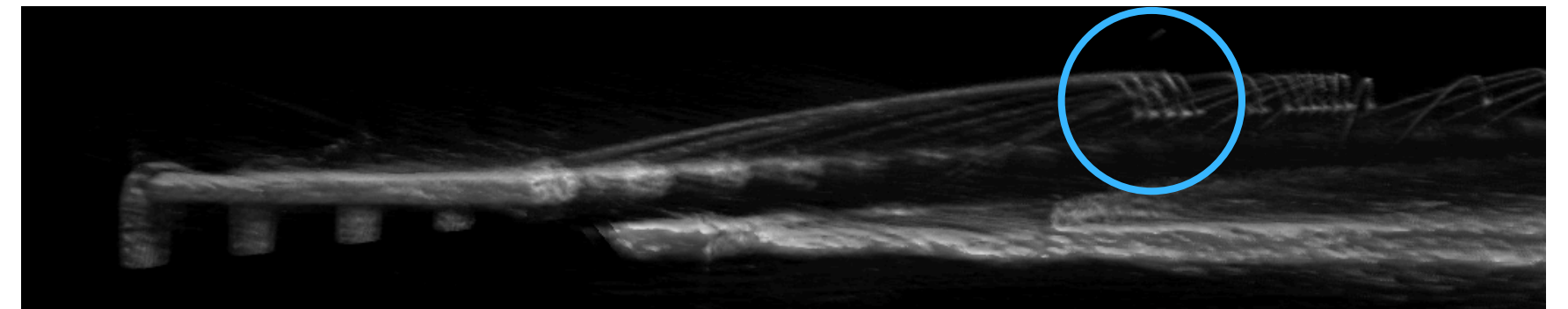
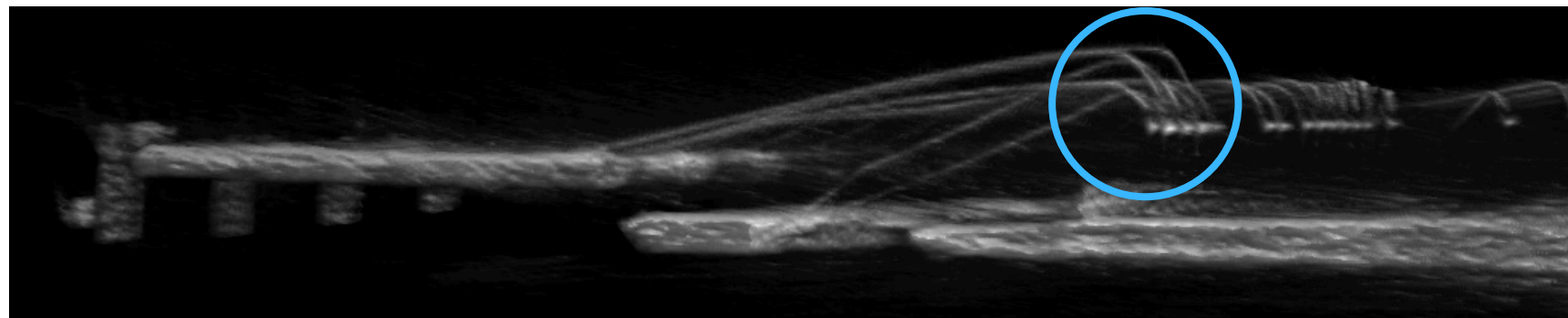
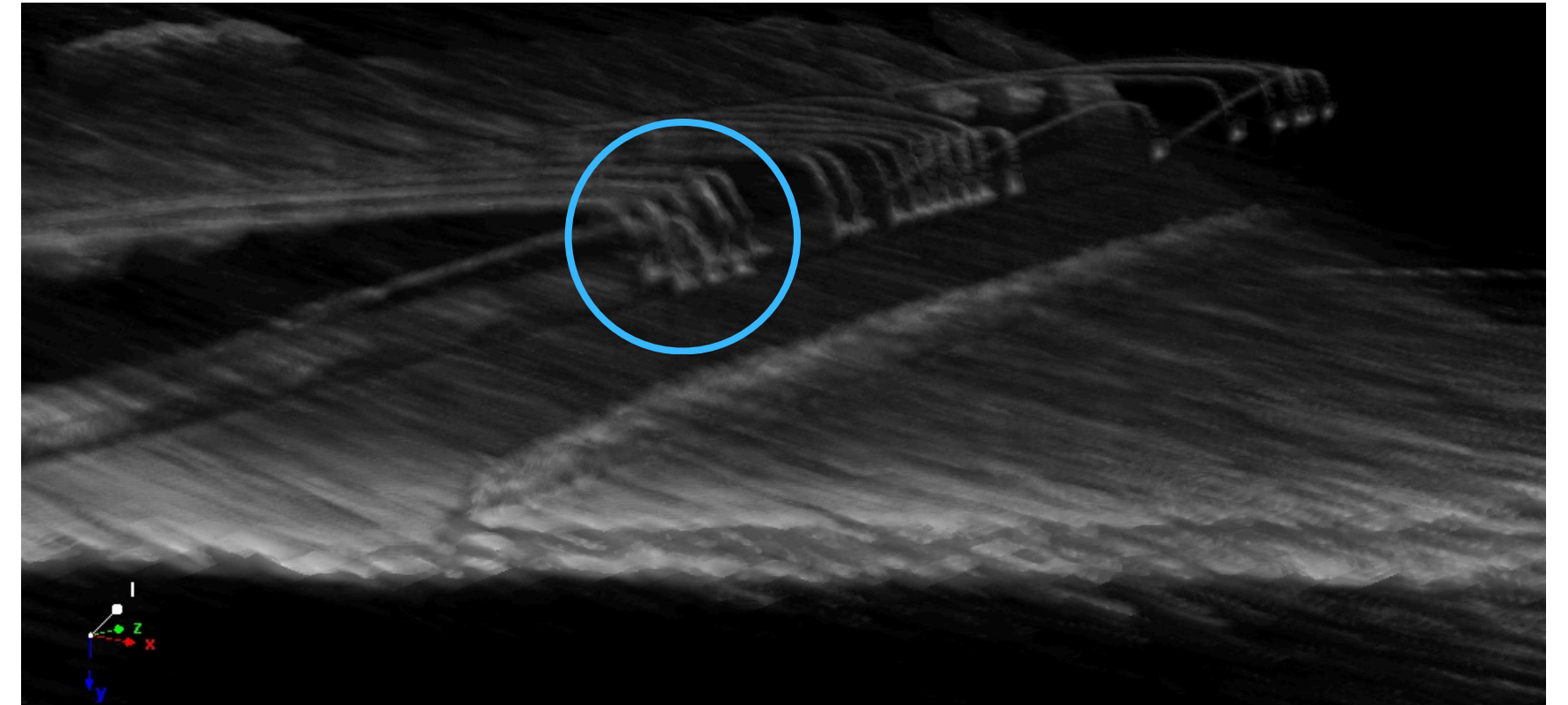
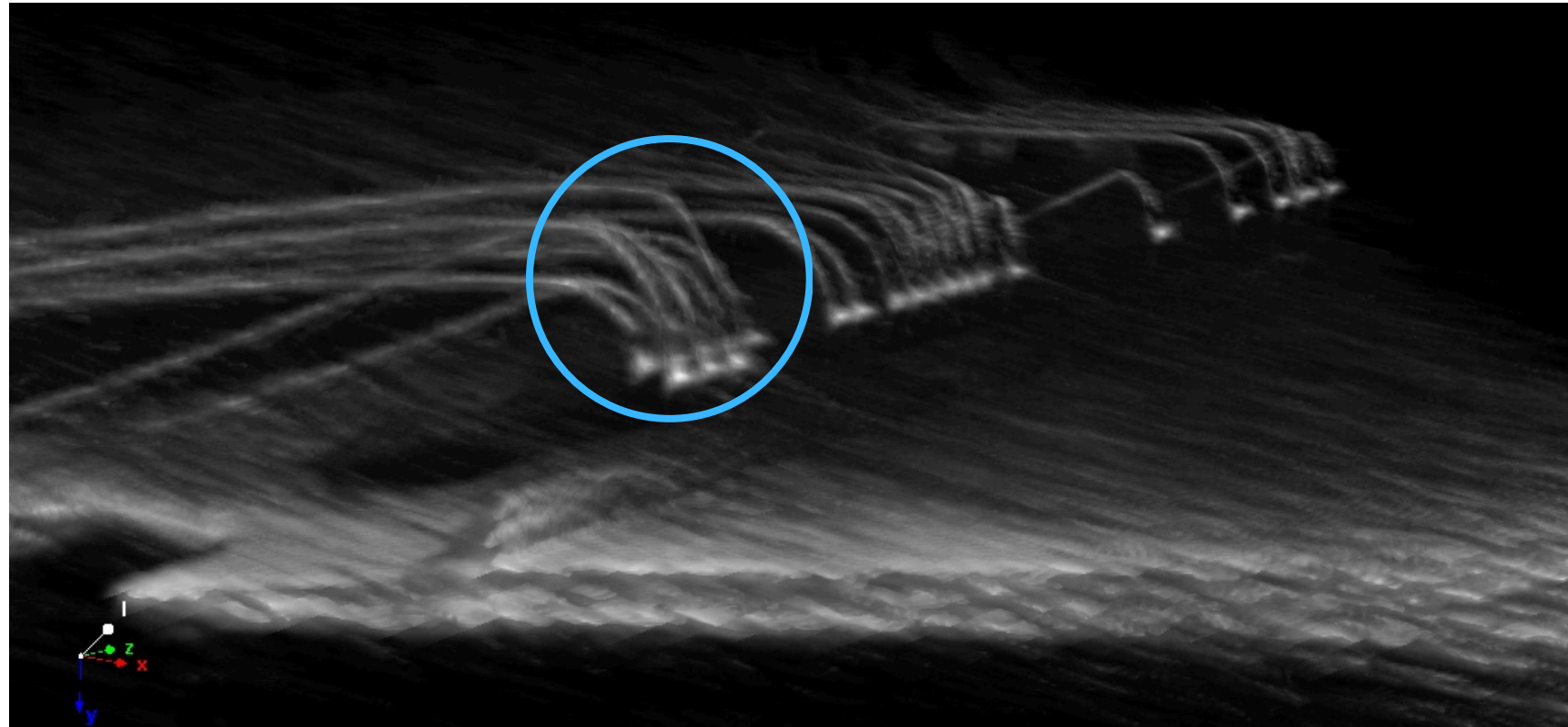


PHL (Phillipines)



CHN (China)

WIREBOND ANALYSIS (CT)

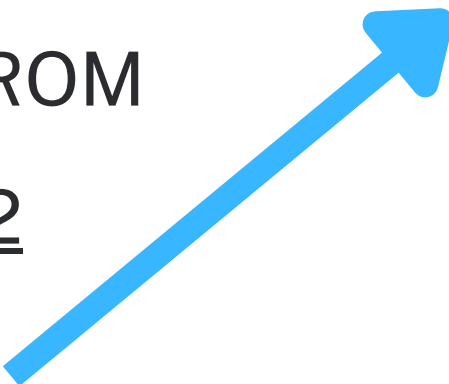


PHL (Phillipines)

CHN (China)

STM32 BOOTLOADER

- After downgrading to RDP1 we can access the bootloader serial interface
- Provides a low-level, command-based interface to perform programming-related operations on the STM32
- Supports multiple peripherals depending on version
 - UART, CAN, USB, I2C, SPI
- Stored in the internal (unchangeable) Boot ROM
- AN3155: USART protocol used in the STM32 bootloader

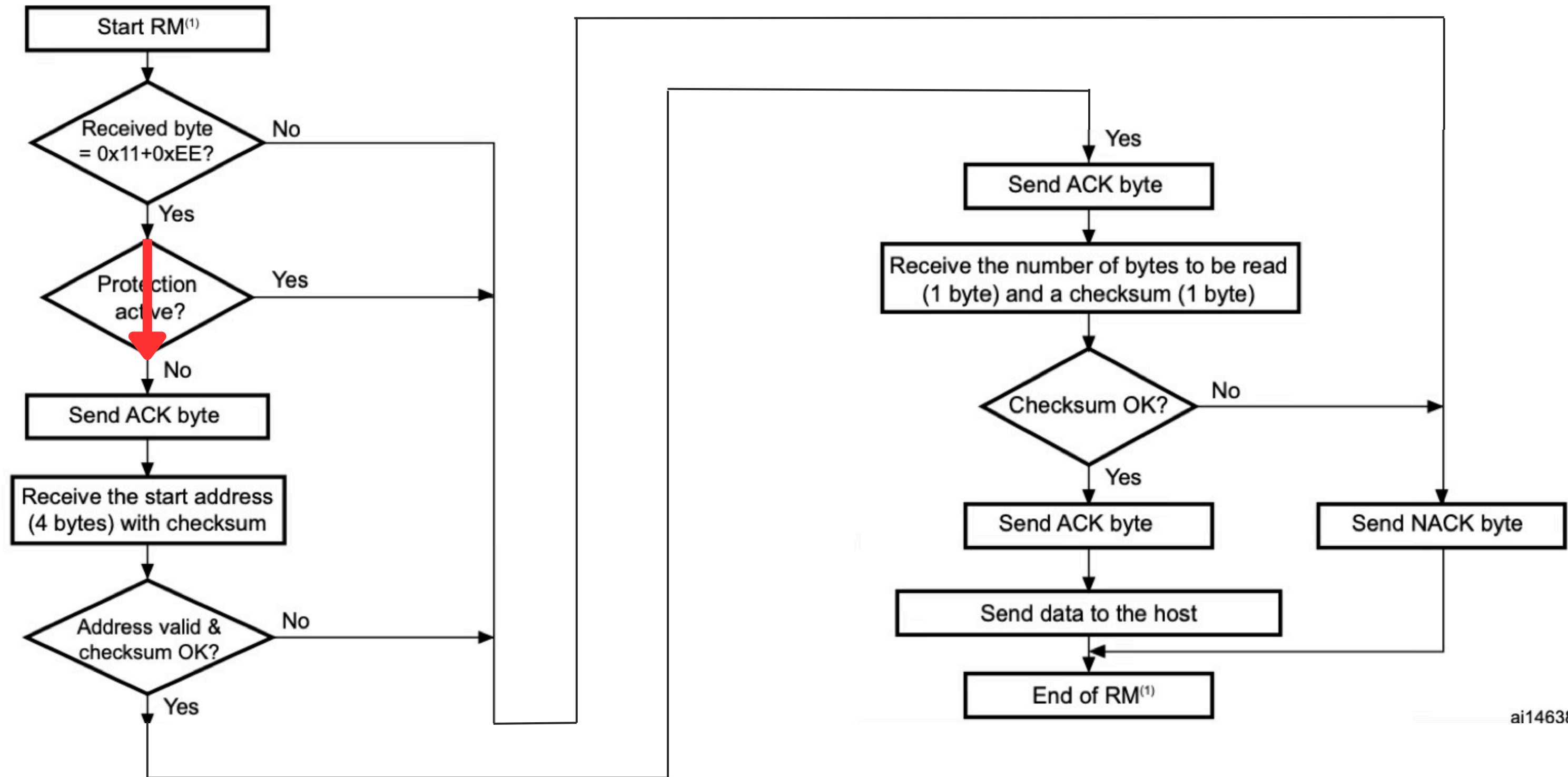


Command ⁽¹⁾	Code	Description
Get ⁽²⁾	0x00	Gets the version and the allowed commands supported by the current version of the protocol.
Get Version ⁽²⁾	0x01	Gets the protocol version.
Get ID ⁽²⁾	0x02	Gets the chip ID.
Read Memory ⁽³⁾	0x11	Reads up to 256 bytes of memory starting from an address specified by the application.
Go ⁽³⁾	0x21	Jumps to user application code located in the internal flash memory or in the SRAM.
Write Memory ⁽³⁾	0x31	Writes up to 256 bytes to the RAM or flash memory starting from an address specified by the application.
Erase ⁽³⁾⁽⁴⁾	0x43	Erases from one to all the flash memory pages.
Extended Erase ⁽³⁾⁽⁴⁾⁽⁵⁾	0x44	Erases from one to all the flash memory pages using two-byte addressing mode (available only for USART bootloader v3.0 and higher).
Special ⁽⁵⁾	0x50	Generic command that allows to add new features depending on the product constraints, without adding a new command for every feature.
Extended Special ⁽⁵⁾	0x51	Generic command that allows the user to send more data compared to the Special command.
Write Protect ⁽⁵⁾	0x63	Enables the write protection for some sectors.
Write Unprotect ⁽⁵⁾	0x73	Disables the write protection for all flash memory sectors.
Readout Protect	0x82	Enables the read protection.
Readout Unprotect ⁽²⁾	0x92	Disables the read protection.
Get Checksum ⁽⁵⁾	0xA1	Computes a CRC value on a given memory area with a size multiple of 4 bytes.

1. If a denied command is received or an error occurs during the command execution, the bootloader sends an NACK byte, and goes back to command checking.

2. Protection. When active, only this limited subset of commands is available, all others are NACK-ed, and have no effect on the device. The protection depends upon the product.

STM32 BOOTLOADER: READ MEMORY



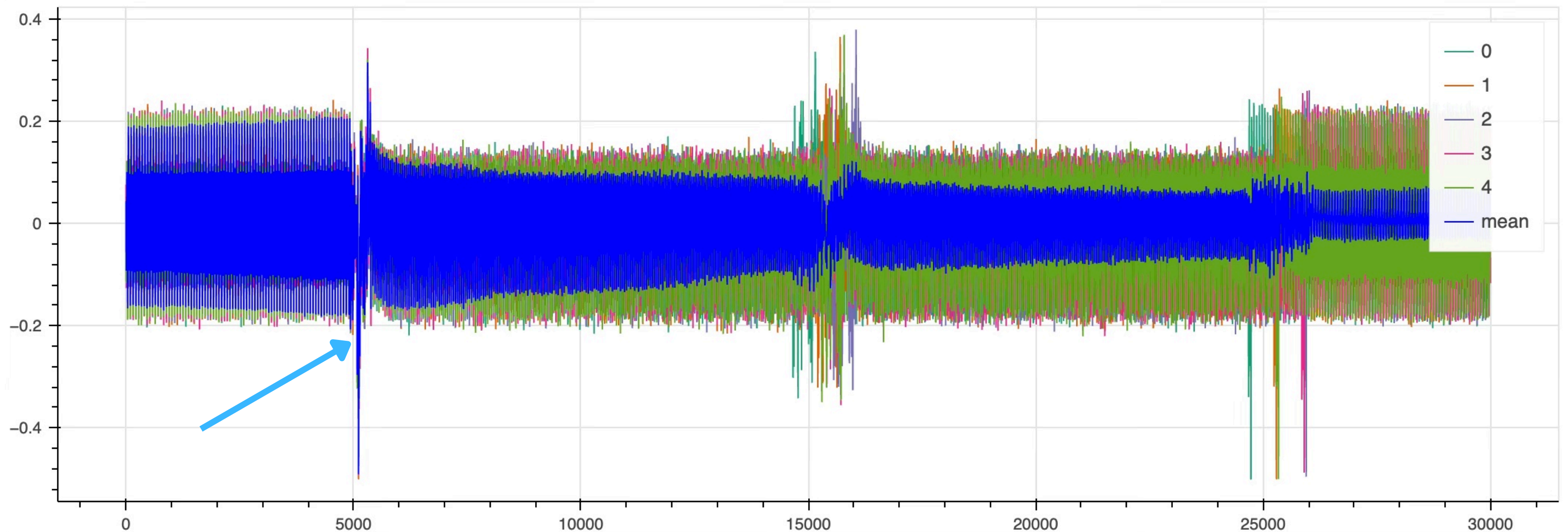
STM32 BOOTLOADER: TRIGGERING

- Jitter due to UART, clocks not synchronized
- Risk of option byte corruption and/or wipe!



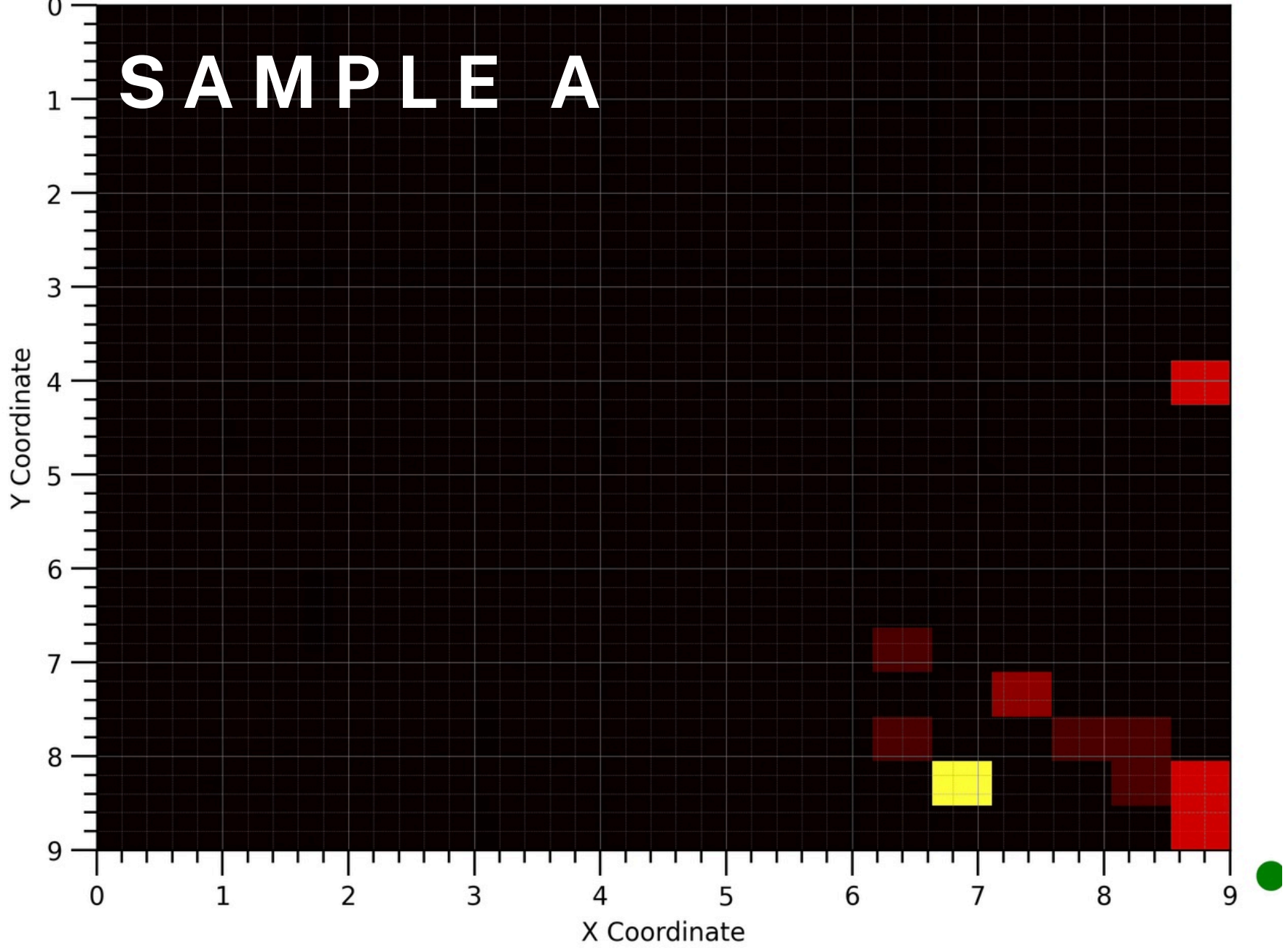
STM32 BOOTLOADER: POWER ANALYSIS

- Represents exactly when the chip is processing command
- Much better repeatability, consistency across devices

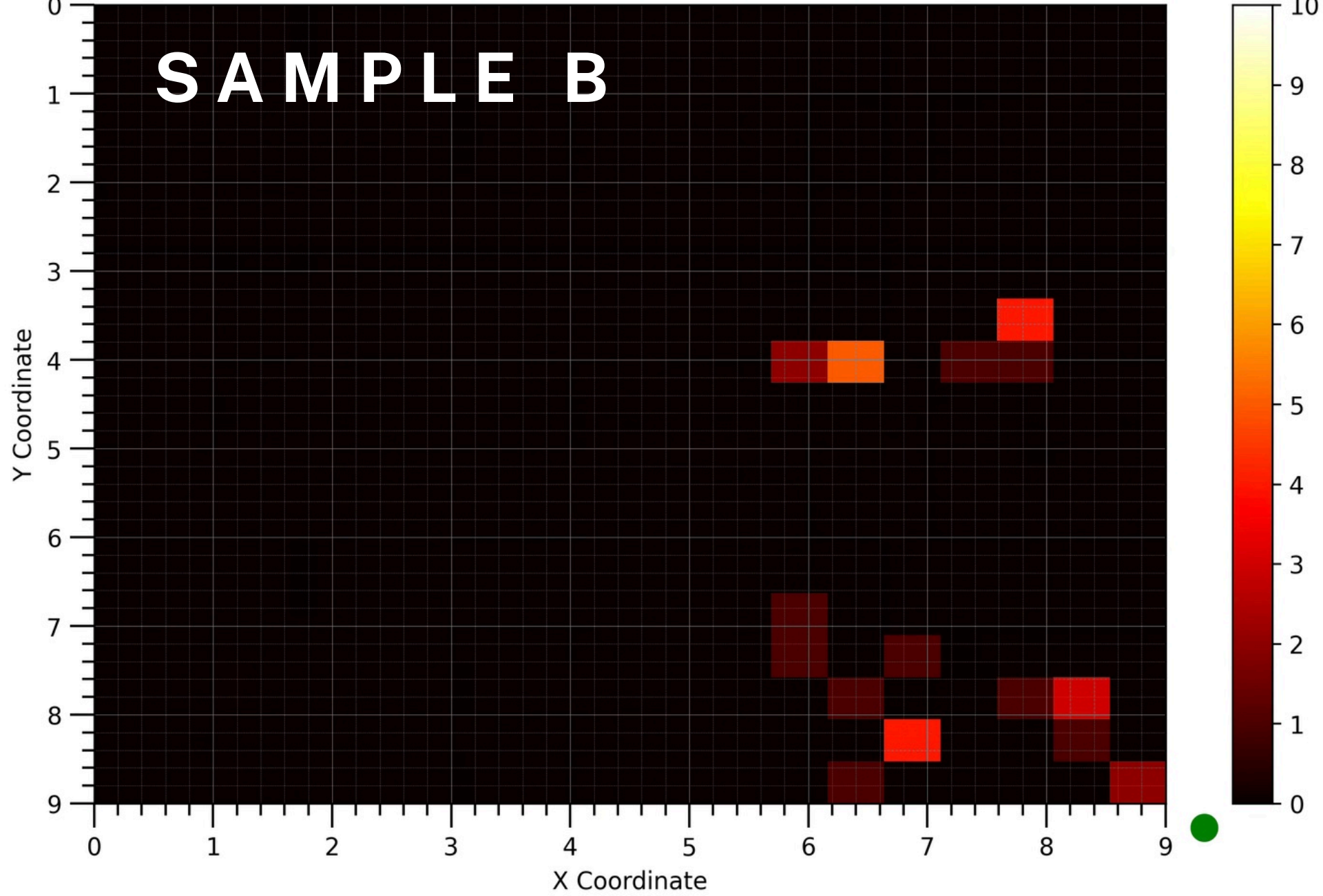


READ MEMORY GLITCH: POSITION (CCW)

STM32F205 BL Command: Fault, 4mm CCW, Step 0.5



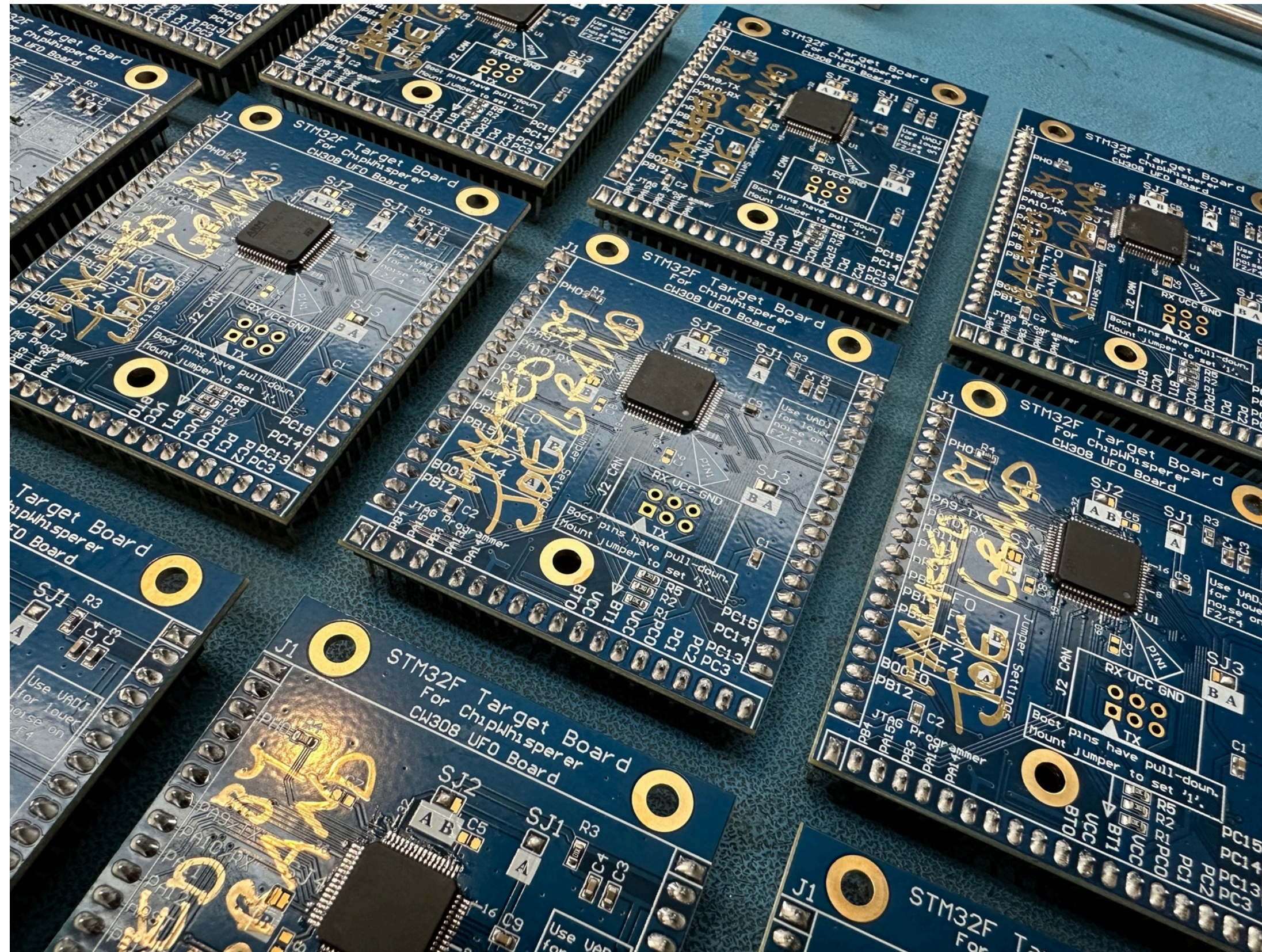
STM32F205 BL Command: Fault, 4mm CCW, Step 0.5



STM32F2 TAKEAWAYS

- Downgrading the F2 with EMFI
 - Probe location and voltage are important, offset from trigger is not
 - Downgrade offset range is very early, likely before the ROM bootloader even starts
 - We never observed any corruption or wipe while using this technique
- Glitching the read memory command can cause data loss
 - ADC level or SAD triggering is a requirement for consistent timing
 - Careful characterization over multiple target samples
 - For a hardware wallet, we only have to glitch the read memory function a few times to extract the needed data

WISHLIST COMPLETE (F2)



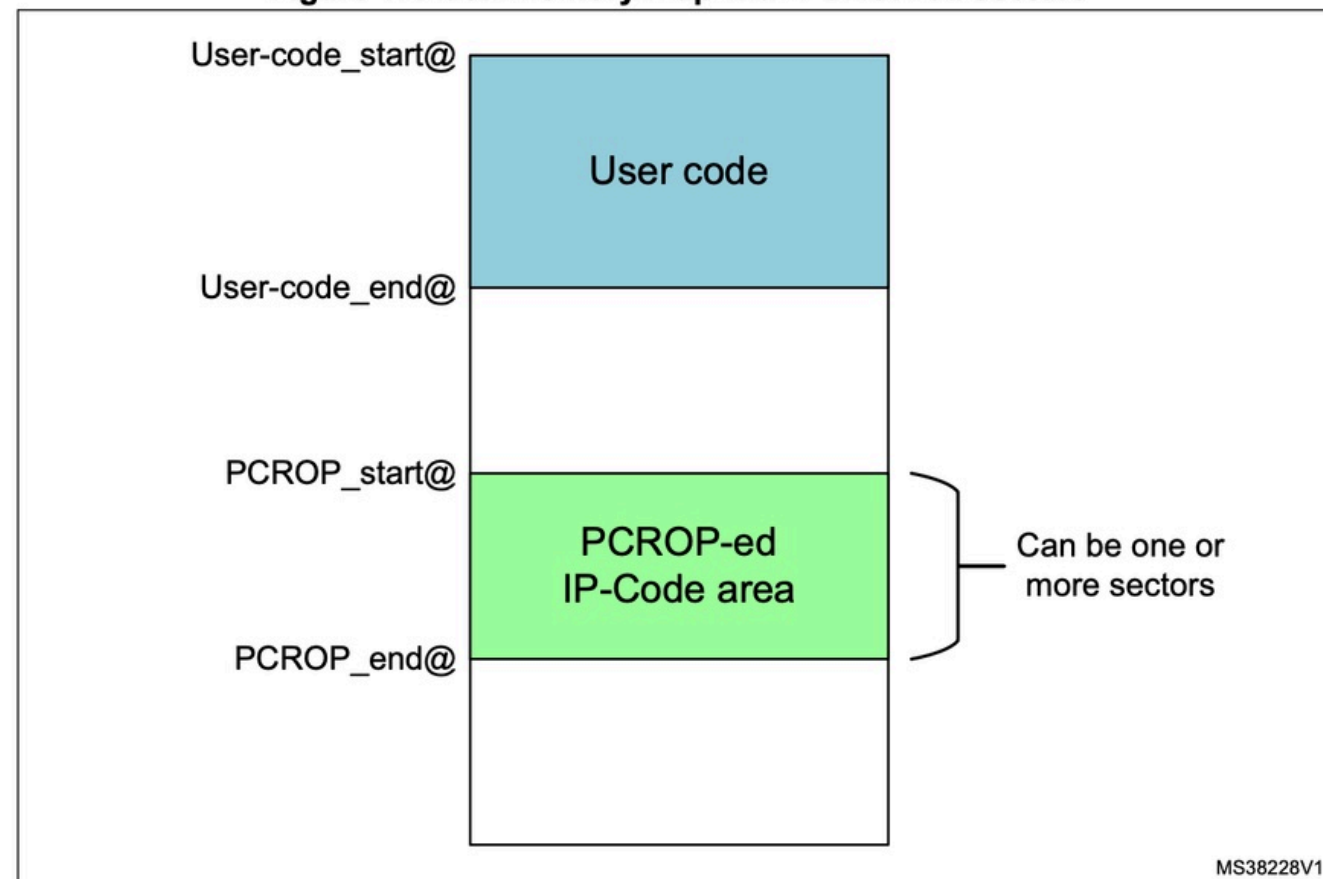
STM32F205 → F427

- Difficulties with downgrading the STM32F427
 - Early downgrade didn't work
 - Permanent Option Byte corruption
 - Wipes during Read Memory
 - Additional security features: Proprietary Code Read Out Protection (PCROP)
- Developing a reliable downgrade strategy and the move back to VFI
- Creating our previously unpublished attack that allows to extract the full Flash memory using only two faults

PCROP

- Proprietary Code Read Out Protection
- AN4701: Proprietary code read-out protection on microcontrollers of the STM32F4 Series
- Any read attempt of PCROP-ed sectors results in 0xFF
- If PCROP permanently set during fault injection → BAD

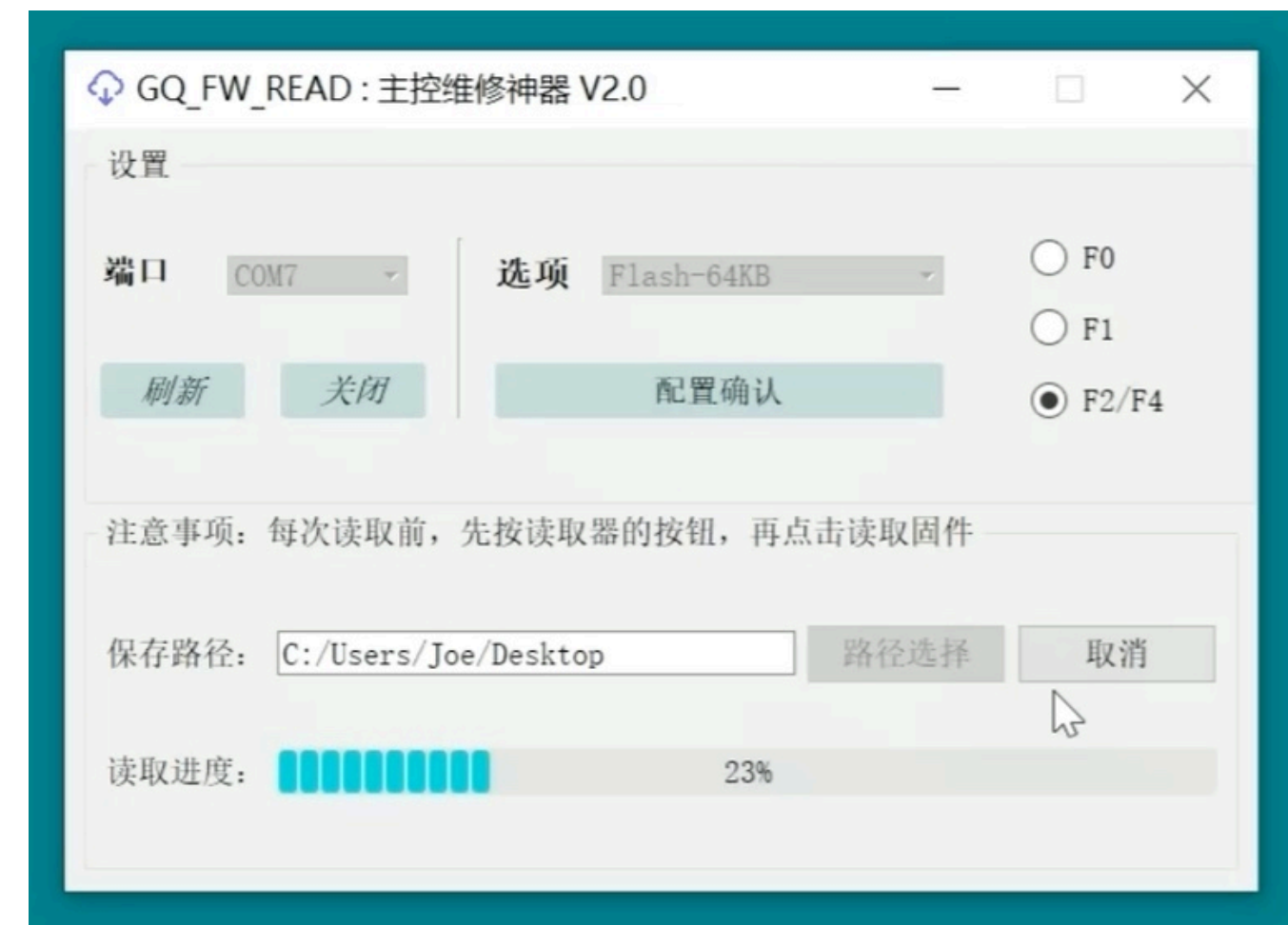
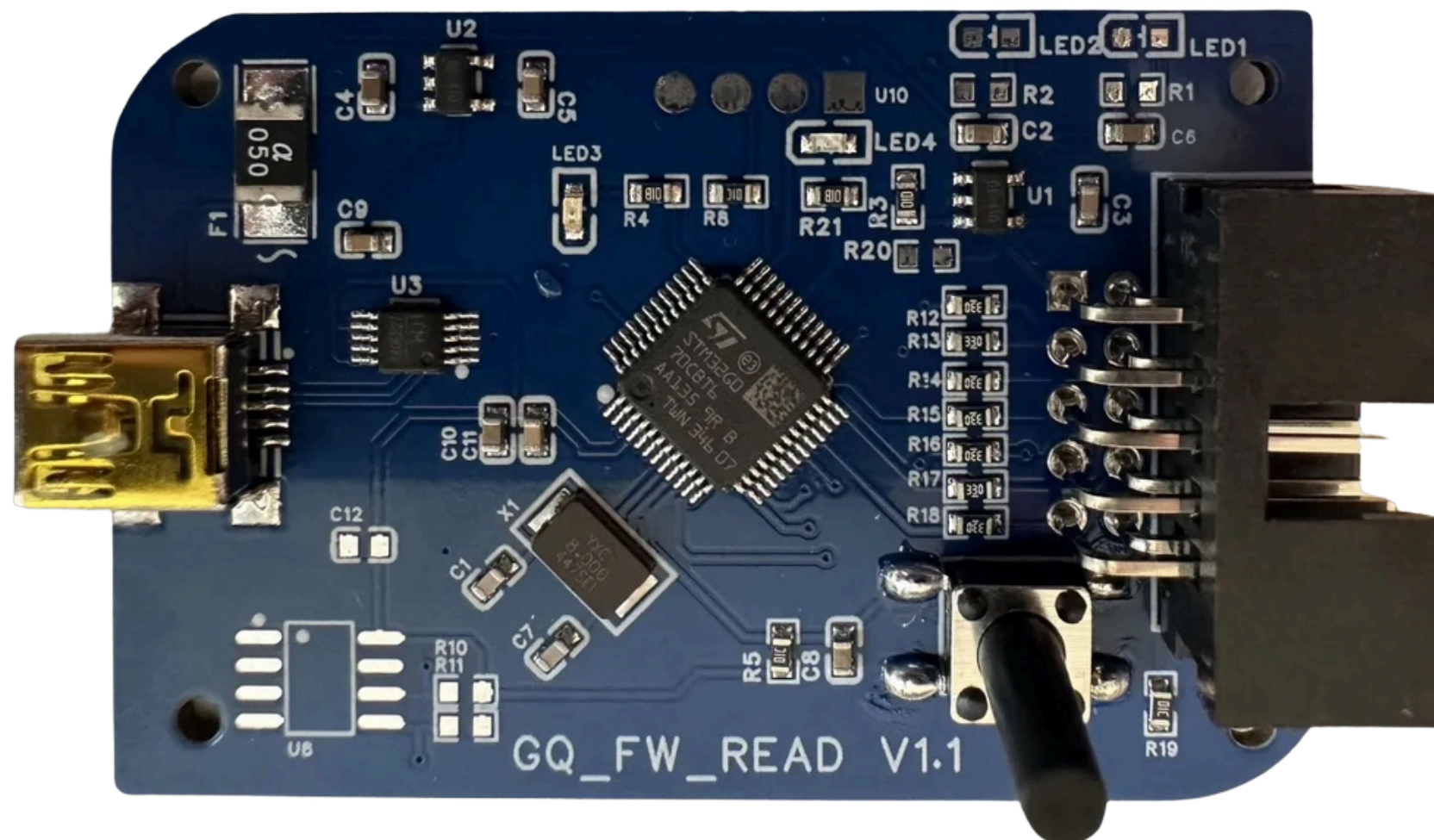
Figure 1. Flash memory map with PCROP-ed sectors



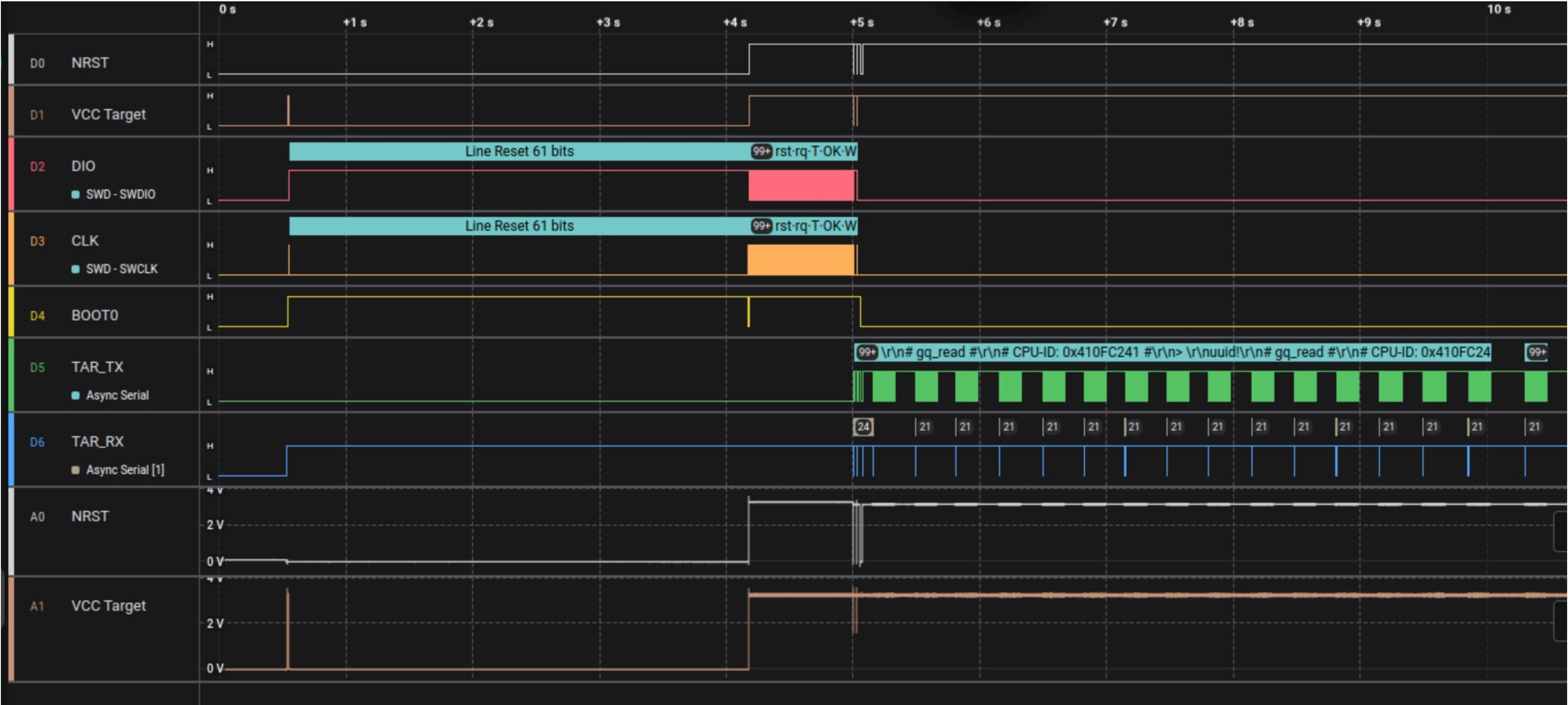
Option bytes (word, address 0x1FFF C008)	
Bit 15	SPRMOD : Selection of protection mode of nWPRi bits 0: nWPRi bits used for sector i write protection (Default) 1: nWPRi bits used for sector i PCROP protection (Sector)
Bit 14	DB1M : Dual bank 1 Mbyte Flash memory devices 0: 1 Mbyte single Flash memory (contiguous addresses in bank 1) 1: 1 Mbyte dual bank Flash memory. The Flash memory is organized as two banks of 512 Kbytes each (see Table 7: 1 Mbyte Flash memory single bank vs dual bank organization (STM32F42xxx and STM32F43xxx) and Table 9: 1 Mbyte dual bank Flash memory organization (STM32F42xxx and STM32F43xxx)). To perform an erase operation, the right sector must be programmed (see Table 7 for information on the sector numbering scheme).
Bits 13:12	0x3: not used
nWRP : Flash memory write protection option bytes for bank 1. Sectors 0 to 11 can be write protected.	
Bits 11:0	nWRPi : If SPRMOD is reset (default value): 0: Write protection active on sector i. 1: Write protection not active on sector i. If SPRMOD is set (active): 0: PCROP protection not active on sector i. 1: PCROP protection active on sector i.

SIDE QUEST: AN STM32 “DECRYPTOR”

- We purchased a commercial STM32 "decryptor" from China (Xianyu / Goofish) advertised to extract flash from RDP1-protected devices
 - We learned about these devices through a talk from Chen Guannan (OSR Security)
 - Reverse engineering this tool gave us an idea for a new approach!



SIDE QUEST: AN STM32 “DECRYPTOR”



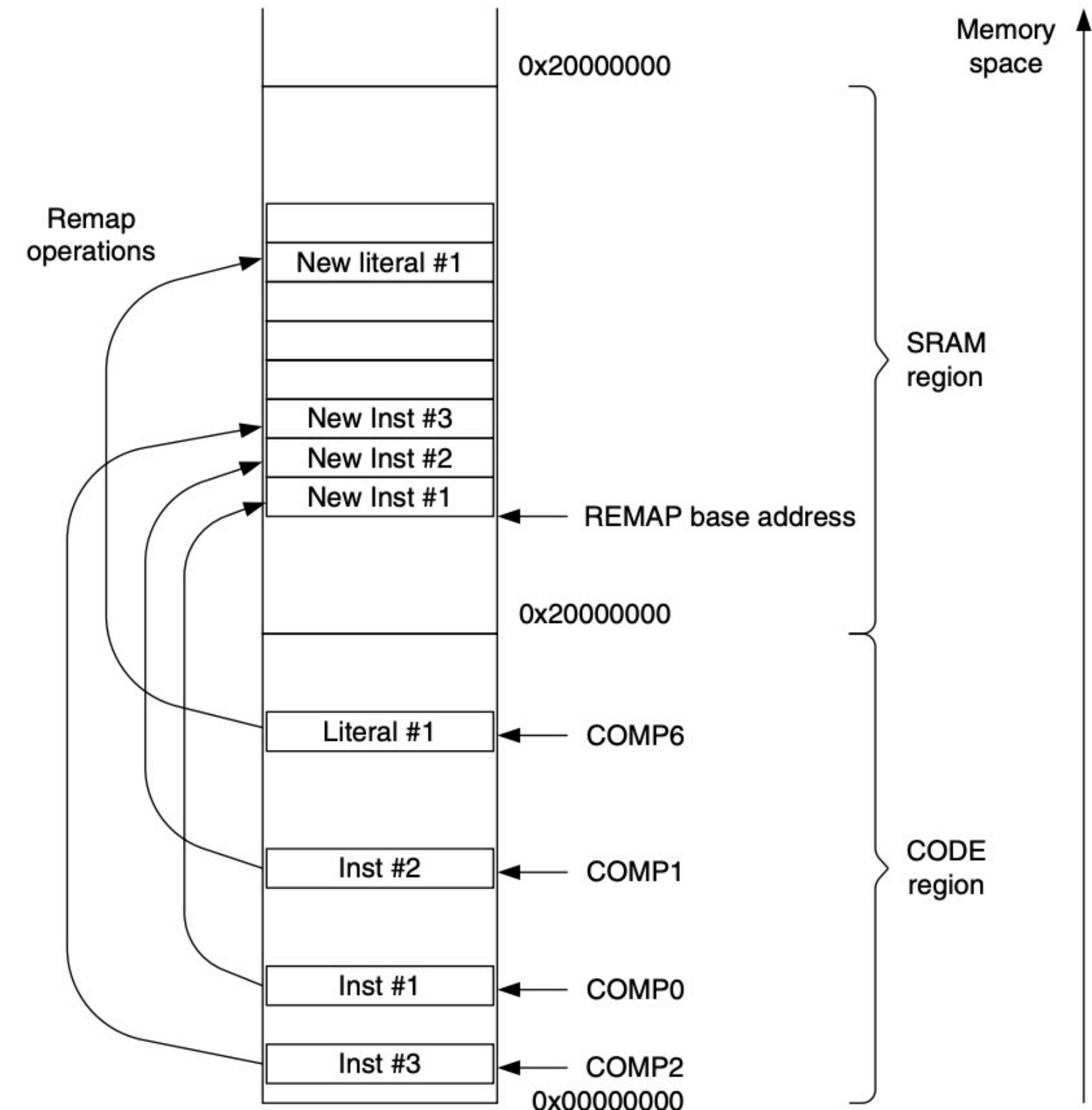
SIDE QUEST: AN STM32 “DECRYPTOR”

The screenshot displays a debugger interface with the following components:

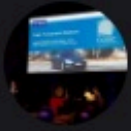
- Program Trees:** Shows the file structure of `payload.bin`, including `sram`, `TIM2_TIM3_TIM4_TIM5_TIM6_TIM7_TIM12_TIM13`, `RTC_WWDG_IWDG_I2S2ext_SPI2_SPI3_I2S4`, `CAN1_CAN2`, `PWR_DAC_UART7_UART8`, `TIM1_TIM8`, `USART1_USART6`, `ADC1`, and `ADC2`.
- Symbol Tree:** Lists symbols including `Imports`, `Exports`, and `Functions` such as `command_parser`, `exception_handler`, `FUN_20000184`, `FUN_2000019c`, `FUN_200001dc`, `FUN_20000434`, and `infiniteLoop`.
- Data Type Manager:** Shows data types including `BuiltInTypes`, `payload.bin`, and `generic_clib`.
- Listing: payload.bin:** Displays assembly code with addresses, hex values, and mnemonics. Key instructions include `ldr r2, [r3, #0x0] => Peripherals::RCC`, `lsrs r2, r2, #0x19`, `beq LAB_200004e2`, `ldr r3, [r1, #0x0] => RCC.CFGR`, `orr r3, r3, #0x2`, `str r3, [r1, #0x0] => RCC.CFGR`, `movs r4, #0x0`, `LAB_200004f2`, `ldr r3, [sp, #0x0] => local_18`, `cmp r3, #0x2`, `bne LAB_200005c4`, `ldr r2, [DAT_20000614]`, `ldr r4, [DAT_20000618]`, `ldr r3, [r2, #0x0] => RCC.AHB1ENR`, `ldr r5, [DAT_2000061c]`, `orr r3, r3, #0x1`, `str r3, [r2, #0x0] => RCC.AHB1ENR`, `ldr r3, [r2, #0x14] => RCC.APB2ENR`, `orr r3, r3, #0x10`, `str r3, [r2, #0x14] => RCC.APB2ENR`, `ldr r3, [DAT_20000620]`, `str r3, [r0, #0x0] => DAT_200003004`, `ldr r3, [DAT_20000624]`, `ldr r0 => s_gq_read_#_CPU-ID: 0x_2000080e, [D`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `bic r2, r2, #0xc0000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `orr r2, r2, #0x80000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `bic r2, r2, #0x300000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [r3, #0x0] => Peripherals::GPIOA`, `orr r2, r2, #0x200000`, `str r2, [r3, #0x0] => Peripherals::GPIOA`, `ldr r2, [DAT_2000062c]`, `ldr r3, [r2, #0x0] => GPIOA.OTYPER`, `bic r3, r3, #0x200`, `str r3, [r2, #0x0] => GPIOA.OTYPER`, `ldr r3, [r2, #0x4] => GPIOA.OSPEEDR`, `orr r3, r3, #0xc0000`, `str r3, [r2, #0x4] => GPIOA.OSPEEDR`, `ldr r3, [r2, #0x8] => GPIOA.PUPDR`, `bic r3, r3, #0x300000`, `str r3, [r2, #0x8] => GPIOA.PUPDR`.
- Decompile: main - (payload.bin):** Shows the decompiled C code, including initialization of `uVar1` and `uVar2`, setting of `Peripherals::RCC.AHB1ENR` and `Peripherals::RCC.APB2ENR`, and a loop that configures `GPIOA` and prints the CPU ID and UUID.

FPB

- Flash Patch and Breakpoint
- [ARM Cortex-M3 Technical Reference Manual](#)
- [The Definitive Guide to the ARM Cortex-M3](#)
- Implements hardware breakpoints, patches instruction or data from code space to SRAM
- Allows for in-the-field, live code patches



NEW RDP2 ATTACK METHODOLOGY



lennert 10/22/25, 9:28 AM

I spent some more time on this but had no luck whatsoever. The downgrade part works pretty well even with VFI, and I checked that the main application can still run after the downgrade. Loading and reading back the payload from SRAM works fine. But no matter what power cycle duration I use I cannot get the payload to execute.

I guess we sort of expected this as executing from SRAM would require a power cycle I think that it is likely that the option bytes are also re-evaluated and that we end up back in RDP2, prohibiting boot from SRAM. (edited)

Some things we could still try if we want to make this work:

- Downgrade, load the payload in SRAM, glitch the VCAP just long enough so the core resets (maybe that doesn't reset whatever is in flash_optcr, or doesn't trigger it re-reading option bytes). Right now I'm power cycling VDD, so maybe doing it for just the core domain makes a difference.
- We could try downgrading, loading the payload, doing a short power cycle and immediately glitch again. But we'd have to get lucky and I'd have to look at the bootloader code again to see if that even makes sense.

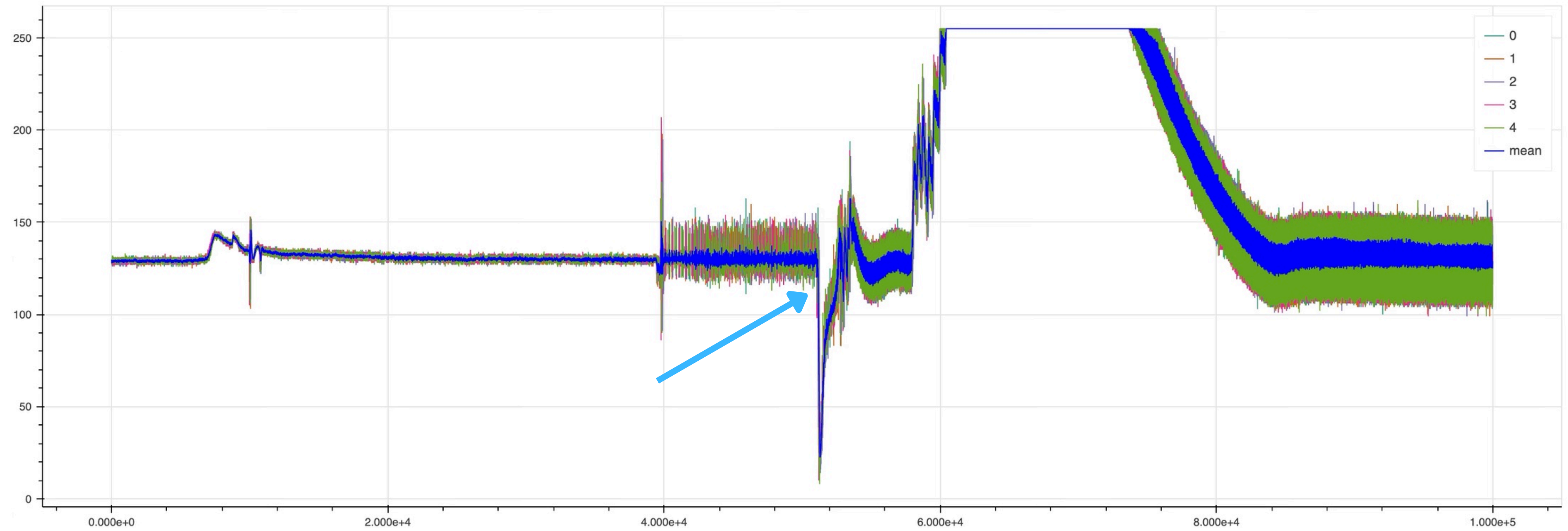
NEW RDP2 ATTACK METHODOLOGY

1. Downgrade RDP2-secured MCU into RDP1
2. Load payload into the now-accessible SRAM
3. Power cycle the chip with BOOT0 = 1 and BOOT1 = 1 to execute code from SRAM
 - Downgrade RDP2-secured MCU into RDP1 (again)
 - SRAM will need to persist!
 - Note: If MCU uses internal oscillator, user code may execute after an unsuccessful downgrade attempt, which could affect the payload in RAM
4. First stage of payload that was previously loaded into SRAM will execute
 - Payload will setup the FPB peripheral to redirect the reset vector to the second stage of the payload in SRAM

NEW RDP2 ATTACK METHODOLOGY

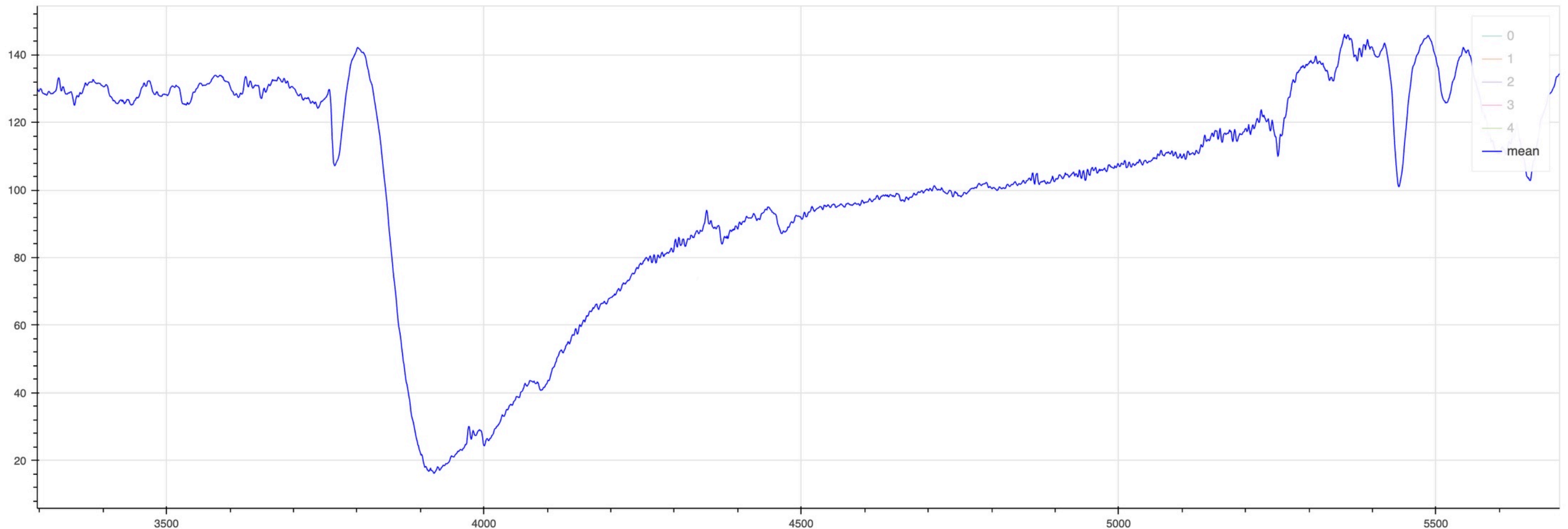
5. Perform a soft reset with $BOOT0 = 0$ and $BOOT1 = X$ to execute code from Flash
6. MCU fetches the reset vector and is redirected to the second stage of the payload in SRAM
 - MCU still thinks it's executing from Flash!
 - It's now in an RDP1 state with Flash memory accessible and the payload executing from SRAM
7. The second payload stage can read all Flash memory

F4 RDP DOWNGRADE: POWER ANALYSIS

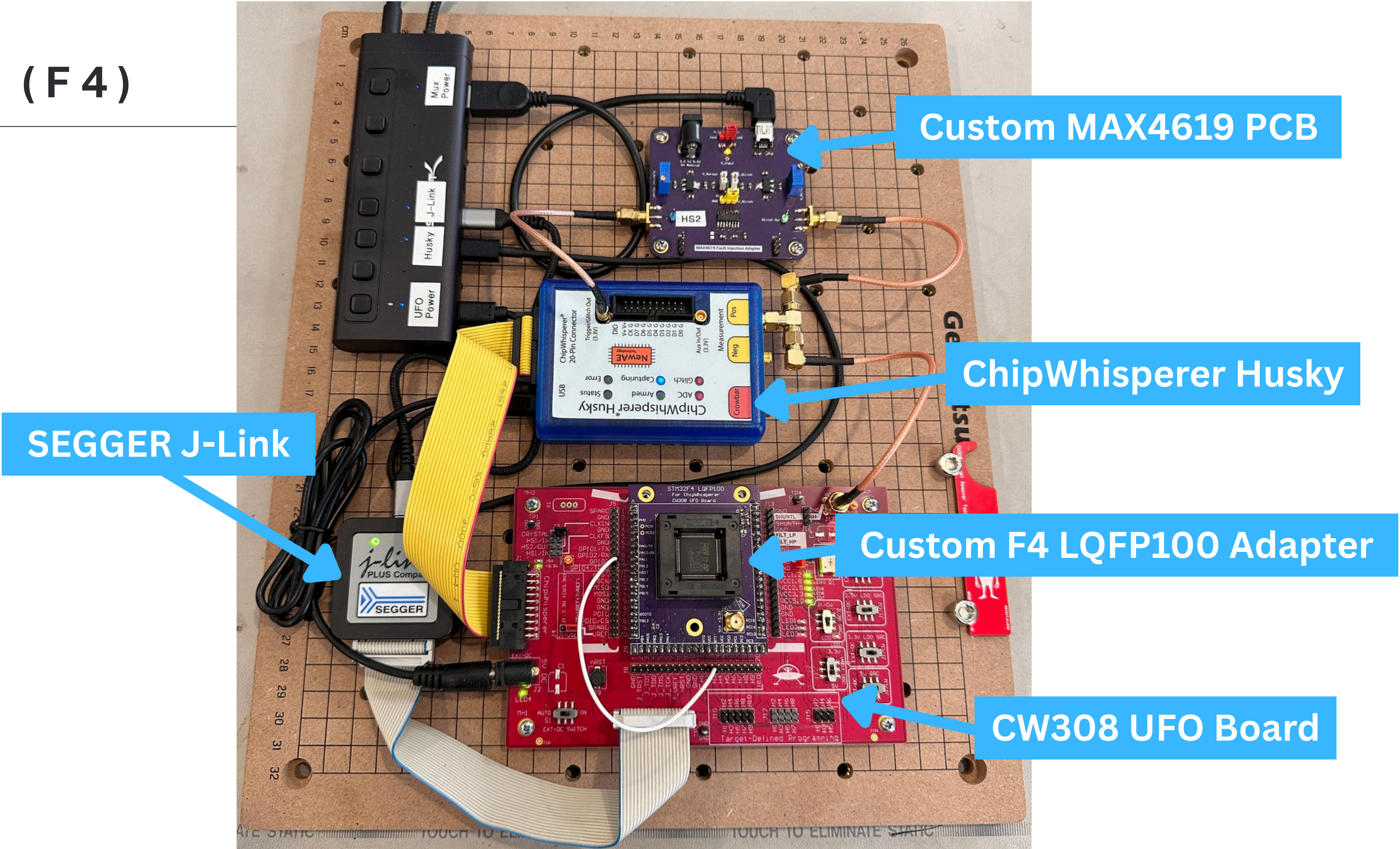


F4 RDP DOWNGRADE: POWER ANALYSIS

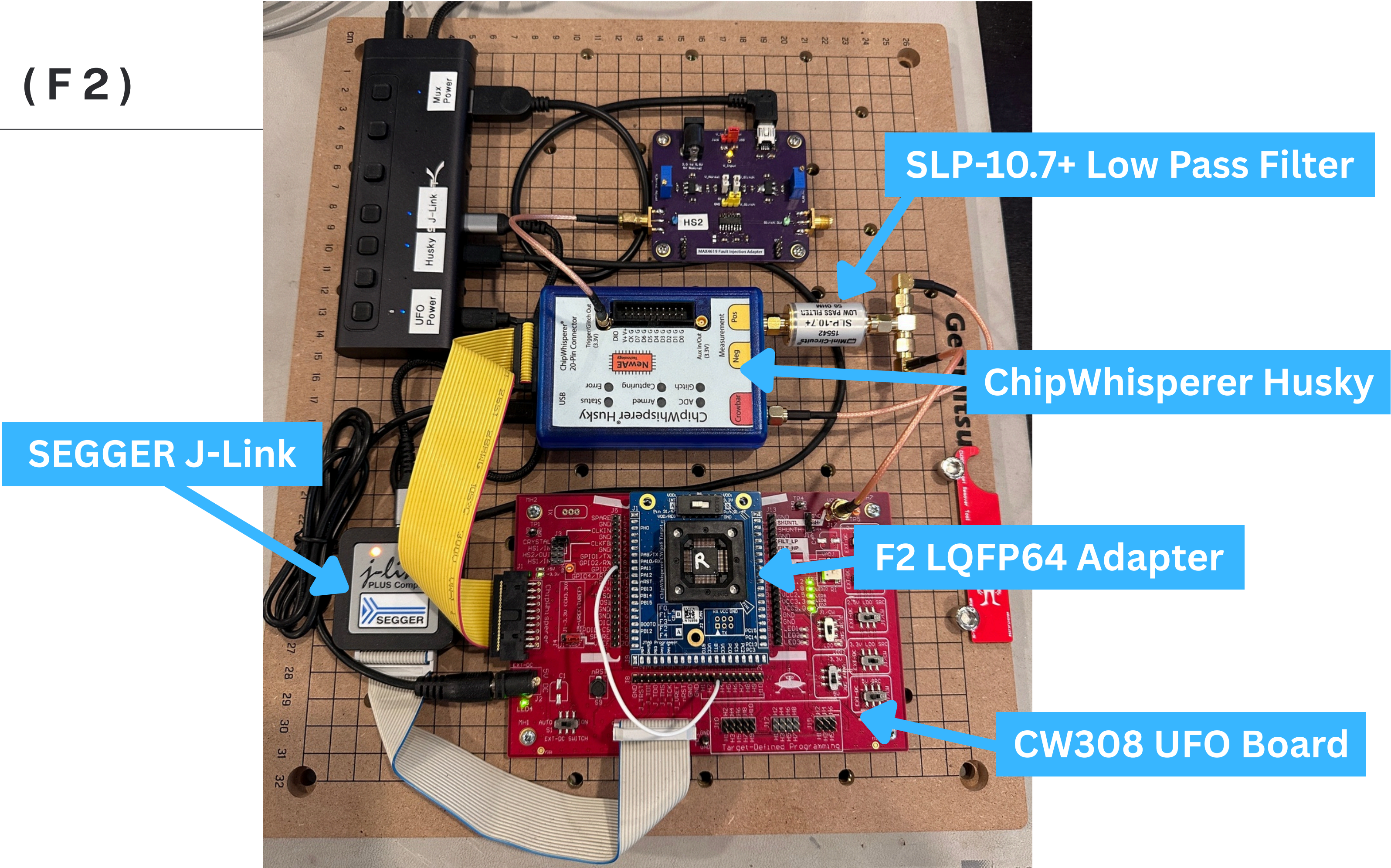
- After NRST trigger, need to sequence with SAD for most reliable triggering
- Finding the best offset requires characterization, staring at side-channel traces and likely a few corrupted STM32F4s



VFI (F4)



VFI (F2)



Stop. Demo Time!




```
from bokeh.layouts import column
from bokeh.palettes import Dark2_5 as palette
import numpy as np
import pylink
import struct
import params
import re
```

Connect to the ChipWhisperer Husky

`ScopeWrapper` is convenience wrapper over the `Scope` class that adds some convenience functions.

```
[*]: scope = ScopeWrapper()
```

Connect to J-Link

```
[ ]: # https://pylink.readthedocs.io/en/latest/
jlink = pylink.JLink()

if jlink.opened() == False:
    jlink.open()

jlink.set_tif(pylink.enums.JLinkInterfaces.JTAG) # Set Target Interface as JTAG

print(jlink.product_name)
```

```
[ ]: # Retrieve IDCODE via low-level JTAG SHIFTR, does not connect to target
def pylink_idcode():
    jlink.open()
    jlink.set_tif(pylink.enums.JLinkInterfaces.JTAG) # Set Target Interface as JTAG
    jlink.jtag_raw_rw([0], [0x1F], 5) # JTAG Reset (TMS = 1 for 5 bits)
    jlink.jtag_raw_rw([0], [0x02], 4) # JTAG ID read (IDCODE should be in DR upon TAP reset)
    result = jlink.jtag_raw_rw([0]*8, [0]*8) # Shift to see what's on scan chain
    hex_id = hex(struct.unpack("<I", bytes(result[0:4]))[0])
    return(hex_id)
```

IDCODE == 0xFFFFFFFF --> Target chip is in RDP2

IDCODE == 0x4ba00477 --> Target chip is in RDP1 or RDP0

```
[ ]: pylink_idcode()
```

3. Setup SAD trigger for downgrade glitch

1. Take a few side-channel measurements during a power cycle, we use the nRST signal as a trigger
2. Select a pattern for SAD triggering

STM32F4 TAKEAWAYS

- Downgrading the F4 (and F2) with VFI and FPB
 - Requires stable triggering and a precise offset
 - Requires identifying “safer” glitching settings
 - We used our custom MAX4619 board
 - Offset is very consistent between different F4 devices
 - We visually determine the offset for each F4 based on side-channel measurements

WISH LIST COMPLETE!

- Complete Flash extraction from an RDP2-secured MCU 
- Near-zero risk of unintentional corruption or data loss 
- Reliable and repeatable 
- Limit the number and type of faults 
- Use accessible/available hardware 
- No semi-invasive or invasive attacks 

EVERYTHING WAS DISCLOSED MARCH 2026

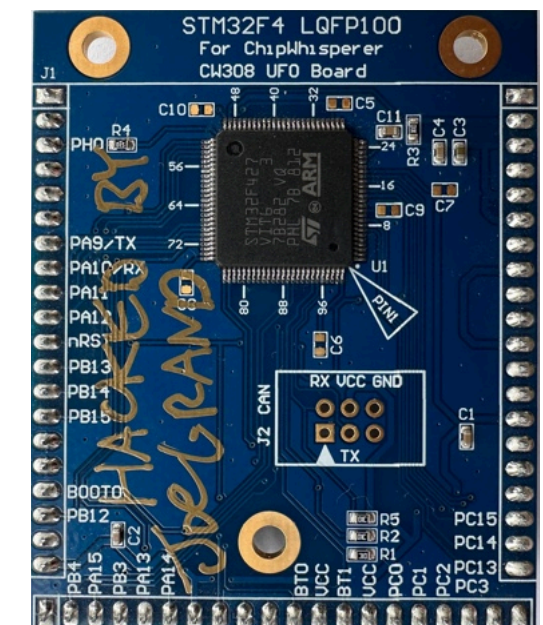
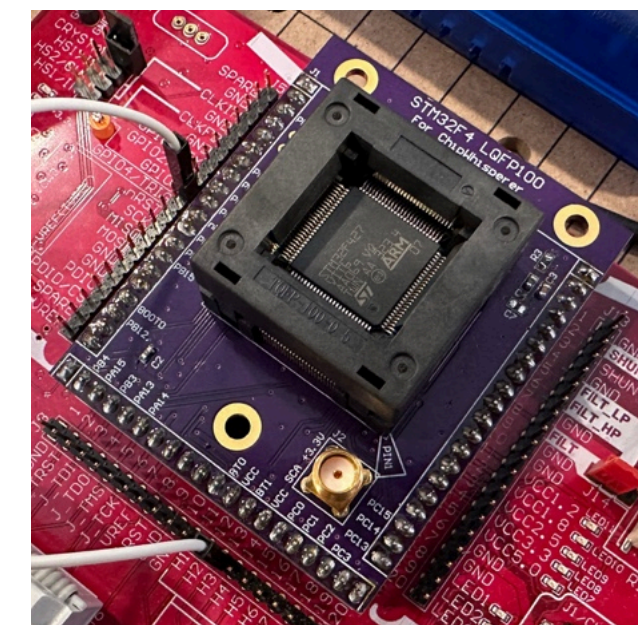
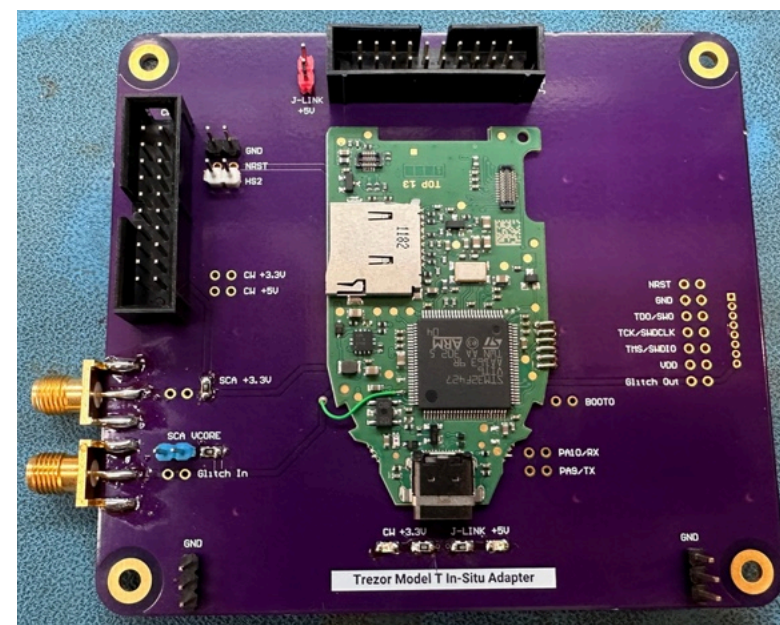
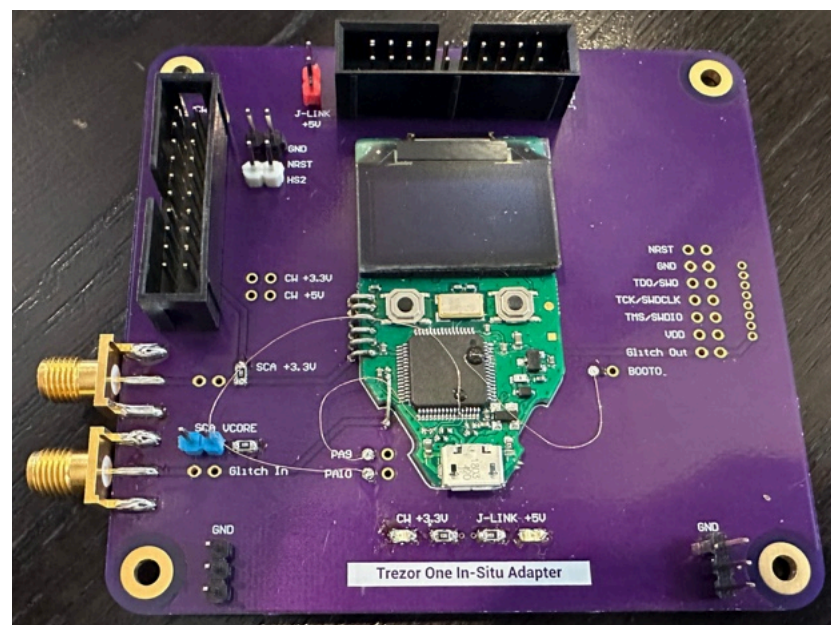
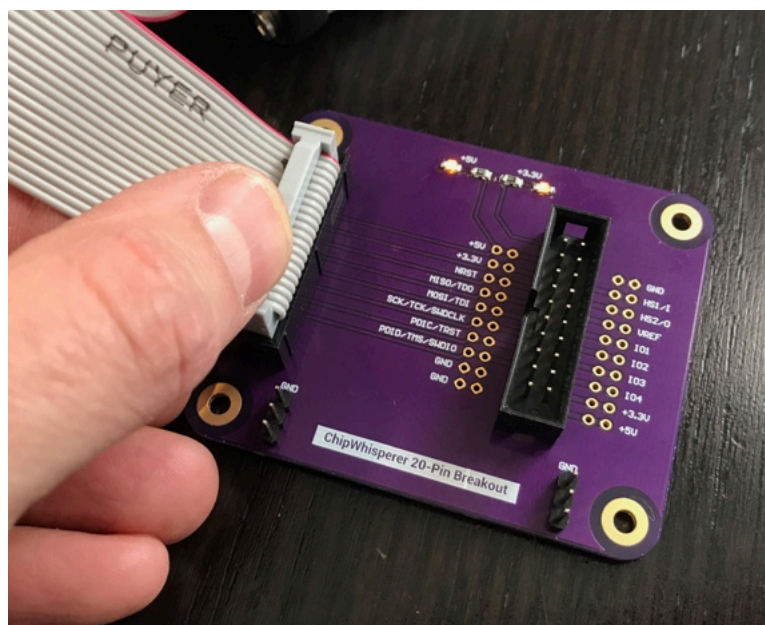
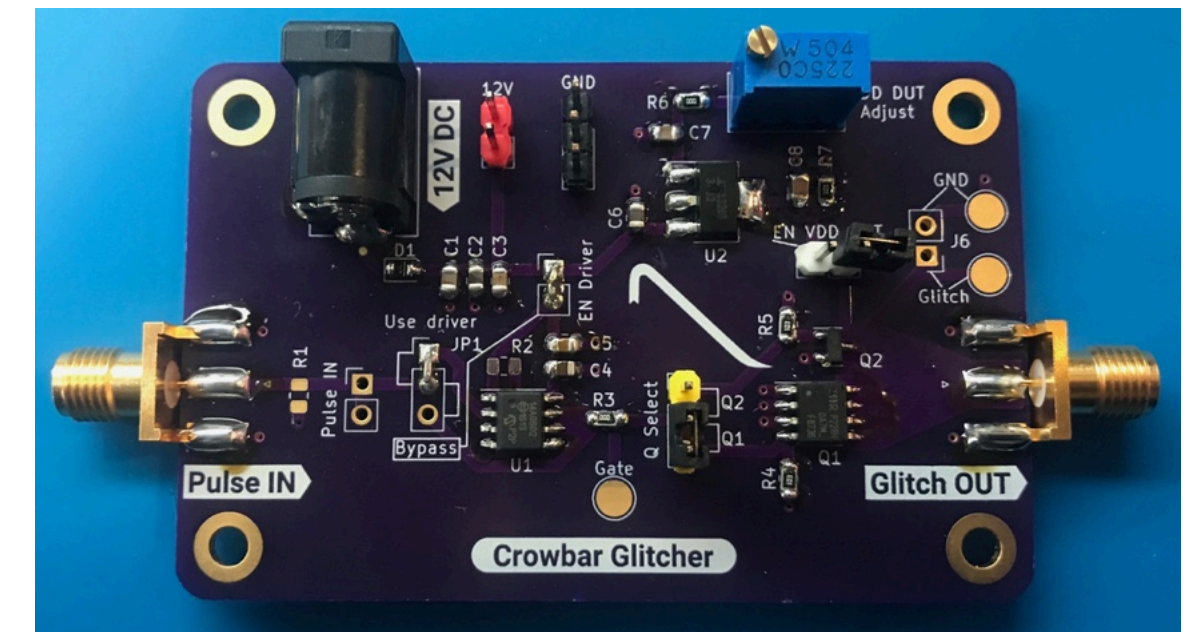
- **ST MICROELECTRONICS (NO RESPONSE)**
- **TREZOR/SATOSHILABS (RESPONSE)**

SOFTWARE RELEASES

- Jupyter notebooks TBD
- <https://github.com/joegrand/>
- <https://grandideastudio.com/portfolio/security/>

HARDWARE DESIGNS

- MAX4619 Analog Switch/Mux
- Crowbar Glitcher
- ChipWhisperer Breakout
- Trezor One In-Situ
- Trezor Model T In-Situ
- STM32F4 LQFP100 UFO Socket
- STM32F4 LQFP100 UFO Non-Socket



BY THE NUMBERS

2
hackers

3.4
years

1
baby

12650
messages

2000+
images/plots

50+
stm32f2

10+
stm32f4

100+ million
total glitches

100%
success rate

Thank you!

FAILURE IS NOT AN OPTION

Joe Grand + Lennert Wouters
REcon 2026, Montreal, Canada