THOMAS SERMPINIS
cr0wsplace.com

THE MATRIX UNLOADED

ESCAPING THE JTAG REALITY

THOMAS SERMPINIS
@cr0wtom

$whoami

- Thomas Sermpinis (@cr0wtom)
  - Technical Director by Day @ Auxilium Pentest Labs
  - Security Researcher by Night @ Cr0w's Place
  - Hack Everything, Everywhere, All at Once (and Legally)

- *Major infosec complainer*
- *Movie and pop-culture afficionado*

- For more: *cr0wsplace.com*

10 YEARS IN THE INDUSTRY

ALMOST NOTHING HAS CHANGED

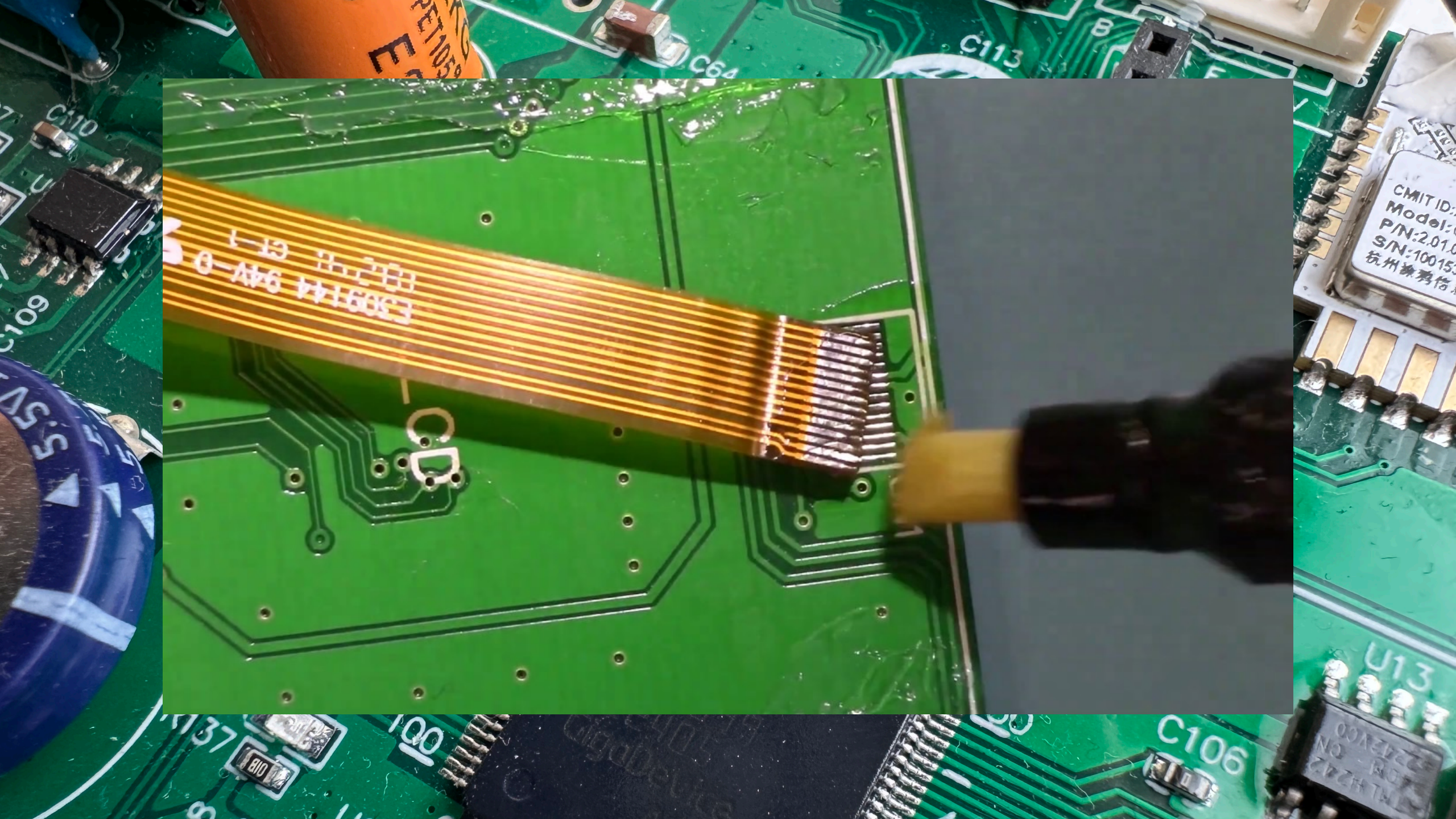
what is this talk about

- Analyze the state of cyber security in the automotive industry
- Talk about all the stupid things the automotive industry is still doing – from a hardware point of view
  - With a sprinkle of forensics in the mix

- Educate the new, the old and the bold
- Lurk more hackers to automotive and hardware security
- Raise and highlight the significance of safety related hardware

[illegible] The State

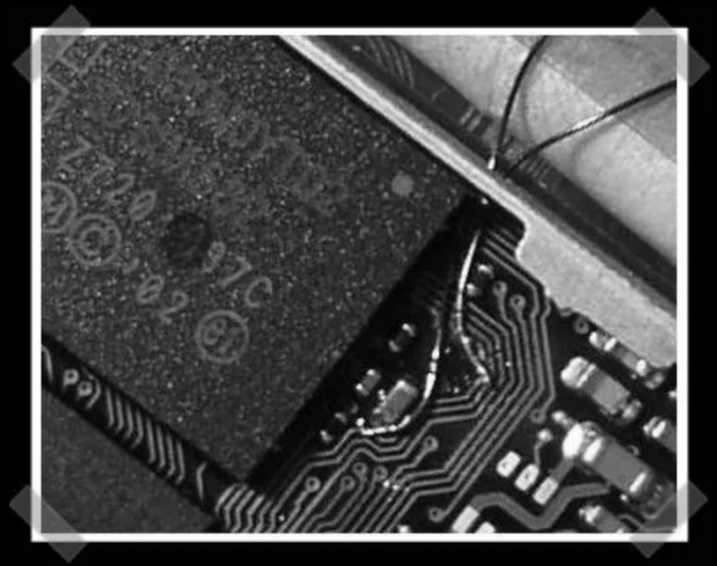

C64
U9
C113
C110
C17
U10
100
CN1
3V3
SWD
CLK
GND
CN12
C109
R126
80
90
100
C105
LED1
R137
50
C106
U13


C110
C109
C64
C113
C106
U13
R137
Q100
CMIT ID:
Model:
P/N:2.01.0
S/N:100153

FREE YOUR IPHONE

August 26, 2007 by Will O'Brien

25 Comments



I wasn't going to post this – it's a freakin phone after all. But I've gotten quite a few tips on it, and I'd like them to end. [George] made a concerted effort to hack the iPhone – and it paid off. After his crazy ebay auction that topped out at 99,999,999.99 last time I checked, he ended up trading his first phone for a Nissan 350z and a few more iPhones.

He documented his process, step by step – if you've got the skills, you can probably do it yourself. The soldering work is damn fine work – probably the hardest thing there is. The write up is a little hard to follow, so plan on taking some time to comprehend everything. (Blogging software isn't the best way to organize how-tos, trust me on this.) My hats off to [George], he did some great work. – So, why didn't I want to post it? All this work yielded one thing: carrier choice for the iPhone.

Posted in handhelds hacks, iphone hacks, ipod hacks, Misc Hacks

Tagged hack, iphone, jtag

FREE YOUR IPHONE

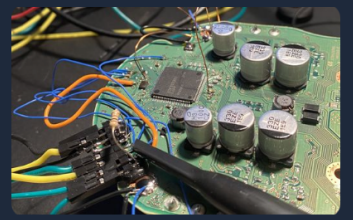
August 26, 2007 by Will O'Brien

25 Comments

Bypassing the Renesas RH850/P1M-E read protection using fault injection

Nov 8, 2022

Using fault injection to bypass the disabled programmer setting on a Renesas RH850/P1M-E automotive microcontroller and access flash memory contents.



Willem Melching

Read more →

He documented his process, step by step – if you've got the skills, you can probably do it yourself. The soldering work is damn fine work – probably the hardest thing there is. The write up is a little hard to follow, so plan on taking some time to comprehend everything. (Blogging software isn't the best way to organize how-tos, trust me on this.) My hats off to [George], he did some great work. – So, why didn't I want to post it? All this work yielded one thing: carrier choice for the iPhone.

Posted in handhelds hacks, iphone hacks, ipod hacks, Misc Hacks
Tagged hack, iphone, jtag


1 05
AM
22°
MUNICH | SUNNY

[redacted] The Problem

Ubiquitous Hardware Vulnerabilities in Automotive

VicOne | ZERO DAY INITIATIVE

PWN2OWN AUTOMOTIVE

January 22-24, 2025

Tokyo, Japan

REGISTER NOW

The Automotive Cybersecurity Event
at the Forefront of Zero-Day Vulnerability Discovery

Target	Cash Prize	Master of Pwn Points
ChargePoint Home Flex (Model CPH50)	$50,000	5
Phoenix Contact CHARX SEC-3150	$50,000	5
WOLFBOX Level 2 EV Charger	$50,000	5
EMPORIA EV Charger Level 2	$50,000	5
Tesla Wall Connector	$50,000	5
Autel MaxiCharger AC Wallbox Commercial (MAXI US AC W12-L-4G)	$50,000	5
Ubiquiti Connect EV Station	$50,000	5


User Manual
WOLFBOX
WOLFBOX
Trouble Code:


WOLFBOX
User Manual
WOLFBOX
Trouble Code:


INOVANCE
WOLFBOX

Hardware Debugging Interfaces

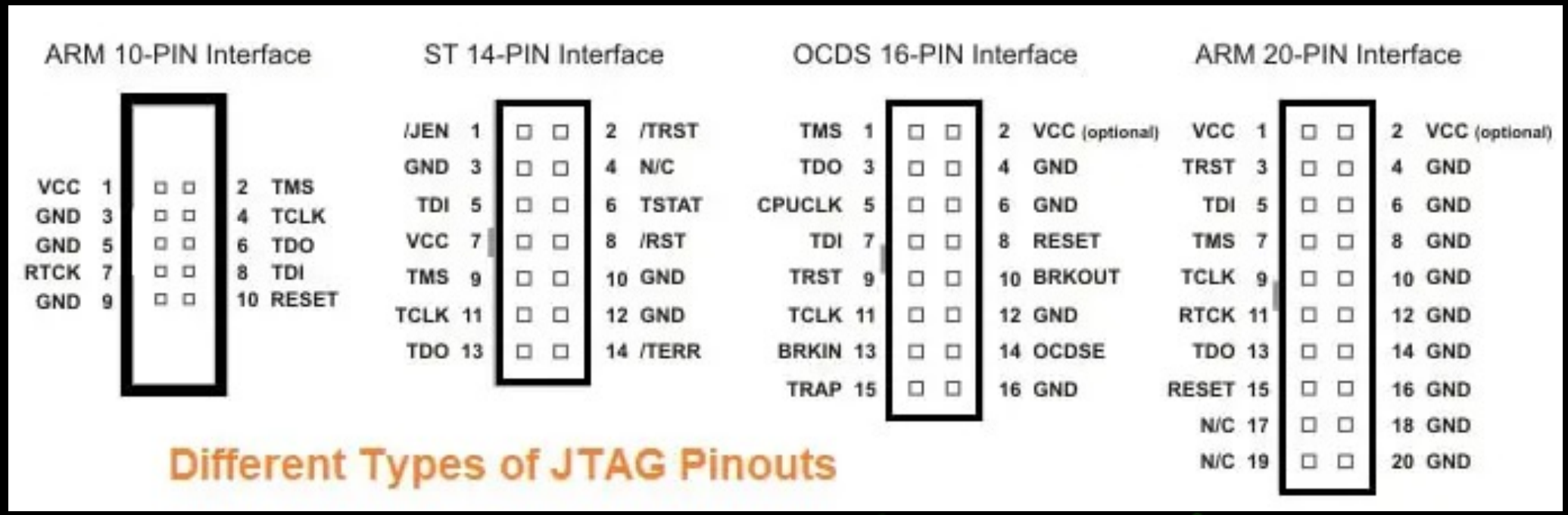
- Direct, low-level access to processor and memory

Hardware Debugging Interfaces

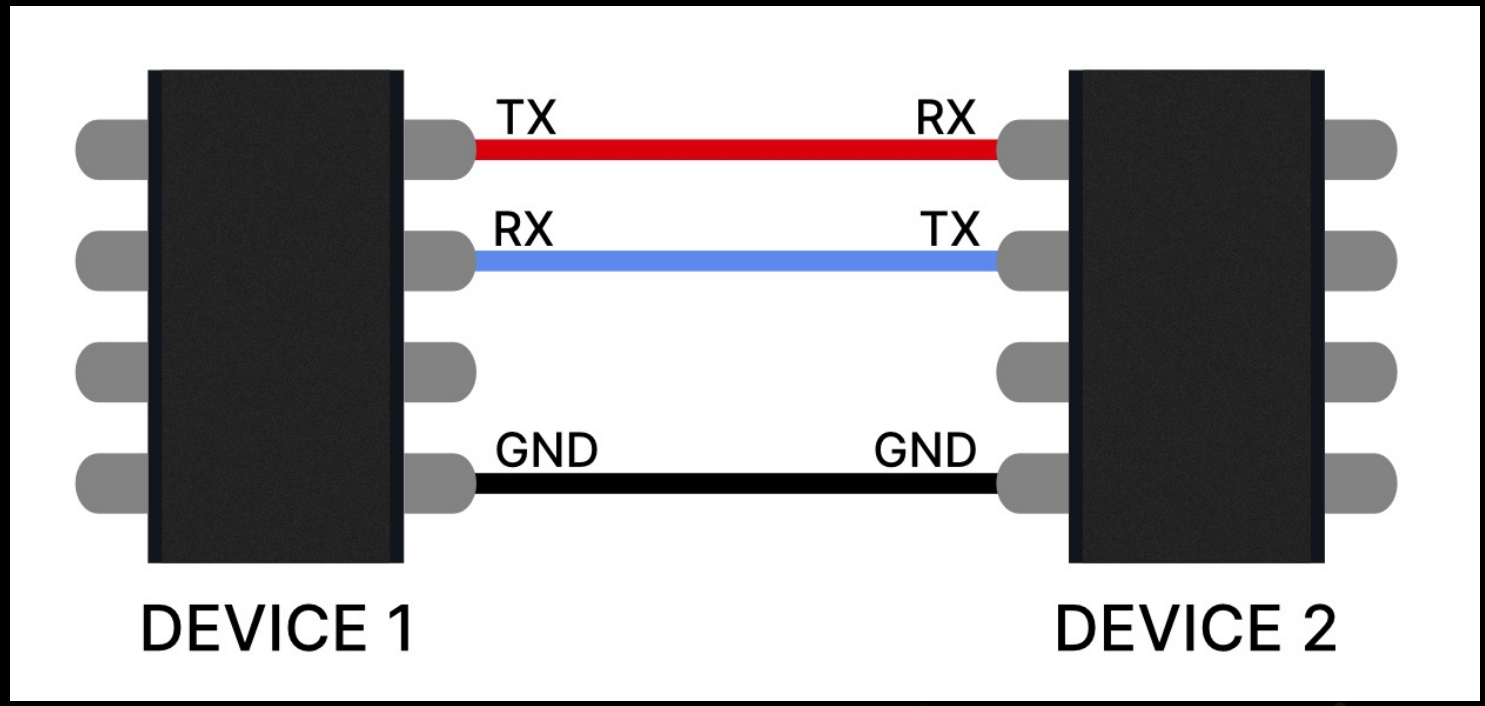
- Direct, low-level access to processor and memory

Interface	Purpose
JTAG	Full debug access
SWD	ARM-specific, lightweight JTAG alternative
UART	Serial console
SPI/I²C	Internal buses

Hardware Debugging Interfaces

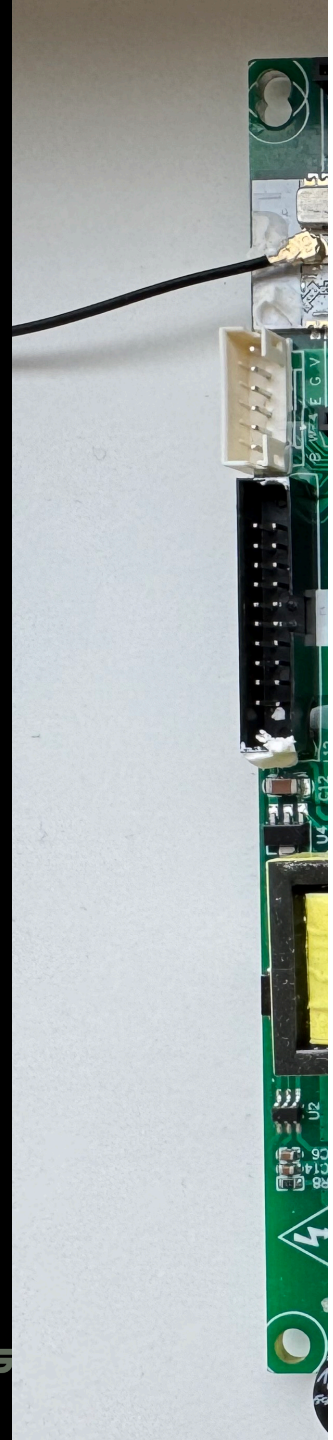


Hardware Debugging Interfaces




QC PASSED JXH
AC50A1
US-50A-BT
Left terminal connection:
L-OUT PE L-IN
Right terminal connection:
N-IN PE N-OUT
ON DIP
1 2 3 4
STOP
Carli


QC PASSED JXH
AC50A1
US-50A-BT
Left terminal connection:
L-OUT PE L-IN
Right terminal connection:
N-IN PE N-OUT
ON DIP
1 2 3 4
STOP
Carli



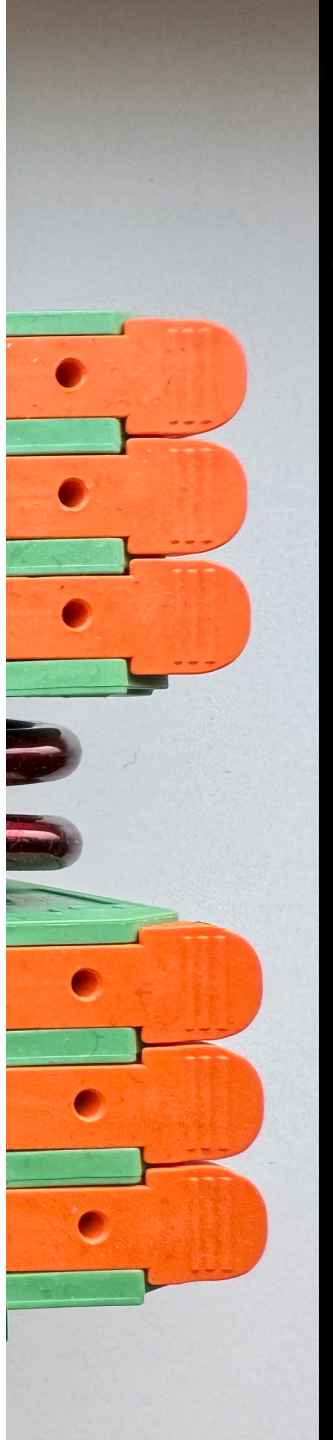
GigaDevice Semiconductor Inc.

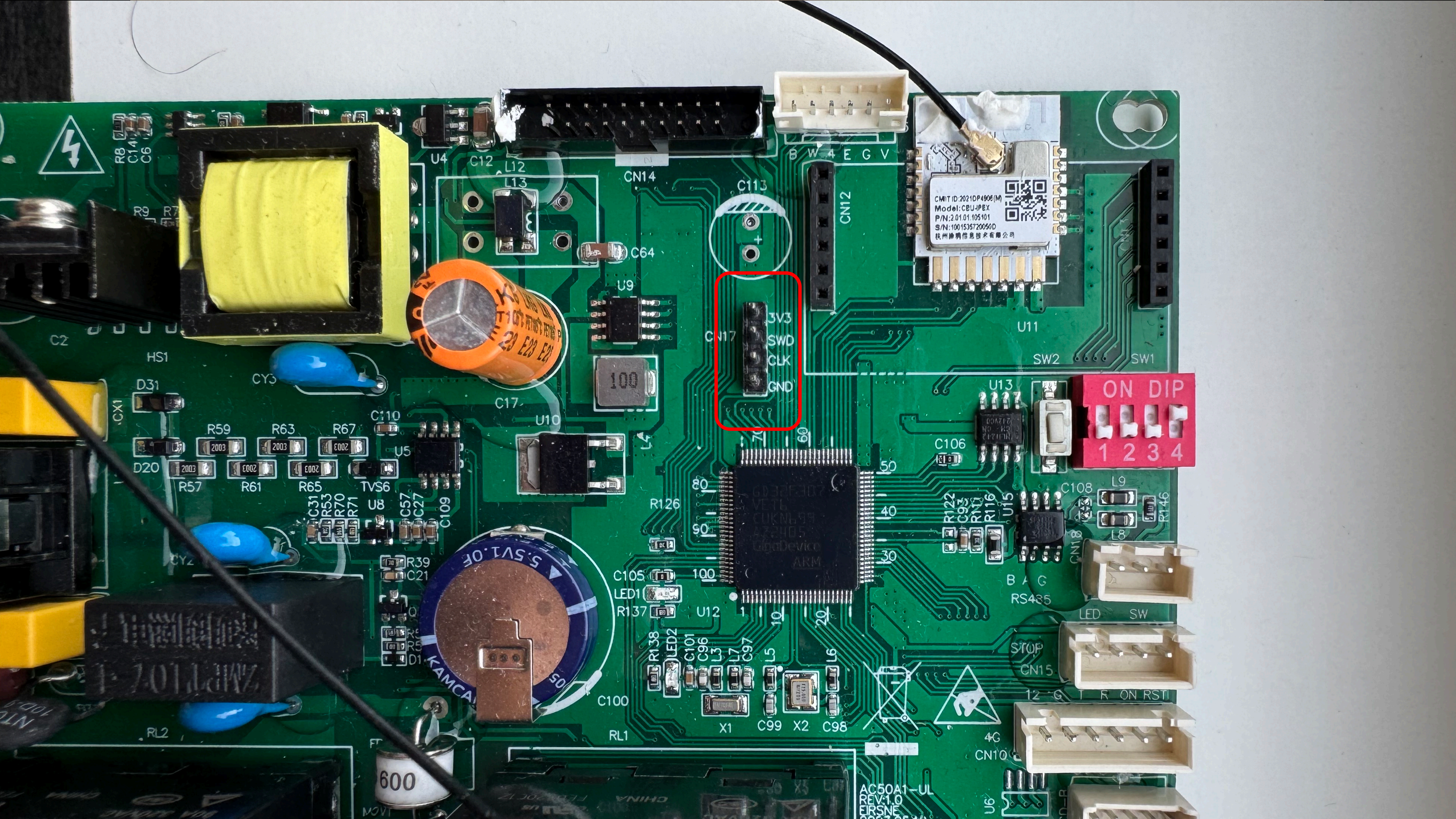
GD32F307xx
Arm® Cortex®-M4 32-bit MCU

Datasheet

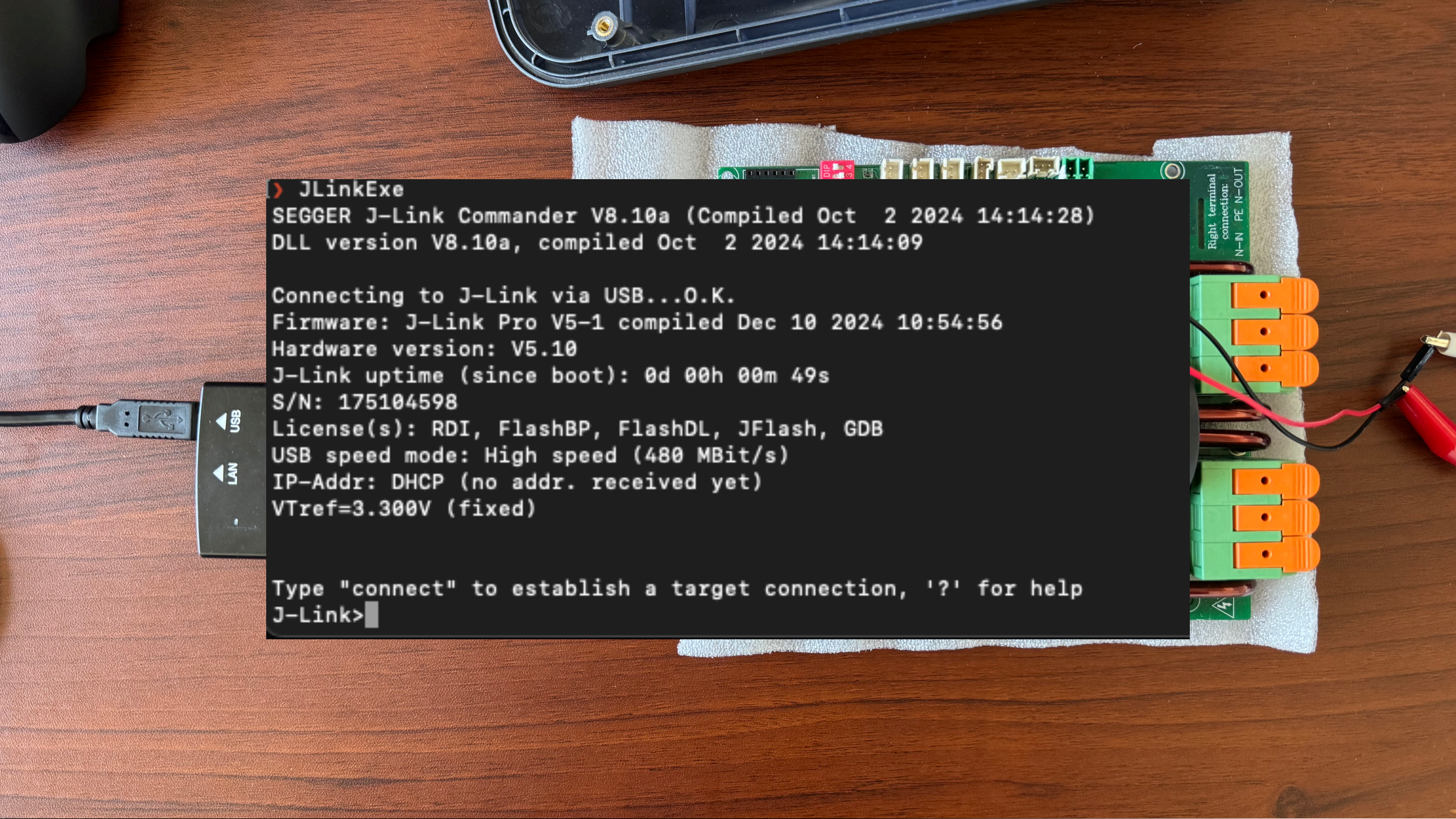
Revision 1.6

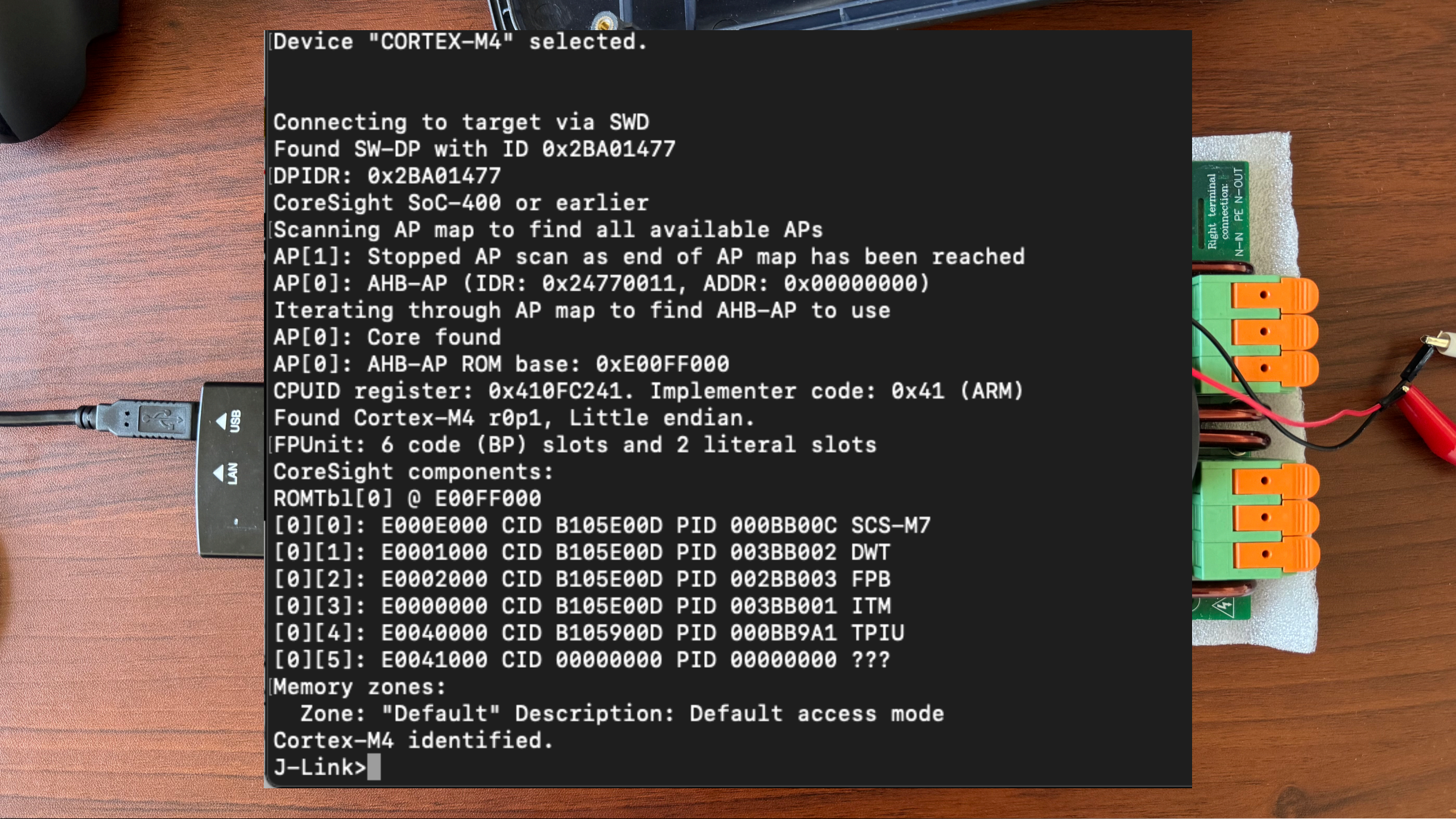
(Sep. 2023)

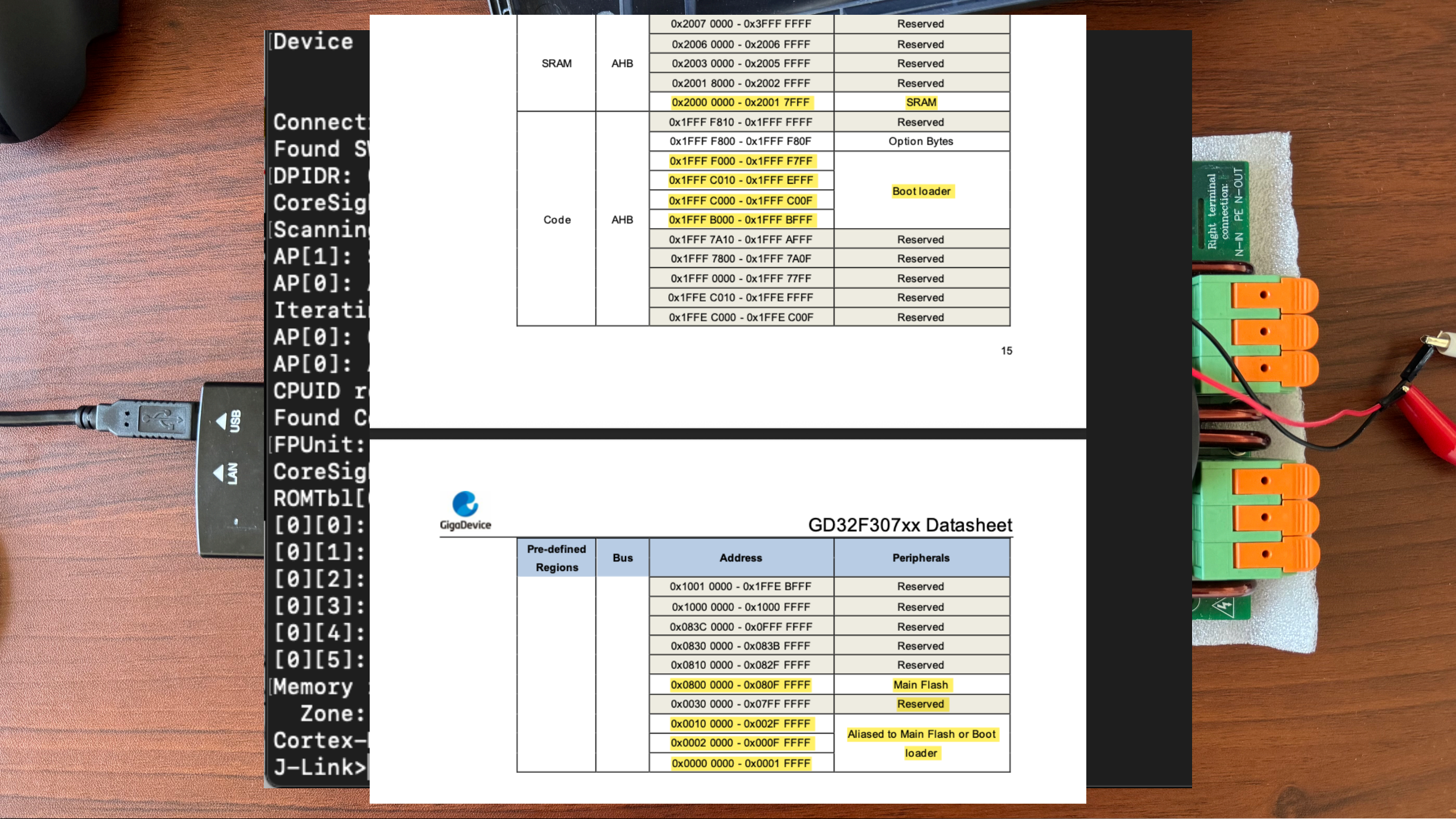



CN17
3V3
SWD
CLK
GND
CN14
CN12
C113
C64
U9
U10
U11
U13
SW1
SW2
ON DIP
1 2 3 4
RS485
CN15
CN10
X1
X2
C99
C98
C100
RL1
RL2
GigaDevice
ARM
AC50A1-UL
REV:1.0


LAN
USB
j-link Pro
SEGGER
www.segger.com
Target
ON DIP
1 2 3 4
AC50A1
Left terminal connection:
L-OUT PE L-IN
Right terminal connection:
N-IN PE N-OUT

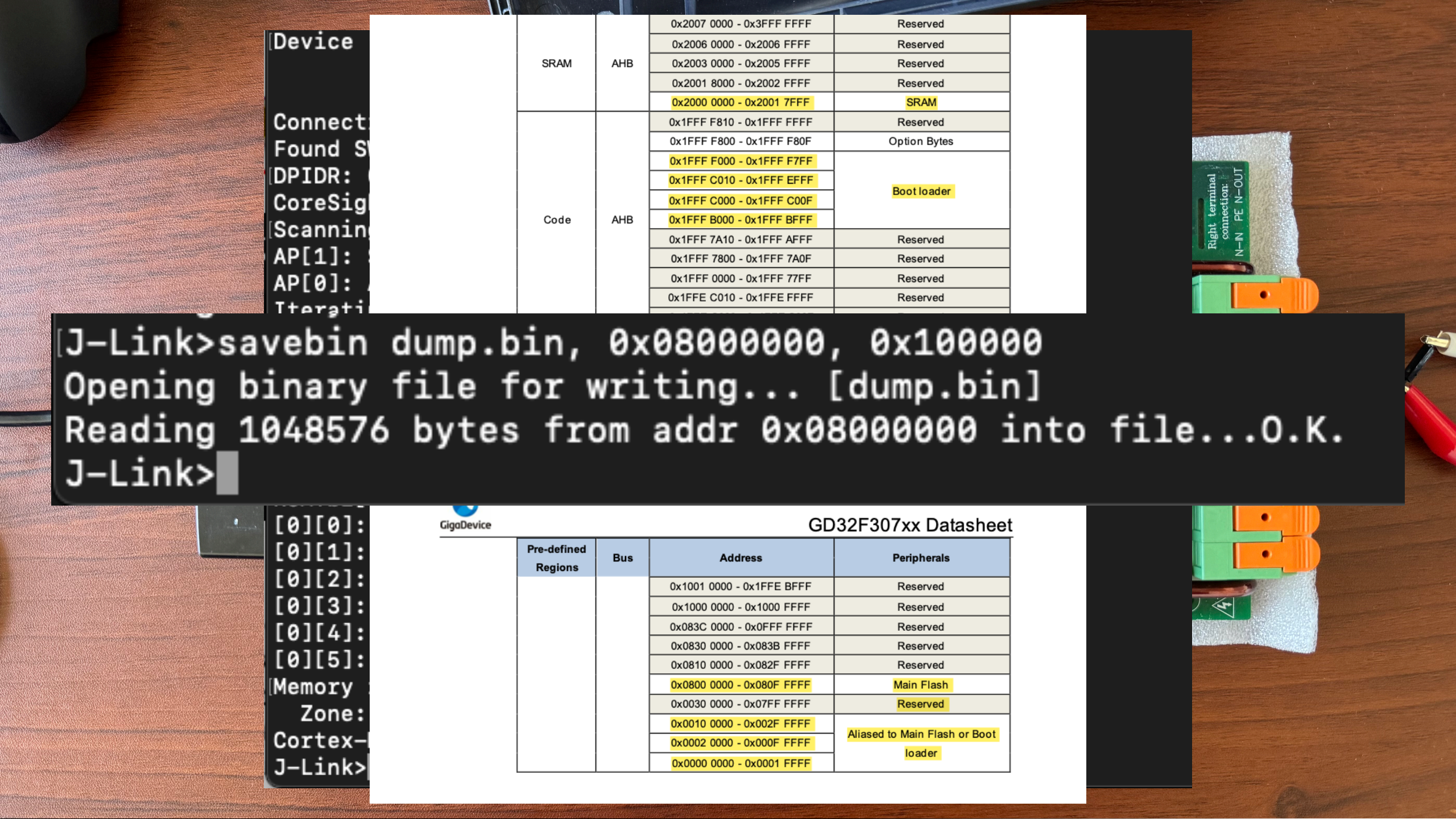

❯ JLinkExe
SEGGER J-Link Commander V8.10a (Compiled Oct 2 2024 14:14:28)
DLL version V8.10a, compiled Oct 2 2024 14:14:09
Connecting to J-Link via USB...O.K.
Firmware: J-Link Pro V5-1 compiled Dec 10 2024 10:54:56
Hardware version: V5.10
J-Link uptime (since boot): 0d 00h 00m 49s
S/N: 175104598
License(s): RDI, FlashBP, FlashDL, JFlash, GDB
USB speed mode: High speed (480 MBit/s)
IP-Addr: DHCP (no addr. received yet)
VTref=3.300V (fixed)
Type "connect" to establish a target connection, '?' for help
J-Link>
USB
LAN
Right terminal connection:
N-IN PE N-OUT


Device "CORTEX-M4" selected.
Connecting to target via SWD
Found SW-DP with ID 0x2BA01477
DPIDR: 0x2BA01477
CoreSight SoC-400 or earlier
Scanning AP map to find all available APs
AP[1]: Stopped AP scan as end of AP map has been reached
AP[0]: AHB-AP (IDR: 0x24770011, ADDR: 0x00000000)
Iterating through AP map to find AHB-AP to use
AP[0]: Core found
AP[0]: AHB-AP ROM base: 0xE00FF000
CPUID register: 0x410FC241. Implementer code: 0x41 (ARM)
Found Cortex-M4 r0p1, Little endian.
FPUnit: 6 code (BP) slots and 2 literal slots
CoreSight components:
ROMTbl[0] @ E00FF000
[0][0]: E000E000 CID B105E00D PID 000BB00C SCS-M7
[0][1]: E0001000 CID B105E00D PID 003BB002 DWT
[0][2]: E0002000 CID B105E00D PID 002BB003 FPB
[0][3]: E0000000 CID B105E00D PID 003BB001 ITM
[0][4]: E0040000 CID B105900D PID 000BB9A1 TPIU
[0][5]: E0041000 CID 00000000 PID 00000000 ???
Memory zones:
Zone: "Default" Description: Default access mode
Cortex-M4 identified.
J-Link>
USB
LAN
Right terminal connection:
N-IN PE N-OUT



SRAM	AHB	0x2007 0000 - 0x3FFF FFFF	Reserved
		0x2006 0000 - 0x2006 FFFF	Reserved
		0x2003 0000 - 0x2005 FFFF	Reserved
		0x2001 8000 - 0x2002 FFFF	Reserved
		0x2000 0000 - 0x2001 7FFF	SRAM
Code	AHB	0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80F	Option Bytes
		0x1FFF F000 - 0x1FFF F7FF	Boot loader
		0x1FFF C010 - 0x1FFF EFFF	
		0x1FFF C000 - 0x1FFF C00F	
		0x1FFF B000 - 0x1FFF BFFF	
		0x1FFF 7A10 - 0x1FFF AFFF	Reserved
		0x1FFF 7800 - 0x1FFF 7A0F	Reserved
		0x1FFF 0000 - 0x1FFF 77FF	Reserved
		0x1FFE C010 - 0x1FFE FFFF	Reserved
		0x1FFE C000 - 0x1FFE C00F	Reserved

Pre-defined Regions	Bus	Address	Peripherals
		0x1001 0000 - 0x1FFE BFFF	Reserved
		0x1000 0000 - 0x1000 FFFF	Reserved
		0x083C 0000 - 0x0FFF FFFF	Reserved
		0x0830 0000 - 0x083B FFFF	Reserved
		0x0810 0000 - 0x082F FFFF	Reserved
		0x0800 0000 - 0x080F FFFF	Main Flash
		0x0030 0000 - 0x07FF FFFF	Reserved
		0x0010 0000 - 0x002F FFFF	Aliased to Main Flash or Boot loader
		0x0002 0000 - 0x000F FFFF	
		0x0000 0000 - 0x0001 FFFF	



SRAM	AHB	0x2007 0000 - 0x3FFF FFFF	Reserved
		0x2006 0000 - 0x2006 FFFF	Reserved
		0x2003 0000 - 0x2005 FFFF	Reserved
		0x2001 8000 - 0x2002 FFFF	Reserved
		0x2000 0000 - 0x2001 7FFF	SRAM
Code	AHB	0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80F	Option Bytes
		0x1FFF F000 - 0x1FFF F7FF	Boot loader
		0x1FFF C010 - 0x1FFF EFFF	
		0x1FFF C000 - 0x1FFF C00F	
		0x1FFF B000 - 0x1FFF BFFF	
		0x1FFF 7A10 - 0x1FFF AFFF	Reserved
		0x1FFF 7800 - 0x1FFF 7A0F	Reserved
		0x1FFF 0000 - 0x1FFF 77FF	Reserved
		0x1FFE C010 - 0x1FFE FFFF	Reserved

```
[J-Link>savebin dump.bin, 0x08000000, 0x100000
Opening binary file for writing... [dump.bin]
Reading 1048576 bytes from addr 0x08000000 into file...O.K.
J-Link>
```

GigaDevice GD32F307xx Datasheet

Pre-defined Regions	Bus	Address	Peripherals
		0x1001 0000 - 0x1FFE BFFF	Reserved
		0x1000 0000 - 0x1000 FFFF	Reserved
		0x083C 0000 - 0x0FFF FFFF	Reserved
		0x0830 0000 - 0x083B FFFF	Reserved
		0x0810 0000 - 0x082F FFFF	Reserved
		0x0800 0000 - 0x080F FFFF	Main Flash
		0x0030 0000 - 0x07FF FFFF	Reserved
		0x0010 0000 - 0x002F FFFF	Aliased to Main Flash or Boot loader
		0x0002 0000 - 0x000F FFFF	
		0x0000 0000 - 0x0001 FFFF	

SO, YOU DON'T CARE AT ALL?

CodeBrowser: EyeQ:/Wolfbox_EVCharger_app_1.00_decrypted.bin

Tools Window Help

ox_EVCharger_app_1.00_decrypted.bin

```
000d3fd1 63              ??         63h    c
000d3fd2 3a              ??         3Ah    :
000d3fd3 25              ??         25h    %
000d3fd4 73              ??         73h    s
000d3fd5 00              ??         00h
000d3fd6 66 77 2d        ds         "fw->sw_ver:%s"
         3e 73 77
         5f 76 65 ...
000d3fe4 66 77 2d        ds         "fw->file_size:%d"
         3e 66 69
         6c 65 5f ...
000d3ff5 73 65 6e        ds         "send syn err:%d"
         64 20 73
         79 6e 20 ...
000d4005 75 70 67        ds         "upgrade_start_cb fault:%d"
         72 61 64
         65 5f 73 ...
000d401f 73 74 61        ds         "start tuya_iot_upgrade_dev"
         72 74 20
         74 75 79 ...
000d403a 74 75 79        ds         "tuya_iot_upgrade_dev err:%d"
         61 5f 69
         6f 74 5f ...
000d4056 70 72 65        ds         "pre_gw_ug_inform_cb"
         5f 67 77
         5f 75 67 ...
000d406a 6d              ??         6Dh    m
000d406b 63              ??         63h    c
000d406c 75              ??         75h    u
000d406d 2e              ??         2Eh    .
000d406e 76              ??         76h    v
000d406f 65              ??         65h    e
000d4070 72              ??         72h    r
000d4071 00              ??         00h
000d4072 67 65 74        ds         "get map seesion id fault:%d"
         20 6d 61
         70 20 73 ...
000d408e 67 65 74        ds         "get seesion id :%d"
         20 73 65
         65 73 69 ...
```

Defined Strings - 3929 items

Location	String Value	String Representation	Data Type
000d401f	start tuya_iot_upgrade_dev	"start tuya_iot_upgrade_d...	ds
000d403a	tuya_iot_upgrade_dev err:%d	"tuya_iot_upgrade_dev err...	ds
000d4056	pre_gw_ug_inform_cb	"pre_gw_ug_inform_cb"	ds
000d4072	get map seesion id fault:%d	"get map seesion id fault:...	ds
000d408e	get seesion id :%d	"get seesion id :%d"	ds
000d40a1	mul map ver err %d	"mul map ver err %d"	ds
000d40b4	mul map handle err %d	"mul map handle err %d"	ds
000d40ca	stream end err %d	"stream end err %d"	ds
000d40dc	data len err %d	"data len err %d"	ds
000d40ec	stream trans err %d	"stream trans err %d"	ds
000d4100	free mem %d is not enough	"free mem %d is not enou...	ds
000d411a	stream start err %d	"stream start err %d"	ds
000d412e	stream server task init err %d	"stream server task init err...	ds
000d4161	dp %d not found	"dp %d not found"	ds
000d4171	dp %d type invalid %d	"dp %d type invalid %d"	ds
000d4187	dp %d len invalid %d	"dp %d len invalid %d"	ds
000d419c	dp %d data invalid %d	"dp %d data invalid %d"	ds
000d41b2	dp report err %d	"dp report err %d"	ds
000d41ca	dev_cntl is NULL	"dev_cntl is NULL"	ds
000d41db	dp_cntl->dp_desc.mode is M_RO	"dp_cntl->dp_desc.mode ...	ds
000d41f9	dp type is err (not obj_data)	"dp type is err (not obj_dat...	ds
000d4217	dp_cntl is NULL	"dp_cntl is NULL"	ds
000d4227	state err:%d	"state err:%d"	ds
000d4234	close lowpower	"close lowpower"	ds
000d4243	open lowpower	"open lowpower"	ds
000d4251	sta is:%d----lp op err:%d	"sta is:%d----lp op err:%d"	ds
000d426b	nw stat err:%u	"nw stat err:%u"	ds
000d427a	ir stat:%d,wifi stat:%d,work high:%d	"ir stat:%d,wifi stat:%d,w...	ds

Filter:

Decompiler × Defined Strings ×

pting

CodeBrowser: EyeQ:/Wolfbox_EVCharger_app_1.00_decrypted.bin

Tools Window Help

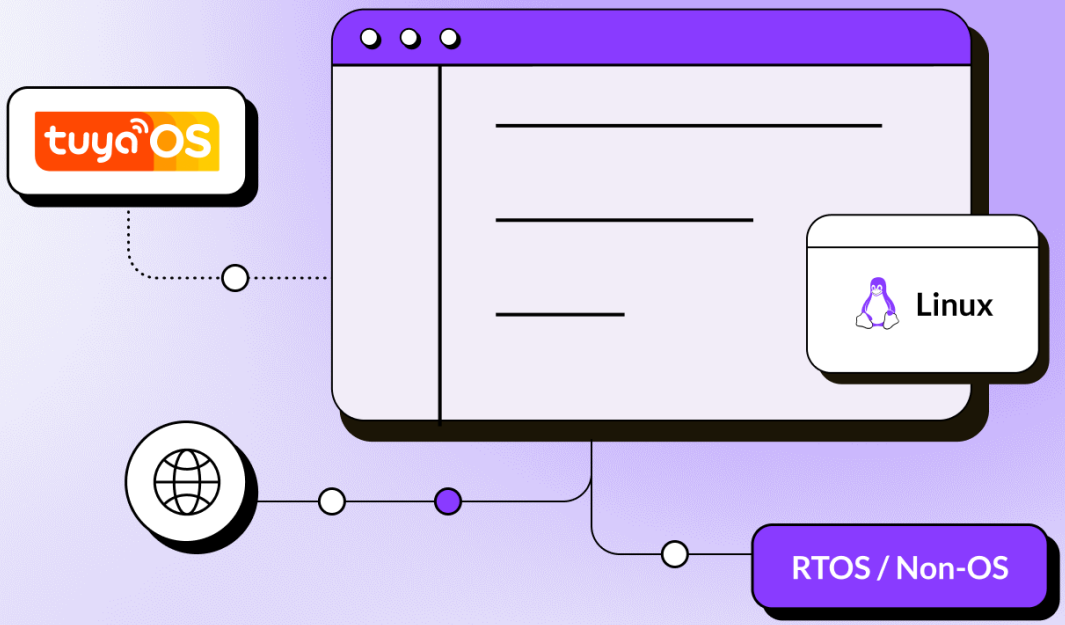
ox_EVCharger_app_1.00_decrypted.bin

```
000d3fd1 63          ??         63h    c
000d3fd2 3a          ??         3Ah    :
000d3fd3 25          ??         25h    %
```

TuyaOS

Built atop RTOS, Linux, and Non-OS, TuyaOS is a distributed and platform-agnostic IoT operating system.

Get Started >




```
000d4071 00          ??         00h
000d4072 67 65 74    ds         "get map seesion id fault:%d"
         20 6d 61
         70 20 73 ...
000d408e 67 65 74    ds         "get seesion id :%d"
         20 73 65
         65 73 69 ...
```

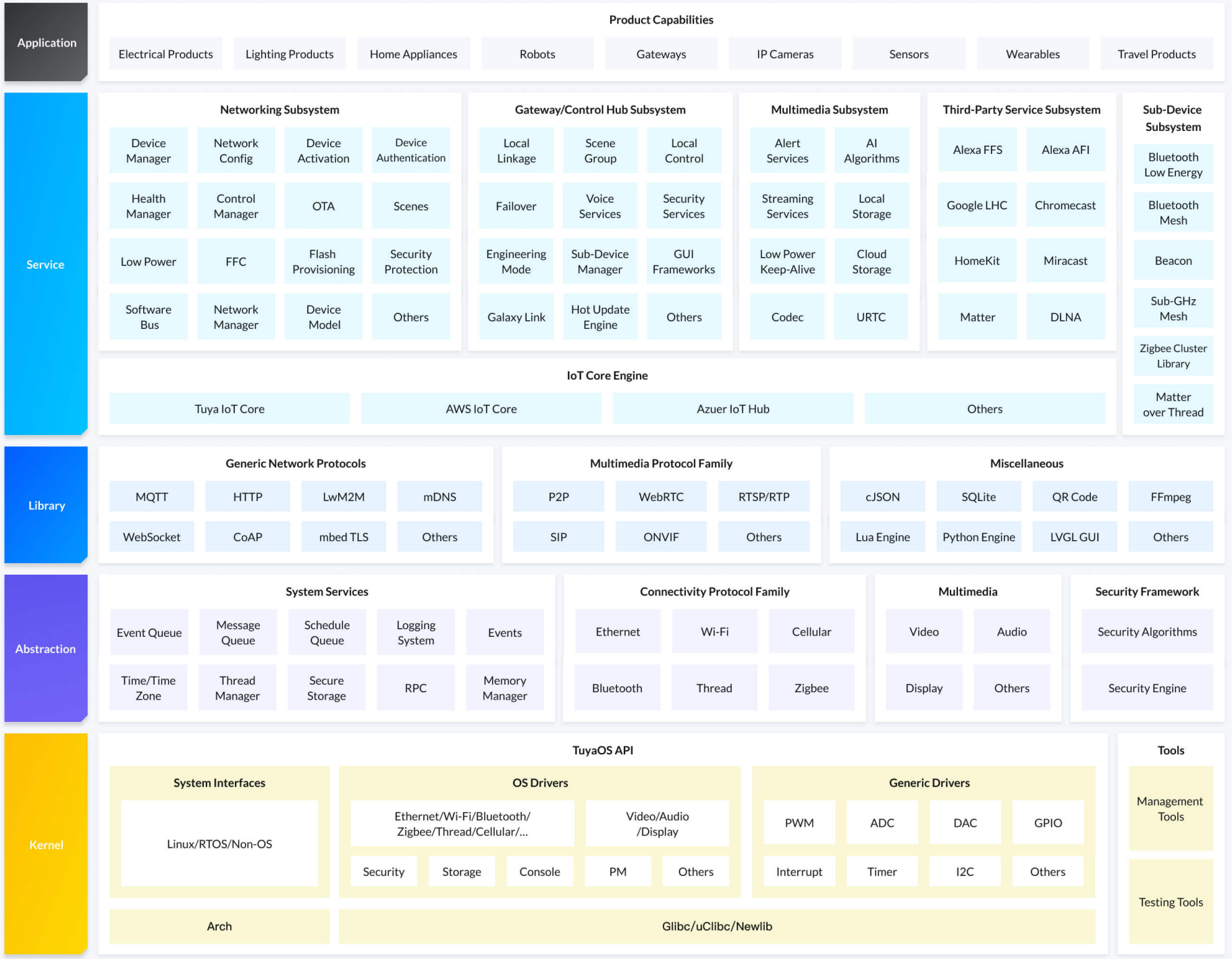
Defined Strings - 3929 items

Location	String Value	String Representation	Data Type
000d401f	start tuya_iot_upgrade_dev	"start tuya_iot_upgrade_d...	ds
000d4251	sta is:%d----lp op err:%d	"sta is:%d----lp op err:%d"	ds
000d426b	nw stat err:%u	"nw stat err:%u"	ds
000d427a	ir stat:%d.wifi stat:%d.work high:%d	"ir stat:%d.wifi stat:%d.w...	ds

Filter:

Decompiler × Defined Strings ×

pting


Application
Product Capabilities
Electrical Products
Lighting Products
Home Appliances
Robots
Gateways
IP Cameras
Sensors
Wearables
Travel Products
Service
Networking Subsystem
Device Manager
Network Config
Device Activation
Device Authentication
Health Manager
Control Manager
OTA
Scenes
Low Power
FFC
Flash Provisioning
Security Protection
Software Bus
Network Manager
Device Model
Others
Gateway/Control Hub Subsystem
Local Linkage
Scene Group
Local Control
Failover
Voice Services
Security Services
Engineering Mode
Sub-Device Manager
GUI Frameworks
Galaxy Link
Hot Update Engine
Others
Multimedia Subsystem
Alert Services
AI Algorithms
Streaming Services
Local Storage
Low Power Keep-Alive
Cloud Storage
Codec
URTC
Third-Party Service Subsystem
Alexa FFS
Alexa AFI
Google LHC
Chromecast
HomeKit
Miracast
Matter
DLNA
Sub-Device Subsystem
Bluetooth Low Energy
Bluetooth Mesh
Beacon
Sub-GHz Mesh
Zigbee Cluster Library
Matter over Thread
IoT Core Engine
Tuya IoT Core
AWS IoT Core
Azuer IoT Hub
Others
Library
Generic Network Protocols
MQTT
HTTP
LwM2M
mDNS
WebSocket
CoAP
mbed TLS
Others
Multimedia Protocol Family
P2P
WebRTC
RTSP/RTP
SIP
ONVIF
Others
Miscellaneous
cJSON
SQLite
QR Code
FFmpeg
Lua Engine
Python Engine
LVGL GUI
Others
Abstraction
System Services
Event Queue
Message Queue
Schedule Queue
Logging System
Events
Time/Time Zone
Thread Manager
Secure Storage
RPC
Memory Manager
Connectivity Protocol Family
Ethernet
Wi-Fi
Cellular
Bluetooth
Thread
Zigbee
Multimedia
Video
Audio
Display
Others
Security Framework
Security Algorithms
Security Engine
Kernel
TuyaOS API
System Interfaces
Linux/RTOS/Non-OS
OS Drivers
Ethernet/Wi-Fi/Bluetooth/ Zigbee/Thread/Cellular/...
Video/Audio /Display
Security
Storage
Console
PM
Others
Generic Drivers
PWM
ADC
DAC
GPIO
Interrupt
Timer
I2C
Others
Arch
Glibc/uClibc/Newlib
Tools
Management Tools
Testing Tools

Application

Product Capabilities

Arch

Glibc/uClibc/Newlib

BK7231T COB Hardware Design Guideline

Last Updated on : 2024-06-24 08:01:48

download

This topic describes hardware design guidelines on development based on the BK7231T Wi-Fi chip.

Application scope

BK7231T is a highly integrated wireless and radio frequency (RF) chip, with a built-in Wi-Fi protocol stack and various library functions. It features a low-power 32-bit CPU, 1T1R WLAN, 256 kbit static RAM, 2 MB flash, and various peripherals. As a real-time operating system (RTOS), BT7231T integrates all Wi-Fi MAC and TCP/IP libraries. You can develop your own embedded Wi-Fi products.

Features

- Built-in low-power 32-bit CPU, 256 kbit static RAM, and 2 MB flash.
- The maximum clock speed is 120 MHz.
- Full functionality of IEEE 802.11b/g/n Wi-Fi connectivity.
- Integrated Bluetooth Low Energy (LE) system.
- The maximum output power is +17 dBm for 802.11b transmission.
- Various peripheral interfaces include PWM, I2C, UART, SPI, SDIO, USB, and IrDA.
- Operating voltage: 3.0V–3.6V
- Operating temperature: -40°C–105°C

Minimal system diagram

Tuya Developer Platform > Product Development > Hardware Reference Design > Radio Frequency Design > Wi-Fi

README MIT license

Tuya Cloudcutter

This repository contains the toolchain to exploit a wireless vulnerability that can jailbreak some of the latest smart devices built with the bk7231 chipset under various brand names by Tuya. The vulnerability as well as the exploitation tooling were identified and created by Khaled Nassar and Tom Clement with support from Jilles Groenendijk.

Our tool disconnects Tuya devices from the cloud, allowing them to run completely locally. Additionally, it can be used to flash custom firmware to devices over-the-air.

ℹ️ Do you like this tool? Please consider giving it a star on Github so it reaches more people. ✨

Minimal system diagram

README MIT license

Tuya Cloudcutter

```
        - not found!
[+] Searching for known exploit patterns
==========================================================================================
[!] The binary supplied appears to be patched and no longer vulnerable to the tuya-cloudcutter exploit.
==========================================================================================
[!] Storage file not found
[+] Device class: bk7231n_common_user_config_ty
[+] Version: 3.1.5
```

Our tool disconnects Tuya devices from the cloud, allowing them to run completely locally. Additionally, it can be used to flash custom firmware to devices over-the-air.

ℹ️ Do you like this tool? Please consider giving it a star on Github so it reaches more people. ✨

Minimal system diagram

README MIT license

Tuya Cloudcutter

```
        - not found!
[+] Searching for known exploit patterns
================================================================================
[!] The binary supplied appears to be patched and no longer vulnerable to the tuya-cloudcutter exploit.
================================================================================
[!] Storage file not found
[+] Device class: bk7231n_common_user_config_ty
[+] Version: 3.1.5
```

Our tool disconnects Tuya devices from the cloud, allowing them to run completely locally. Additionally, it can be used to flash custom firmware to devices over-the-air.

ℹ️ Do you like this tool? Please consider giving it a star on Github so it reaches more people. ✨

Minimal system diagram

README

Tuya

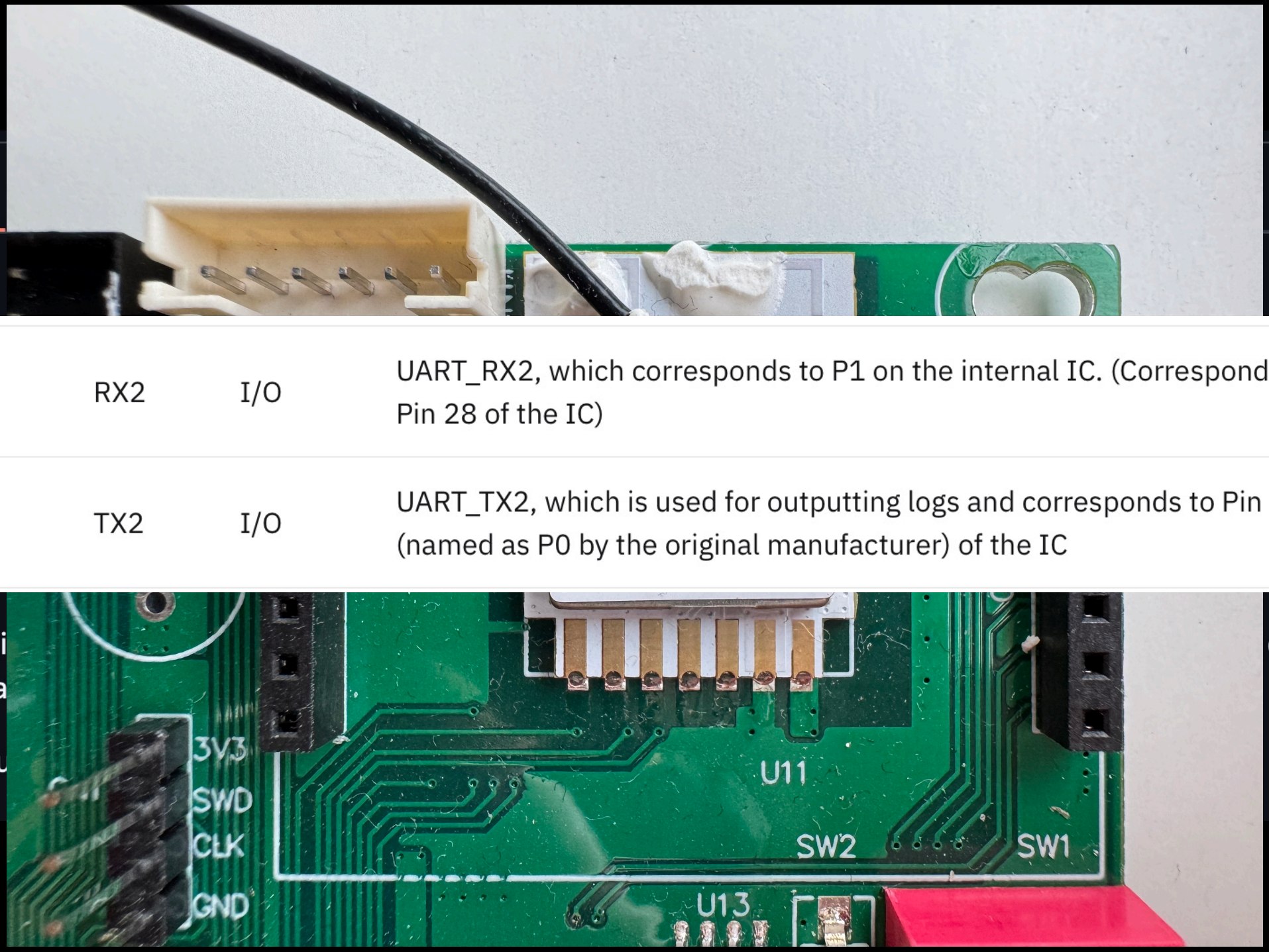
```
[+] Searchin
[!] The bina
[!] Storage
[+] Device c
[+] Version:
```

Our tool di
used to fla

Do you

can be

README



6	RX2	I/O	UART_RX2, which corresponds to P1 on the internal IC. (Correspond to Pin 28 of the IC)
7	TX2	I/O	UART_TX2, which is used for outputting logs and corresponds to Pin 29 (named as P0 by the original manufacturer) of the IC

Our tool di can be used to fla

Do you

AND APPARENTLY, ALL THESE WERE KNOWN

The Disclosure

- Disclosed vulnerabilities to manufacturer in November
  - Would not affect our research endeavor
- Vulnerability was “known” by the manufacturer
- Nothing was fixed, go on, buy that thing and juice it up.

DETAILING THE ATTACK SURFACES OF THE WOLFBOX E40 EV CHARGER

December 03, 2024 | Dmitry Janushkevich

The WolfBox E40 is a Level 2 electric vehicle charge station designed for residential home use. Its hardware has a minimal user interface, providing a Bluetooth Low Energy (BLE) interface for configuration and an NFC reader for user authentication. Typical for this class of devices, the appliance employs a mobile application for the owner's installation and regular operation of the equipment.

At the moment of writing, the software versions were reported as follows:

- Main module: 3.1.17
- MCU module: 1.2.6
- Mobile application: 1.0.3

WOLFBOX EV MOBILE APPLICATION

The manufacturer distributes one application to configure and maintain the device. The application, named WolfBox EV, is available for both Android and iOS users.

From pwn2own@trendmicro.com <pwn2own@trendmicro.com>

Reply | Forward | Archive | Junk | Delete | More

To Thomas Sermpinis <thomas.sermpinis@auxilium-labs.com>

15.01.2025, 11:37

Subject **Re: Pwn2Own Auto 2025 Application: WOLFBOX Level 2 EV Charger Target**

Hello,

Thanks for reaching out. Please excuse our lateness, as this email was in the junk folder.

The Aftermath

- While UAC is implemented, no protection to tampering is implemented
- Malicious actors can potentially tamper the device in minutes
  - Gaining persistence and remote access to the target
  - Most times chargers are exposed in public places, that's why they implement UAC
- Additionally, exposure of connected devices, like vehicles over V2G

BUT. WHY?

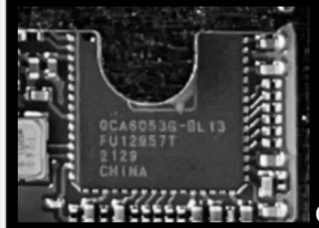
[illegible] The Conflict

Excuses VS Reality


TA2013/1167
APPROVED
0682
ACTIA NORDIC
MADE IN TUNISIA
E5 10 R-04182


dnt
QUALCOMM®
OK
M
JBC
OK
multicomp PRO

High-Speed CAN Transceiver with Standby and Sleep Mode
https://www.nxp.com/docs/en/data-sheet/TJA1043.pdf

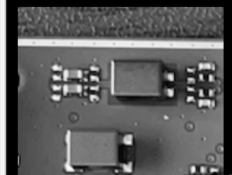


Qualcomm® APQ8053 SoCs for IoT
https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/apq8053-socs-product-brief_87-pg760-1-b.pdf

---

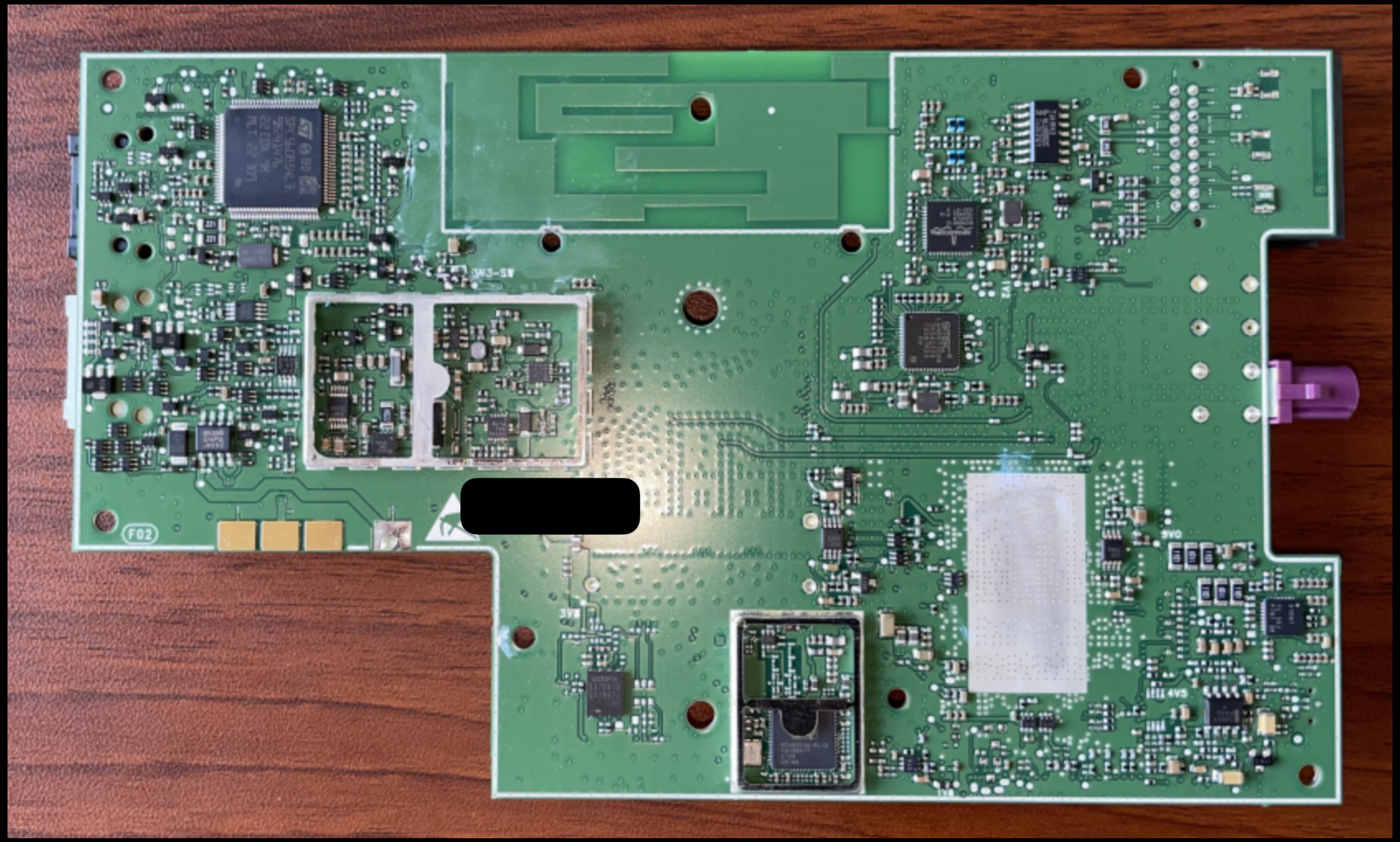
LTE/4G Modem Chip

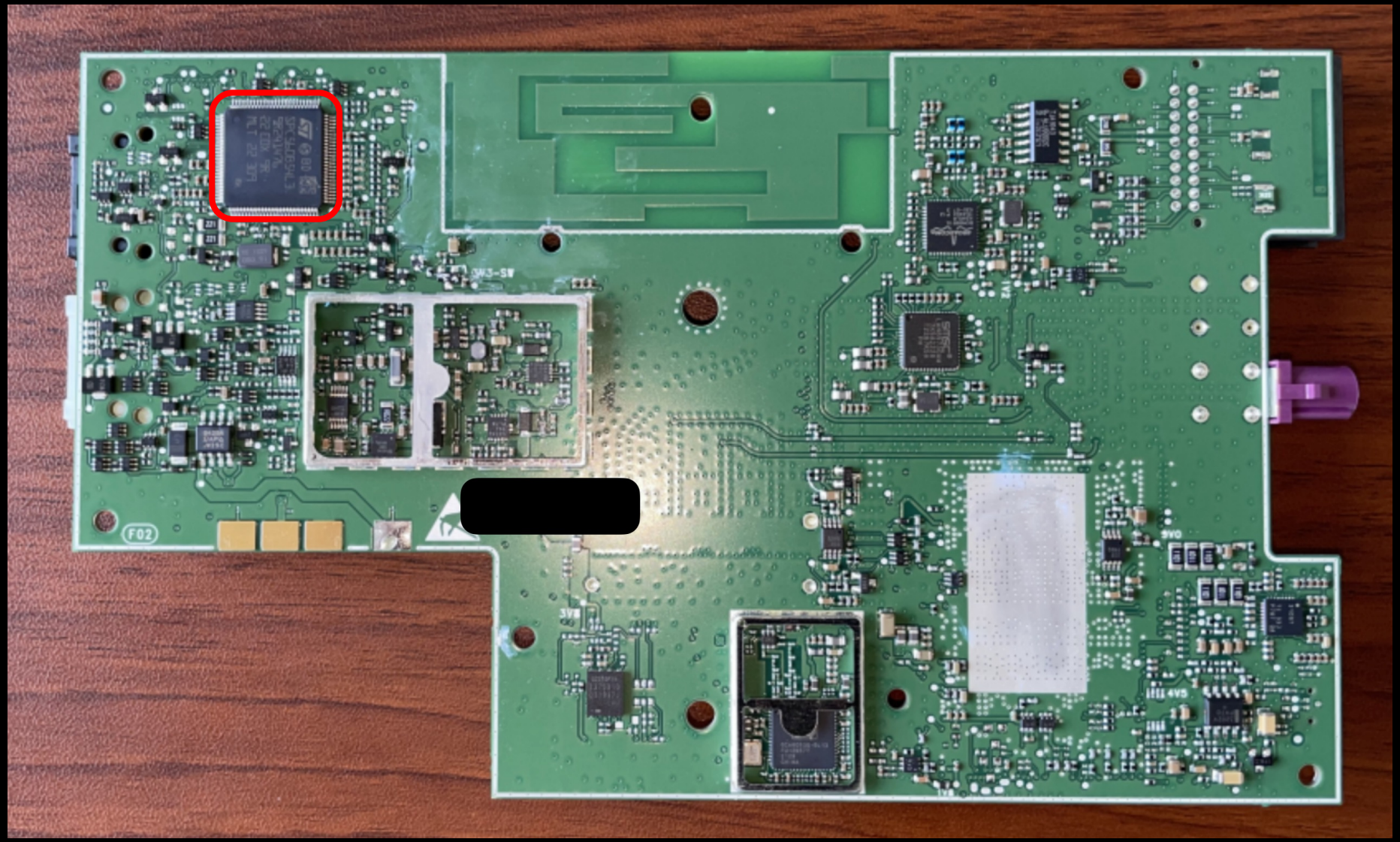
CMOS QUAD 2-INPUT NAND SCHMITT TRIGGER
https://www.ti.com/lit/ds/symlink/cd4093b-q1.pdf?ts=1694496622695

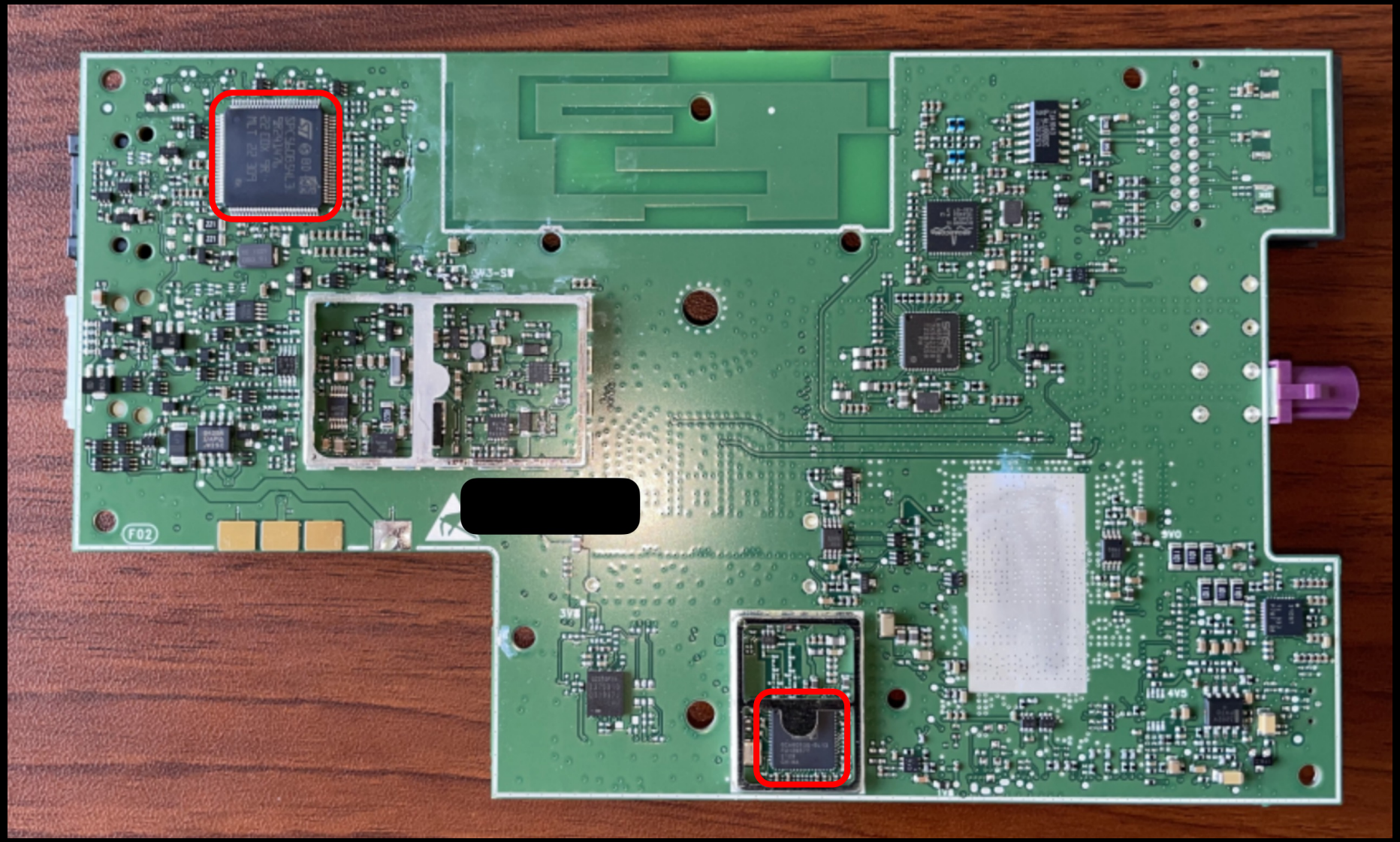

Transformers

Automotive Specific Architecture

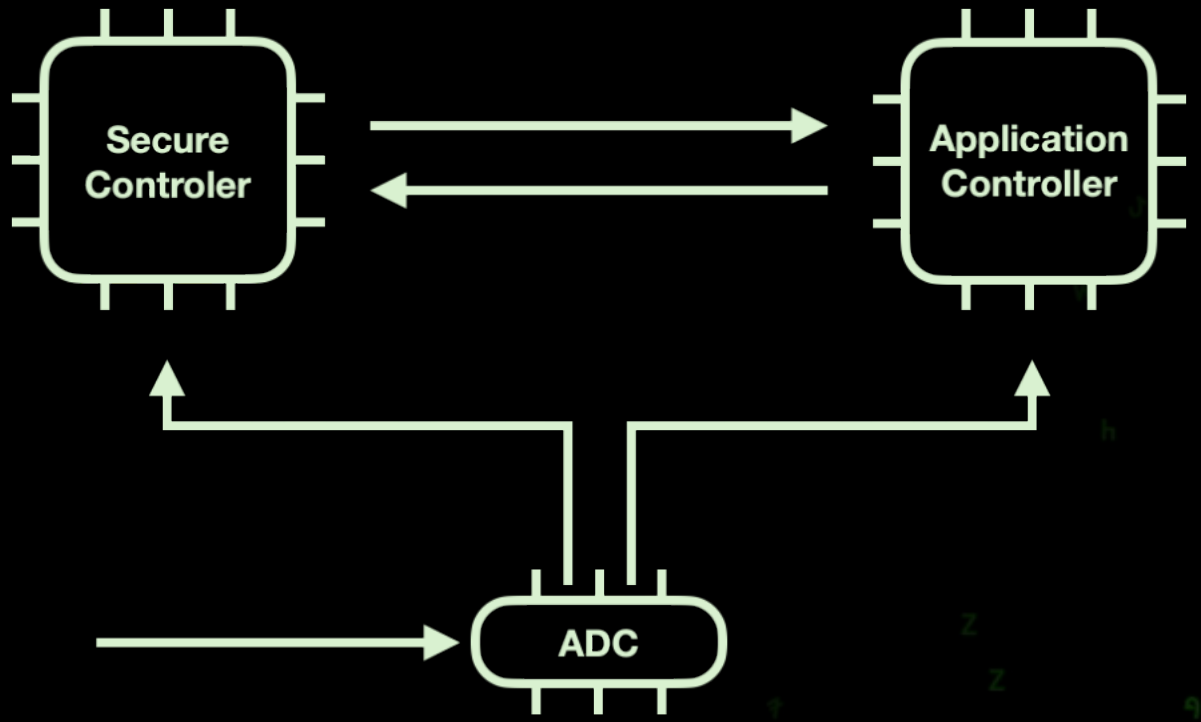
- Most Electronic Control Units (ECUs) use custom automotive specific architectures
  - e.g. Tricore, Renesas, etc.
  - Safety related functionality, with "real-time" responsiveness is required in automotive systems


SW3-SW
F02
3V3
5VO
4VS


SW3-SW
F02
3V3
5VO
4VS


F02
3V3
5VO
4VS

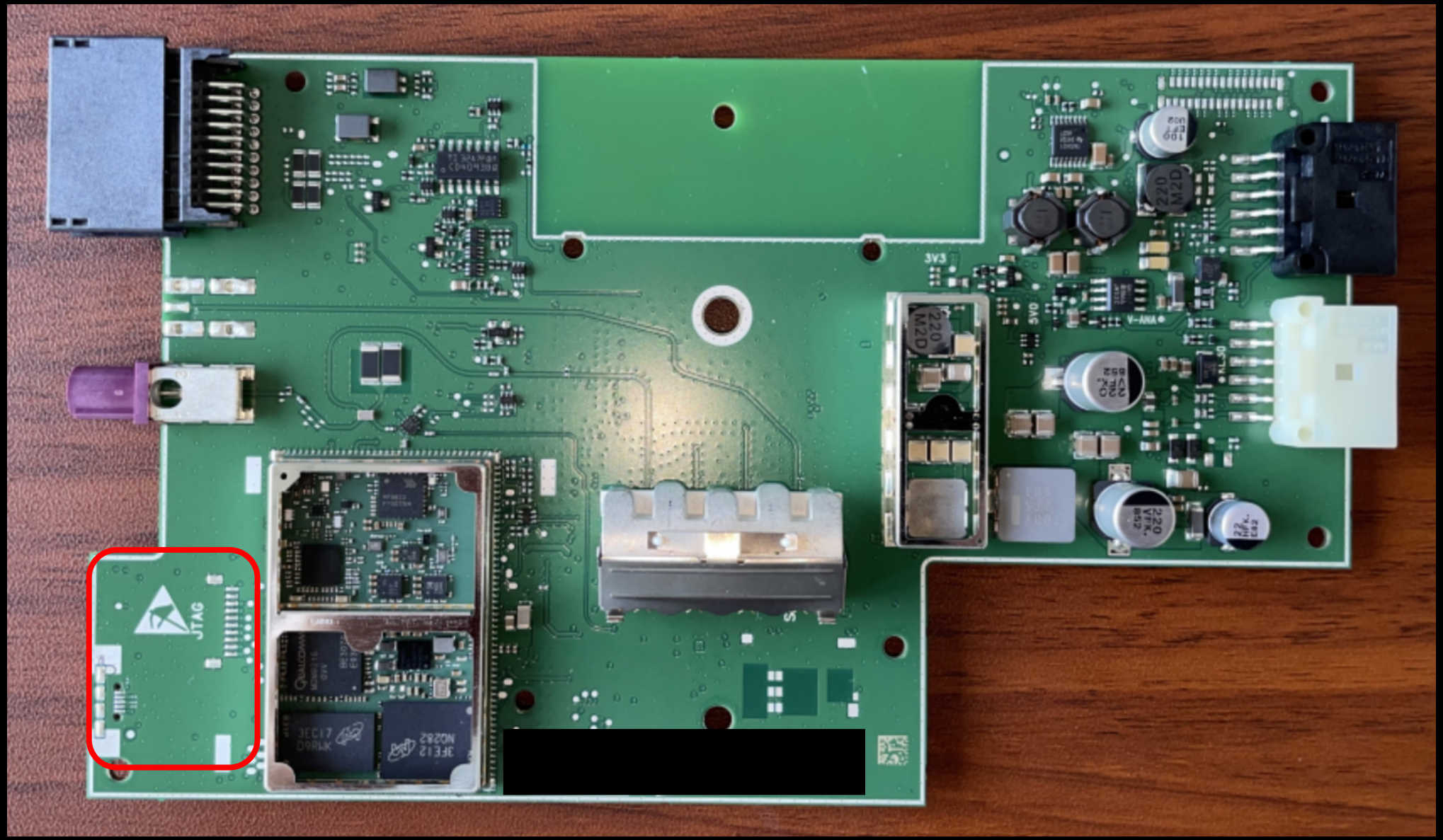
The Two SoCs

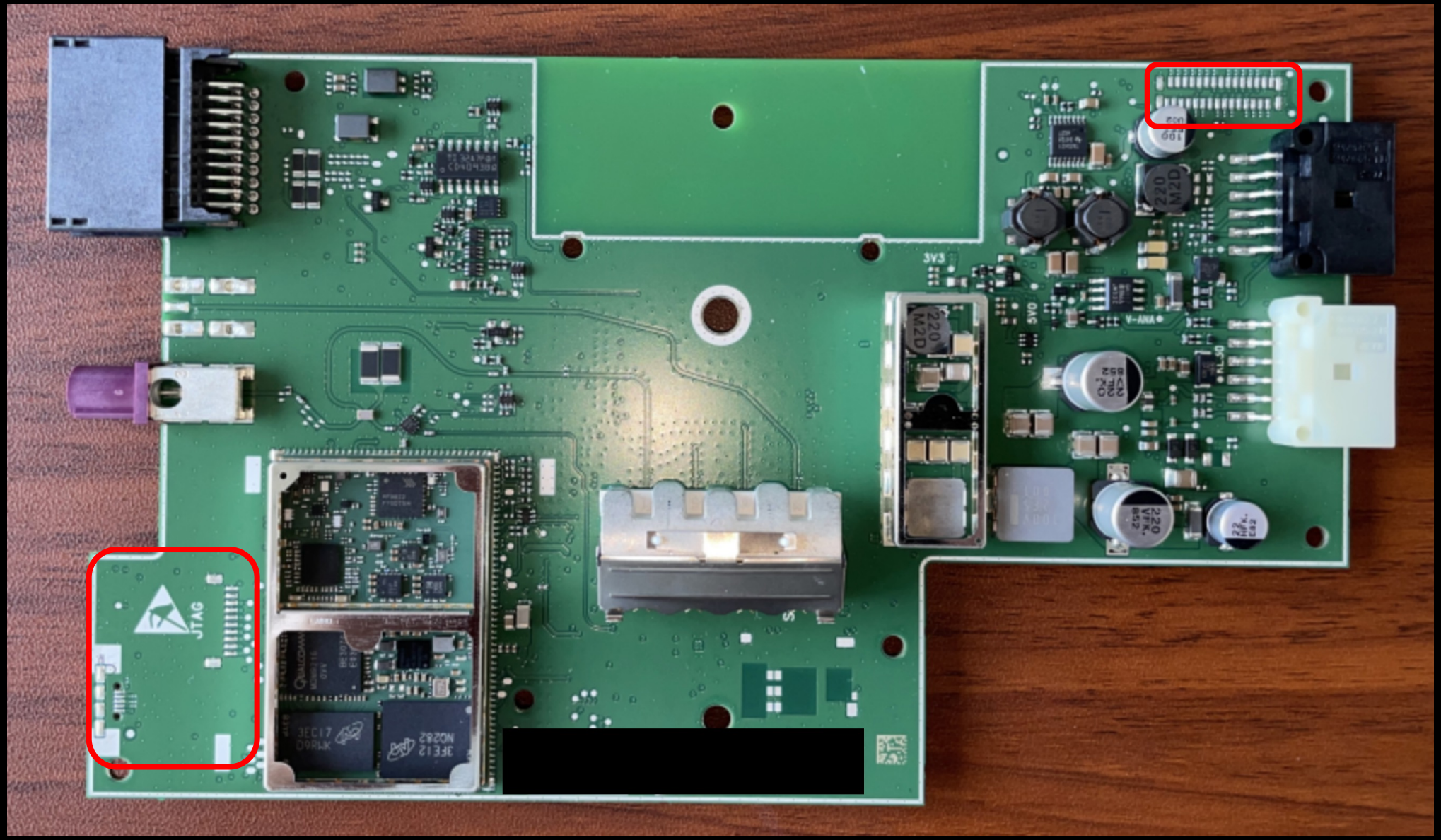


"Safety"-Critical Systems

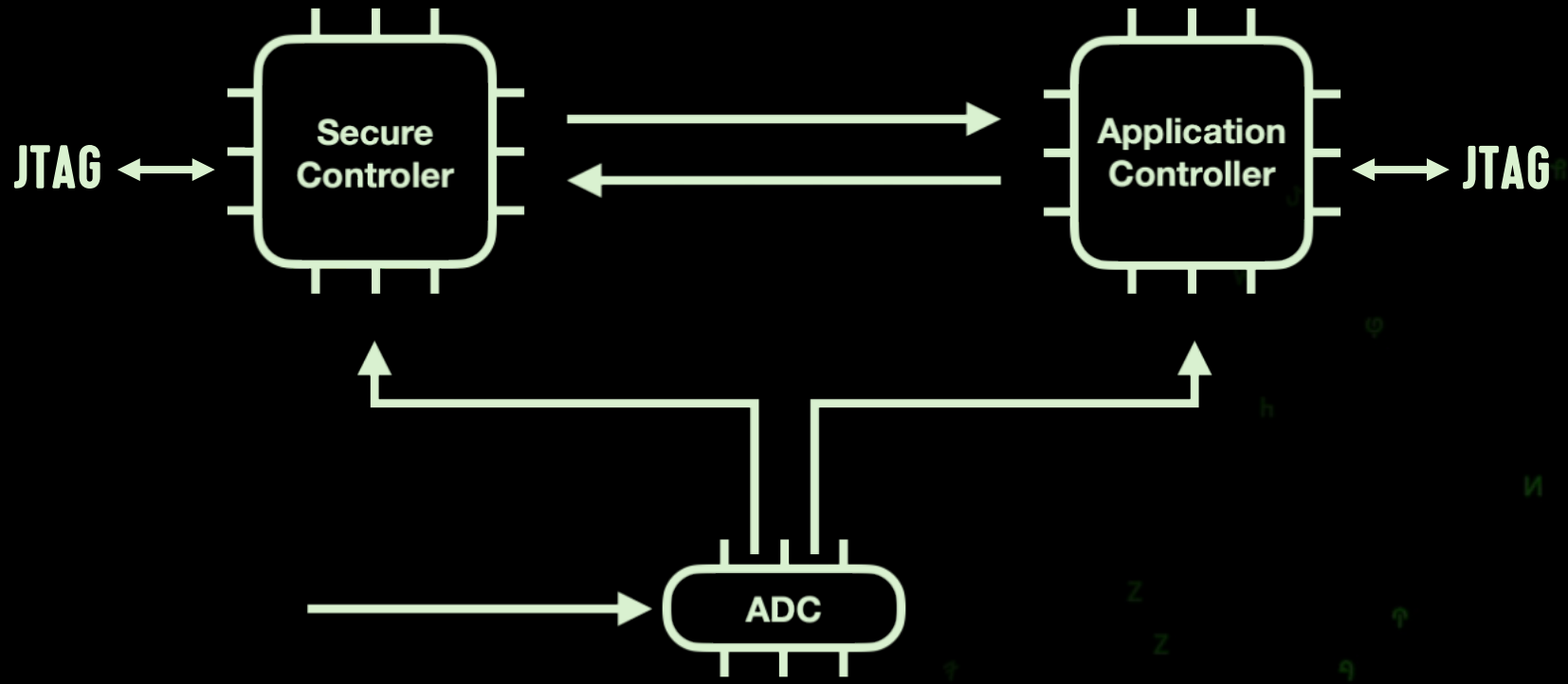
- Many ECUs are considered "Safety-critical systems"
  - Their failure may result in loss or damage to equipment, death or injury, and environmental harm
    - NOT only applicable to the Automotive Industry
  - In automotive, it mostly applies to ECUs which supply critical functionality or can directly affect safety
    - e.g. Airbag, Braking, Power Steering, and others

JTAG
3V3
[redacted]

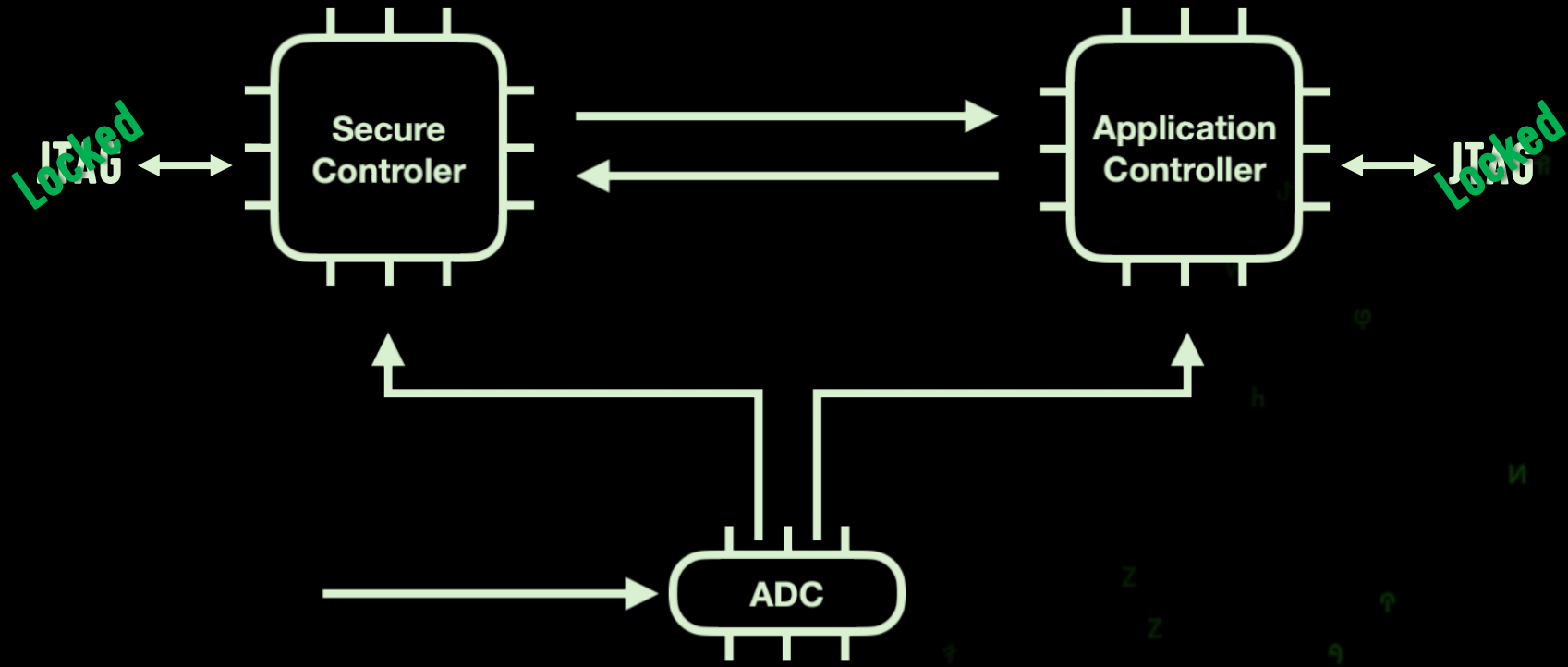

JTAG
3V3

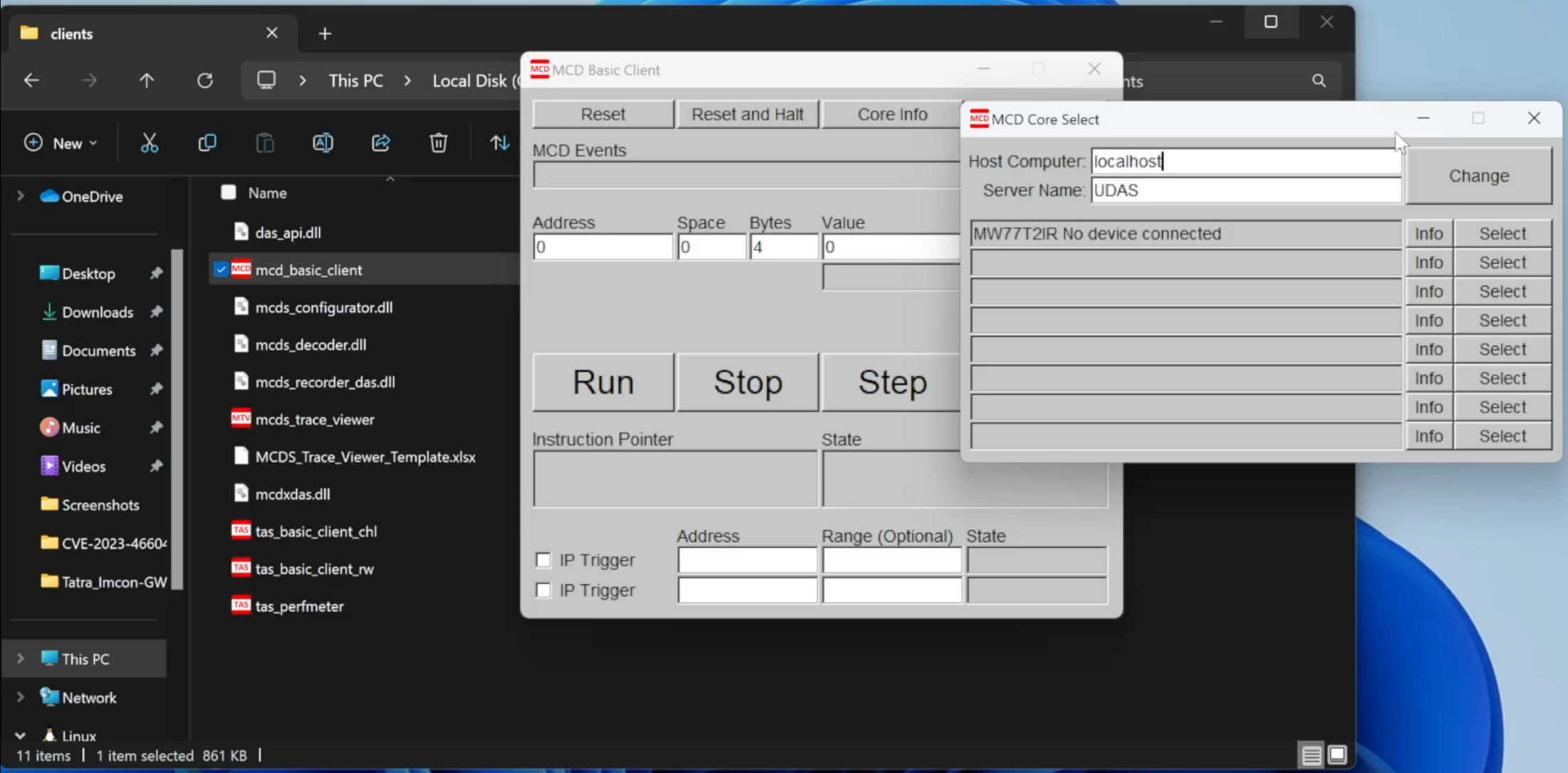

JTAG
3V3

The Two SoCs

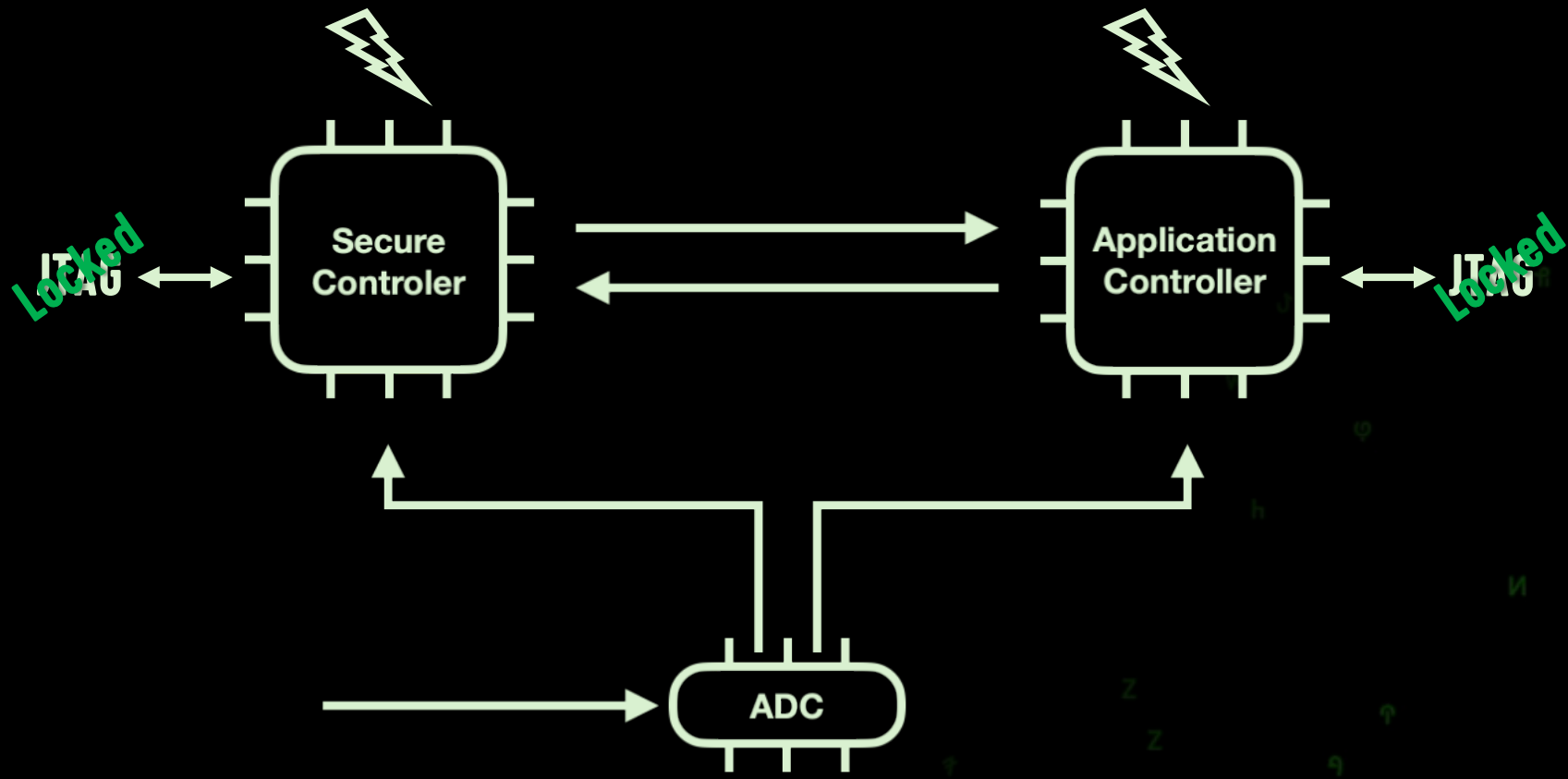


The Two SoCs

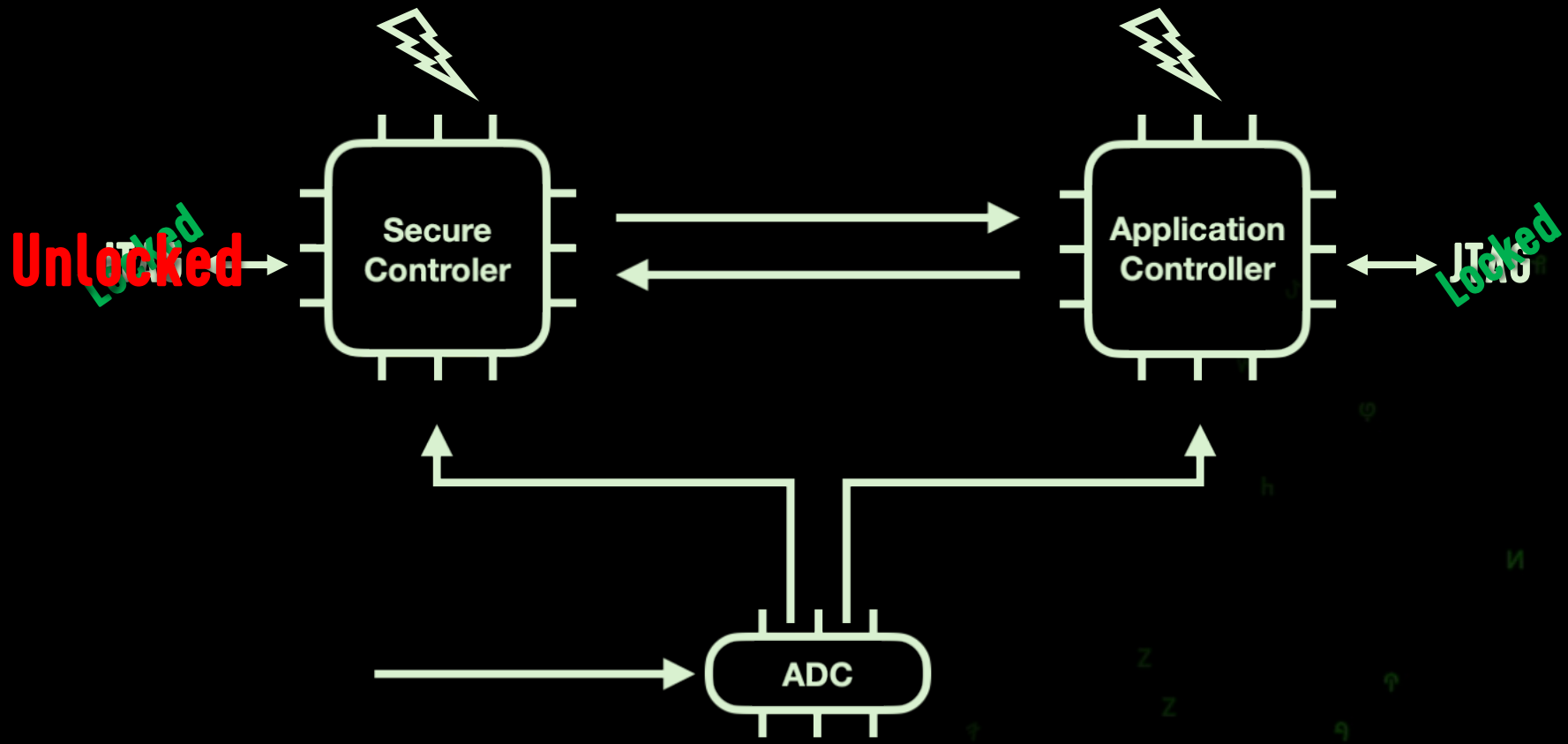

Locked
Secure
Controler
Application
Controller
Locked
ADC


clients
This PC
Local Disk
New
OneDrive
Desktop
Downloads
Documents
Pictures
Music
Videos
Screenshots
CVE-2023-4660
Tatra_Imcon-GW
This PC
Network
Linux
Name
das_api.dll
mcd_basic_client
mcds_configurator.dll
mcds_decoder.dll
mcds_recorder_das.dll
mcds_trace_viewer
MCDS_Trace_Viewer_Template.xlsx
mcdxdas.dll
tas_basic_client_chl
tas_basic_client_rw
tas_perfmeter
11 items | 1 item selected 861 KB
MCD Basic Client
Reset
Reset and Halt
Core Info
MCD Events
Address
Space
Bytes
Value
0
0
4
0
Run
Stop
Step
Instruction Pointer
State
Address
Range (Optional)
State
IP Trigger
IP Trigger
MCD Core Select
Host Computer:
localhost
Server Name:
UDAS
Change
MW77T2IR No device connected
Info
Select

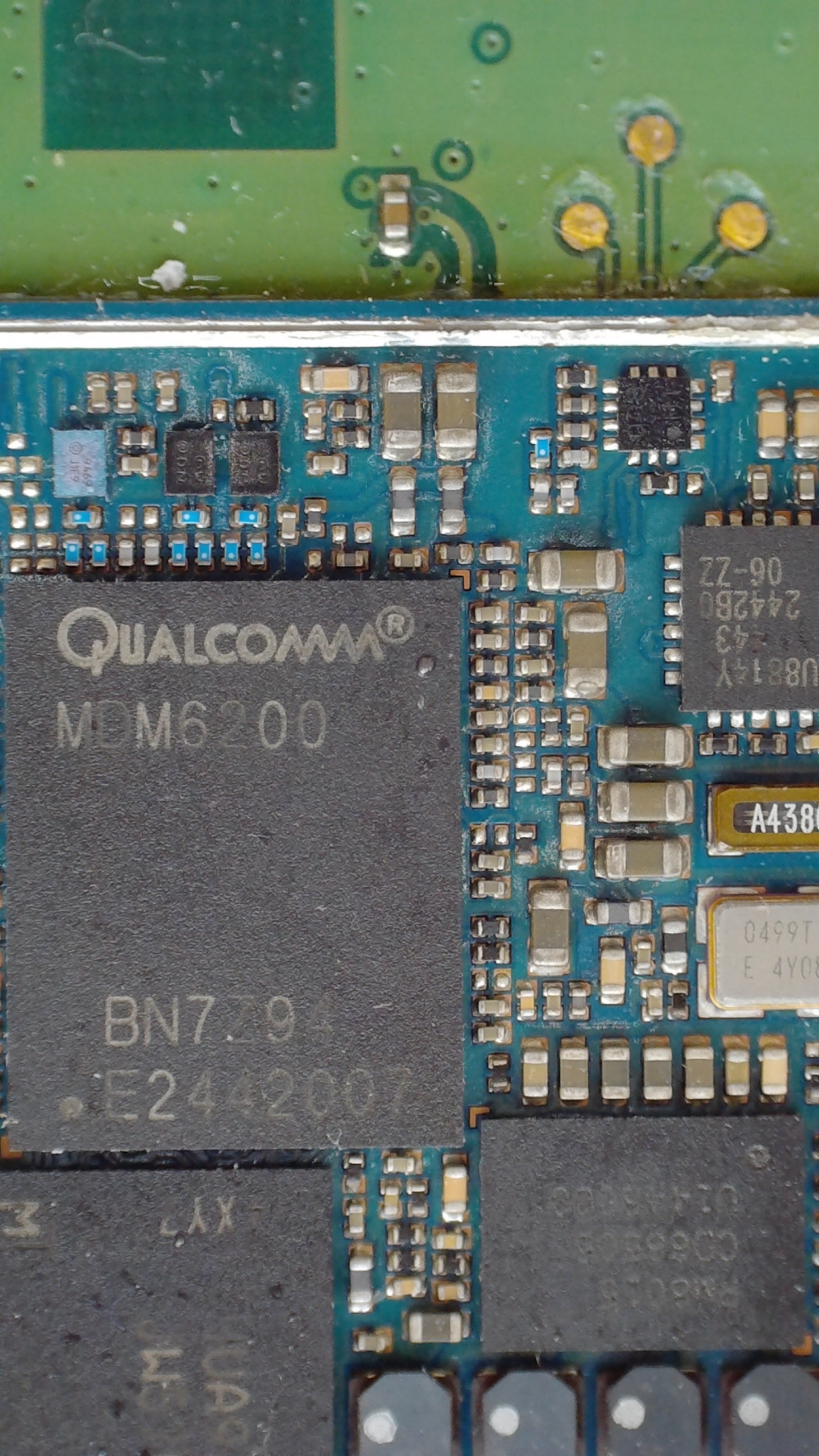
The Two SoCs


Locked
Secure
Controler
Application
Controller
Locked
ADC

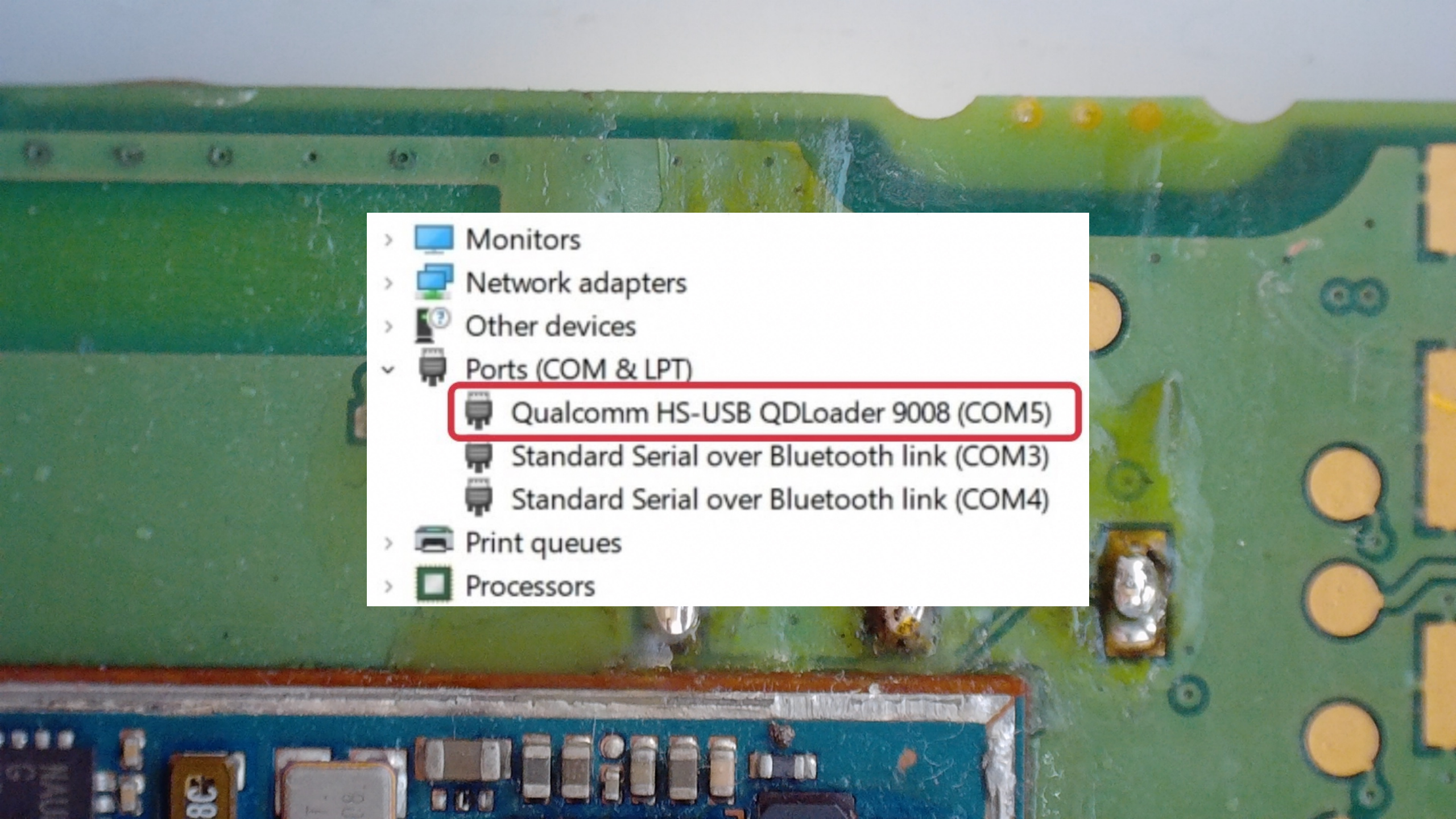
The Two SoCs


Unlocked
Locked
Secure Controler
Application Controller
JTAG
Locked
ADC


SECURITY LEVEL?
MAXIMUM!


QUALCOMM®
MDM6200

A438G
0469T
E 4Y08


Monitors
Network adapters
Other devices
Ports (COM & LPT)
Qualcomm HS-USB QDLoader 9008 (COM5)
Standard Serial over Bluetooth link (COM3)
Standard Serial over Bluetooth link (COM4)
Print queues
Processors

Decompile: FUN_000192f6 – (.bin)

```
1
2  void FUN_000192f6(void)
3
4  {
5    undefined4 uVar1;
6    ushort uVar2;
7    undefined2 uVar3;
8    char cVar4;
9    char cVar5;
10   byte bVar6;
11   int iVar7;
12   int iVar8;
13   uint uVar9;
14
15   cVar4 = DAT_d000df5f;
16   bVar6 = DAT_d000dfd8;
17   cVar5 = DAT_d000df60;
18   if ((byte)(cVar4 - 1U) < 3) {
19     iVar7 = STM_TIM0;
20     iVar8 = STM_CAP;
21     uVar9 = STM_CLC;
22     uVar9 = uVar9 >> 8 & 7;
23     uVar2 = DAT_d000df58;
24     DAT_d000df50 = 0;
25     uVar1 = Ramd0000fcc;
26     DAT_d000df5a = ~uVar2;
27     DAT_d000df54 = uVar1;
28     DAT_d000df40 = CONCAT44(iVar8 * uVar9,iVar7 * uVar9);
29   }
30   uVar3 = DAT_d000df58;
31   DAT_d000df60 = cVar5 + '\x01';
32   DAT_d000dfd8 = bVar6 + 1 & 7;
33   (&DAT_d000dfc8)[bVar6] = uVar3;
34   return;
35 }
```


thomas.sermpinis — pi@raspberrypizero2w: ~/Tools/caringcaribounext — ssh pi@10.42.0.1 — 223×62
~ — pi@raspberrypizero2w: ~/Tools/caringcaribounext — ssh pi@10.42.0.1
pi@raspberrypizero2w:~/Tools/caringcaribounext $ ccn.py -i socketcan -b 500000 -c can0 uds_fuzz def
con_poc -r 0 -g 7 -il 1 -d 1.9 10032703 d3adb3 0x754 0x75c
pi@raspberrypizero2w:~/Tools/caringcaribounext $ candump can0
[1] 0:bash*
"raspberrypizero2w" 23:12 24-Mar-24

Now what?

- Despite the help from the contact point, OEM decided that cost of fixing the issue exceeds the expectations for these models
- **The argument:** *The new E/E architecture will fix those issues in future models – MY25/MY26*
  - Meaning that the actual architecture which is currently sold, is still vulnerable, and will be
  - OEM is not willing to fix or disclose the findings
- Due to the safety aspect of the vulnerabilities, I decided to NOT disclose them publicly and search other avenues of escalation

DEFCON

So what?

- 24 vulnerabilities were responsibly disclosed
  - 4 of them approved and CVE assigned
  - 20 of them were neglected, manufacturer did not answer
  - ALL of them are a result of hardware analysis and firmware extraction

IT'S BY DESIGN


SECURITY THROUGH OBSCURITY
imgflip.com

[redacted] The Solution?

Towards a secure future, with reasonable solutions

The Challenge

- Debug interfaces can be locked with adequate security protections
  - Software protections from passwords to securely generated and stored certificates
  - Hardware protections like removing headers and traces, implement fuses, chip tamperproof protections, etc.

- Though, it's not sunshine and flowers
- Side channel attacks and reverse engineering can lead to potential compromise
  - Main target is how expensive a company makes it for an attacker to achieve their goal

WE NEED TO HAVE SECURE ALTERNATIVES

The Reality

- Automotive components implement several modern and non-modern interfaces:
  - CAN(-FD)
  - Automotive Ethernet
  - FlexRay
- Not only secure implementation in low levels can be achieved, but higher level solutions are available

The Easy Solution

- CAN and UDS
  - CAN FD allows for enhanced bandwidth, encryption and authentication
  - Security features like AES, MAC, SecOC can be implemented
  - Application layer protocols like UDS and XCP can be used in a secure way to allow access to specific regions of memory after secure authentication mechanisms
    - E.g. certificate authentication service 0x29 on UDS, with encrypted data under data identifiers or memory locations

```
└─$ python3 cc.py -i can0 uds dump_dids 0x7df 0x7e8

--------------------
CARING CARIBOU v0.3
--------------------

Loaded module 'uds'

Dumping DIDs in range 0x0000-0xffff

Identified DIDs:
DID    Value (hex)
0x0100 4c696e757820454355203532e31302e36332d76382b20233134353920534d5020505245454d50542057656420f6374203620313630a34323a343920425354203230323120616172636836342047 4e552f4c696e7578
0x010f 2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d
0x0121 2f7573722f76656e646f722f616461732f696e70757470726f636573736f72
0x0125 2e2f616461732d6f75742e6c6f67
0x0129 41444153204c495354454e4552203078323232
0x012d 7569643d3130303128626f7429206769643d3130303128626f74292067726f7570733d3130303128626f7429
0x012f 2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d
0x0141 2f686f6d652f626f742f7564732d7365727665722d32
0x0145 2e2f7365656400
0x0149 554453204c495354454e4552203078376466
0x014d 7569643d3028726f6f7429206769643d3028726f6f74292067726f7570733d3028726f6f7429
0x014f 2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d
0x0161 535755706461746202d2050726f6772616d6d696e672073657373696f6e
0x0165 687474703a2f2f6c6f63616c686f73743a38303830
0x0169 536f667477617265205570646174652066f72204c6576656c203520414441532053595354454d
```

The Obvious Solution

- Automotive Ethernet
  - All the advantages of conventional ethernet on a single pair cable
  - High bandwidth which supports robust security protocols, like MACsec and TLS/DTLS
  - Application level authentication can be easily achieved, bot with automotive specific protocols (UDS over DoIP) or conventional ones

- Perfect solution, with no excuses for ECUs that implement Automotive Ethernet

The Next Challenge

- Trust and integrity have to be maintained

- **HSMs** can provide enhanced security
  - Dedicated hardware for secure key management, cryptographic operations, and secure storage

- **Secure Boot** can provide an established and trusted foundation
  - Ensures only authorized software runs on the ECU

The Next Challenge

- Authentication and Authorization
  - Robust authentication protocols (mutual authentication, cryptographic keys, challenge-response, PUFs)
  - Lightweight protocols for resource-constrained ECUs
  - Integration with HSMs for secure key storage
  - Effective authorization mechanisms (RBAC, ABAC) to prevent misuse
  - Secure session management and auditing of access attempts

The Elephant in the Room

- Remote Forensic Data Retention and retrieval
  - Anonymized data are already shared with the secure backend connection of modern vehicles
  - With real time data sharing, and utilization of the Ecall ECUs for data sharing after the crash, data can be obtained even if ECUs are severely damaged

- Huge privacy concern, especially if not implemented properly

The Poor Mans Idea

- Dedicated Physical Interface
  - Specialized protected port
  - Obscured and difficult to access
  - Completely isolated from other debug functionality
  - Solely for access to the specific memory locations hosting the forensics data

- Increased hardware complexity

11111111 The Final Countdown

Moving Towards Secure Automotive Forensics

DO THEY EVEN CARE?

THE END