SD Card Hacking

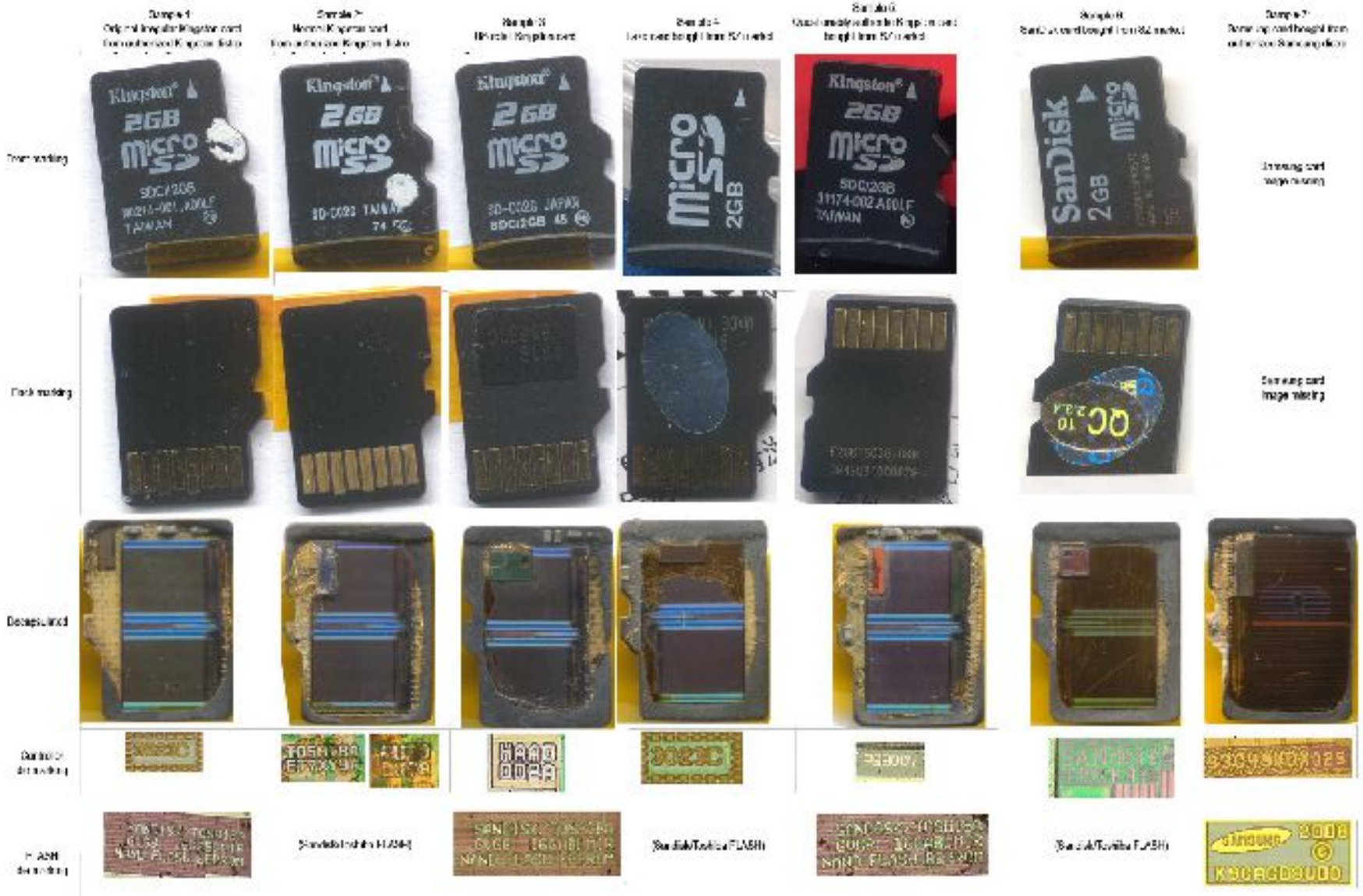
The Exploration and Exploitation of an SD Memory Card

bunnie & xobs

30c3

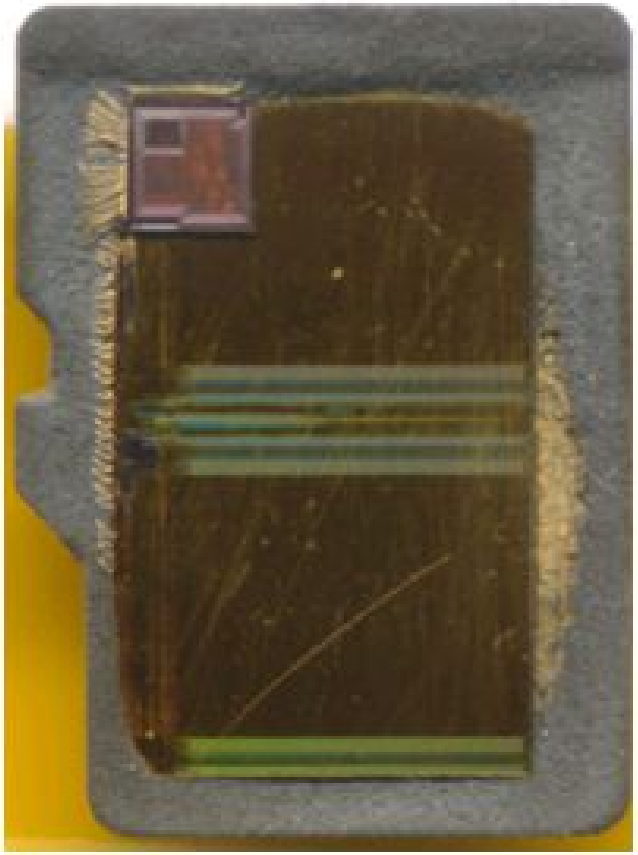
Origin: Searching for Fakes

Card Teardowns



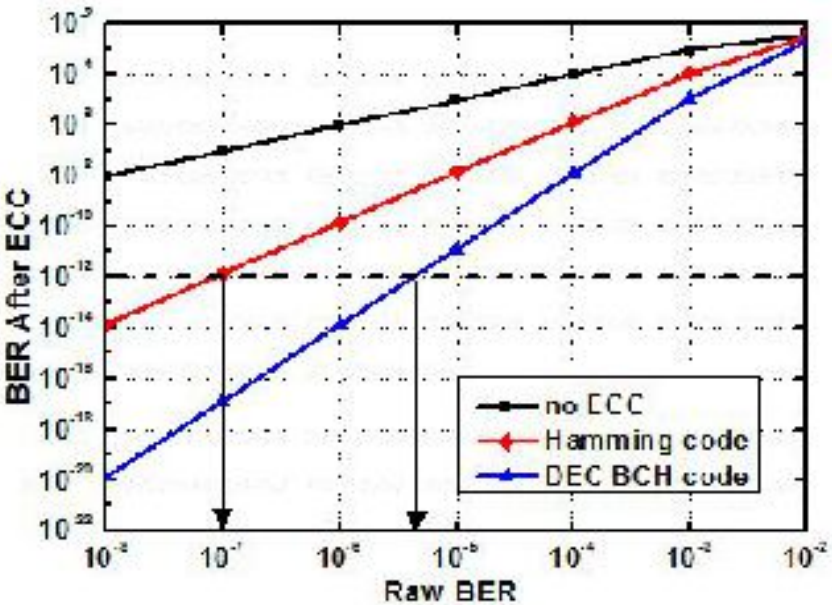
Solution: managed Flash

- Small embedded controller in every "managed Flash" device
  - 8051 or ARM7 CPU
  - 4-8mm^2 silicon = ~$0.15-$0.30 cost add-on
  - Compare to Flash die area = 100mm^2, $2.90 cost
  - Compare to test cost, wafer-scale tester = $1mm = ~$0.45 for a 30 second test (assuming 24 month lifetime and usage 24x7x365)



Faking Reliability

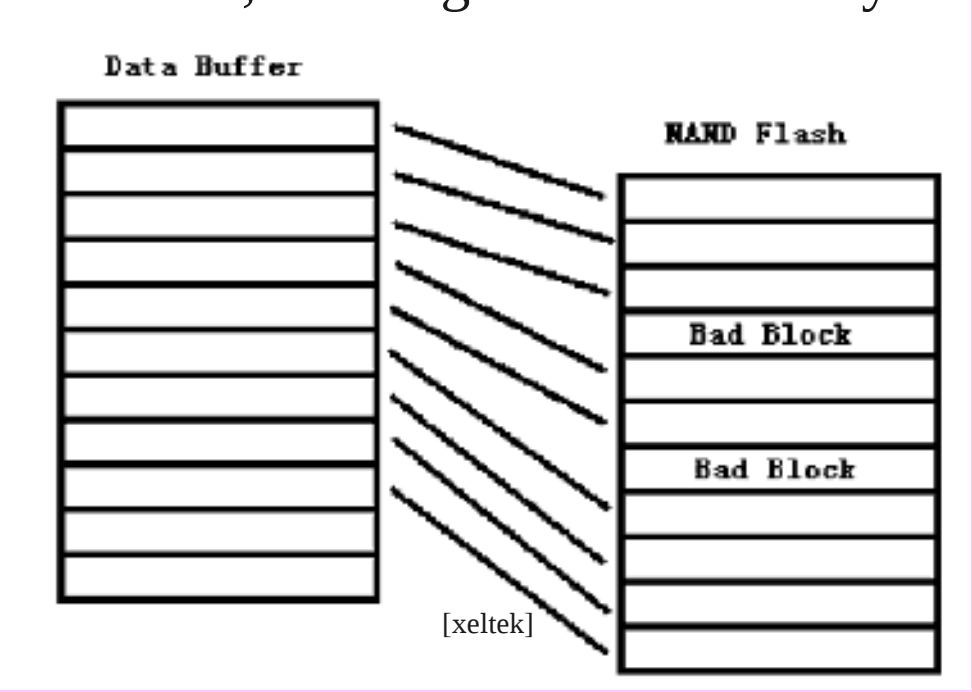
- Flash memory is “unreliable”
  - You are not storing data, you are storing probabilistic approximation of your data
  - Workaround: computational error correction (ECC)



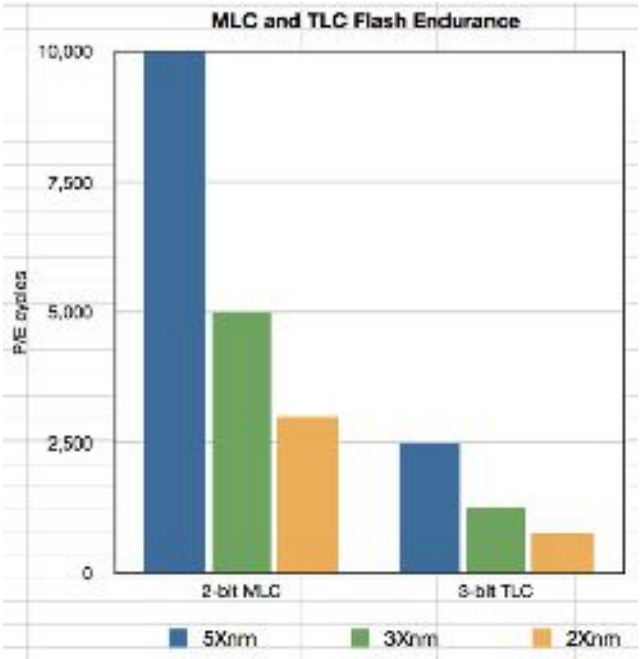
[intechopen.com]

Also, Bad Blocks

- TLC/MLC Flash price is < 0.1nano$/bit
  - Only achievable because every piece of silicon fabricated is sold, regardless of fabrication errors – nothing is thrown away
  - Work around: bad block remapping
    - In some cases, over 80% of blocks are bad (e.g. 16GiB chip sold as 2GiB)
  - Also, blocks go bad with P/E cycles

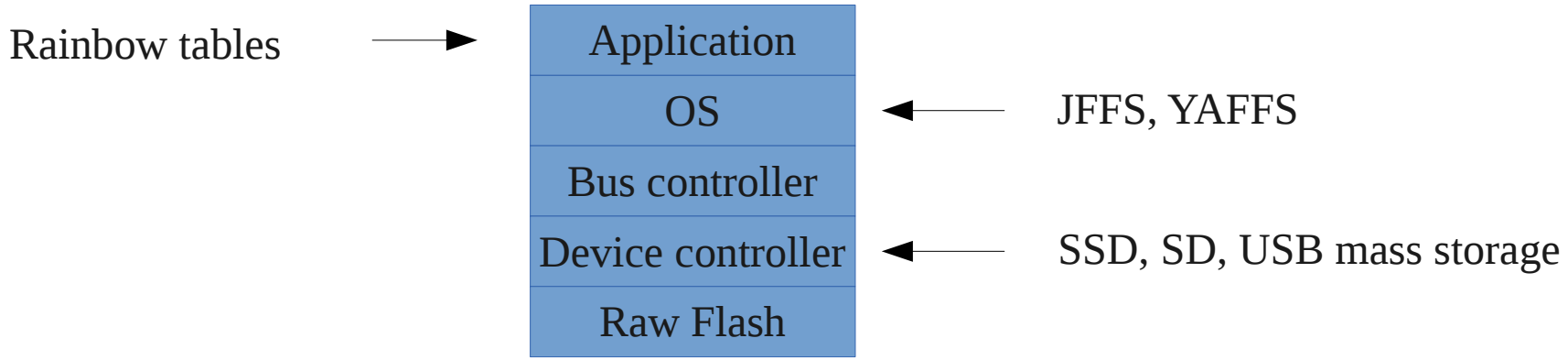


[xeltek]



[theregister.co.uk]

Why do it at this layer?



- Considering:
  - Flash geometry changes every 12-18 mos
    - New ECC rules
    - New page size, block mapping
    - Intensely cost-sensitive market
    - Lowest cost, highest performing Flash chips are proprietary

The Concern

- This is the set up for a MITM attack
- What runs on the microcontroller? Can it be hacked? Can I trust my Flash memory?

Fakes Turn In; New Quest: Hack an SD Card

- Find and hack an SD card
  - Control the micro to make an LED flash, at a bare minimum
  - Challenge: no public docs available on controllers
- Our story
  - Hardware tools developed to inspect, learn, and hack SD cards
  - Software tools and static code analysis to discover back doors and controller structure

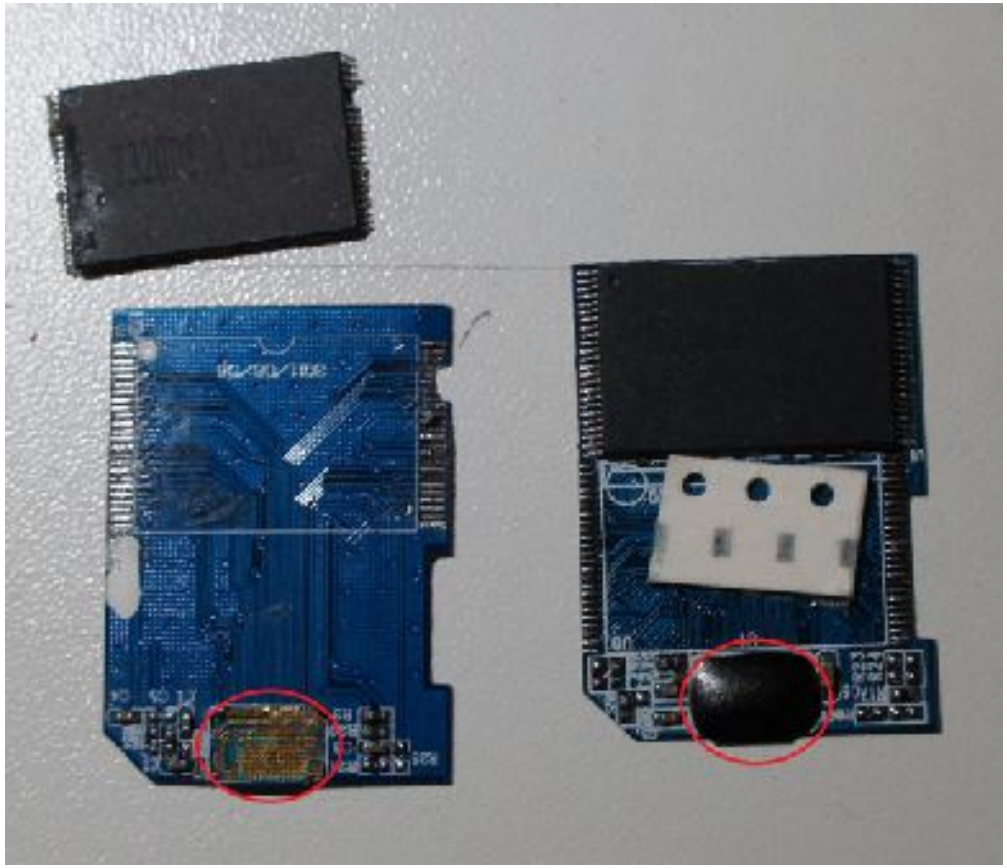
Step 1: Acquire targets

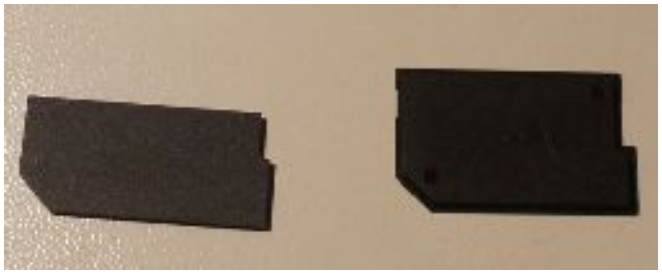
SD Cards Ahoy

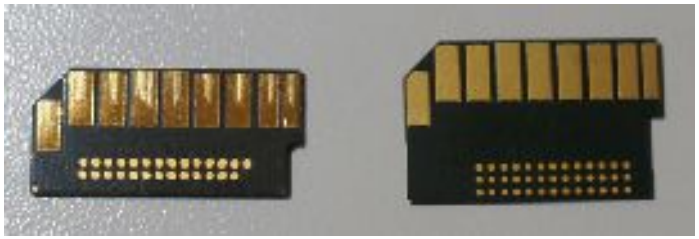
Card Survey


4GB
Kingston
2GB
Kingston
4GB
Kingston
2.0 GB
SanDisk
4GB
Memory Card
ELITE PRO
2.0 GB
SanDisk
ELITE PRO
2GB
Memory Card

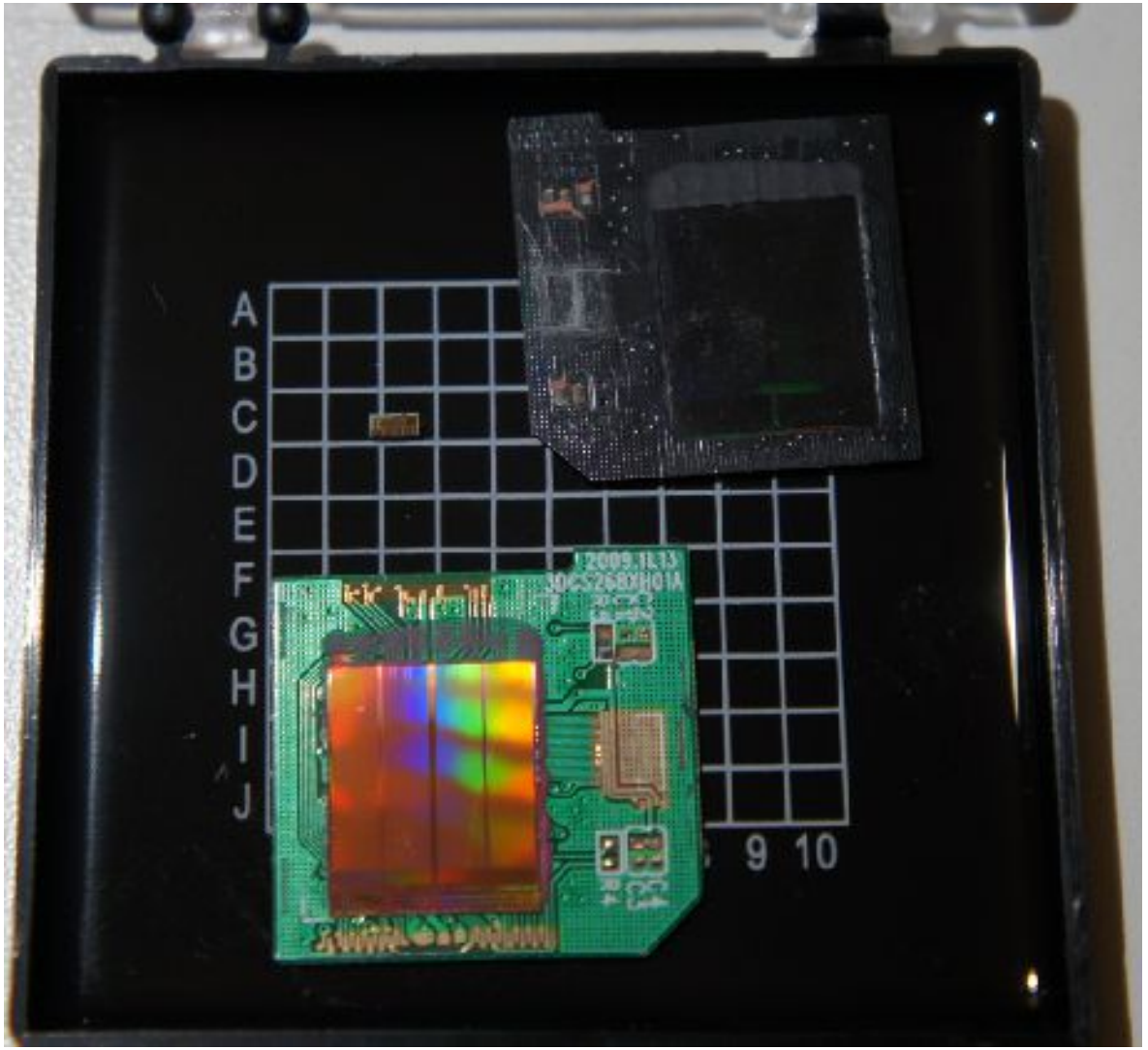
What's inside



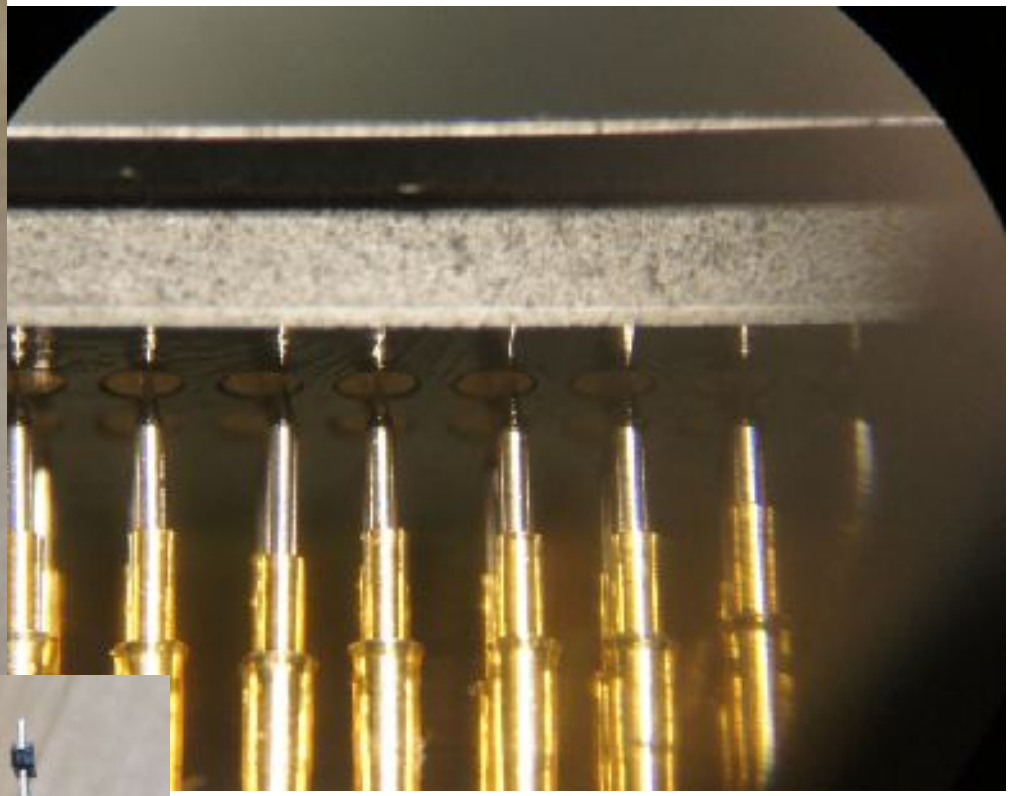


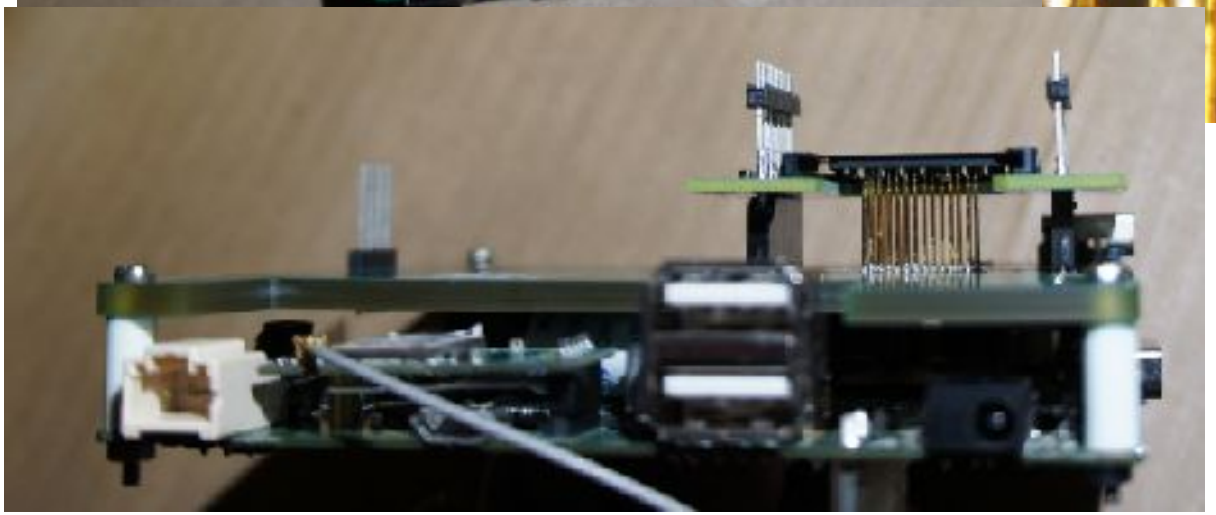


Easy mode decap

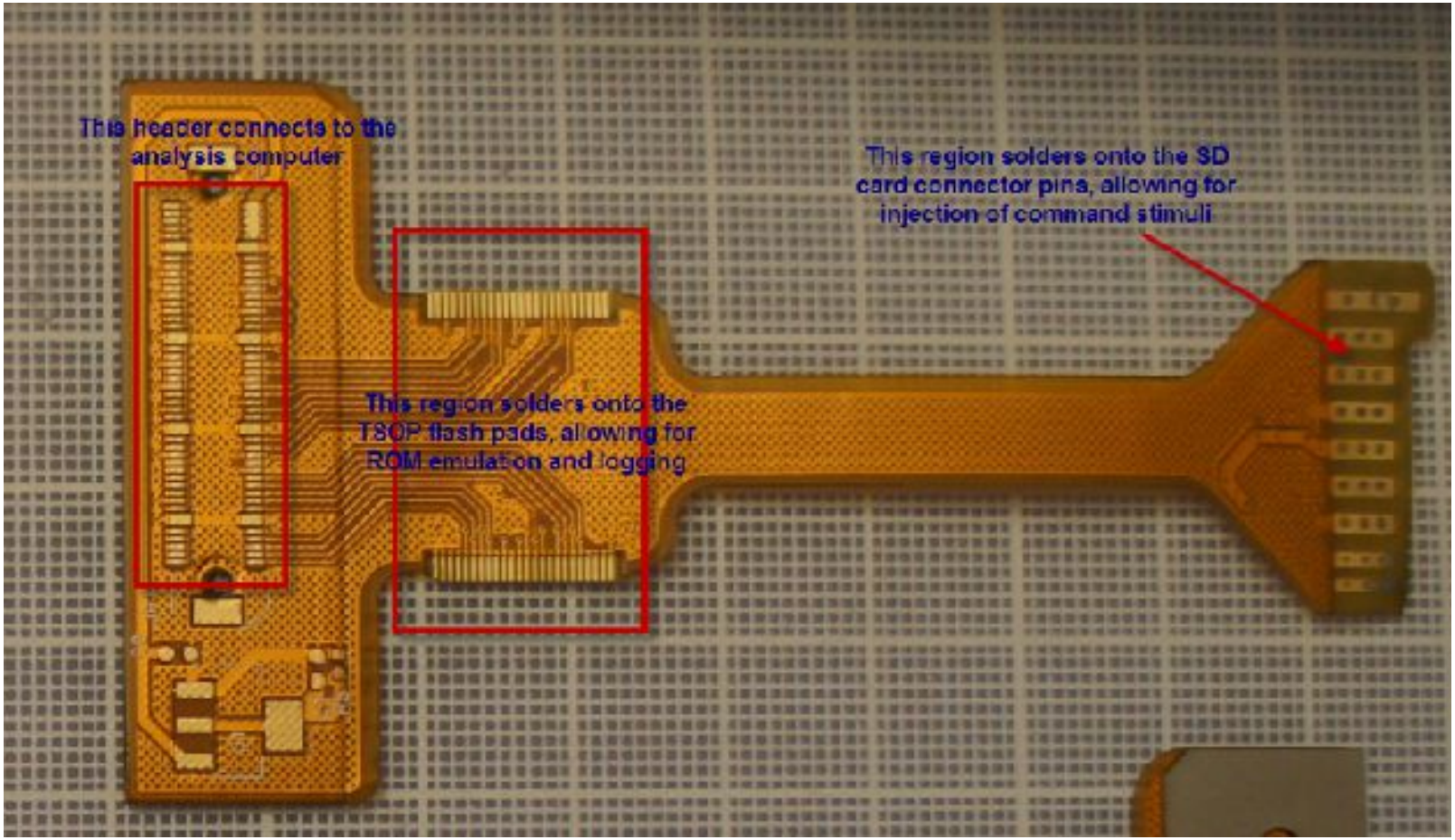


Taps: gen 1 monolithic

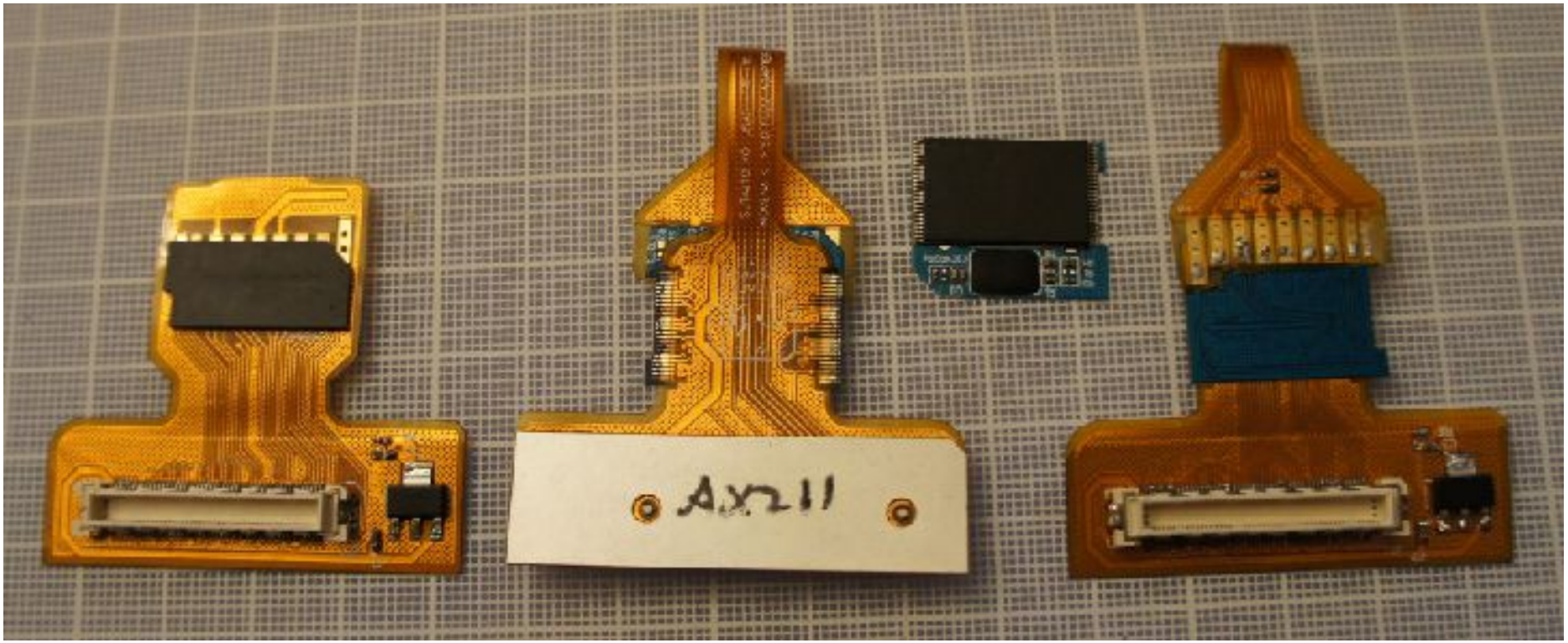




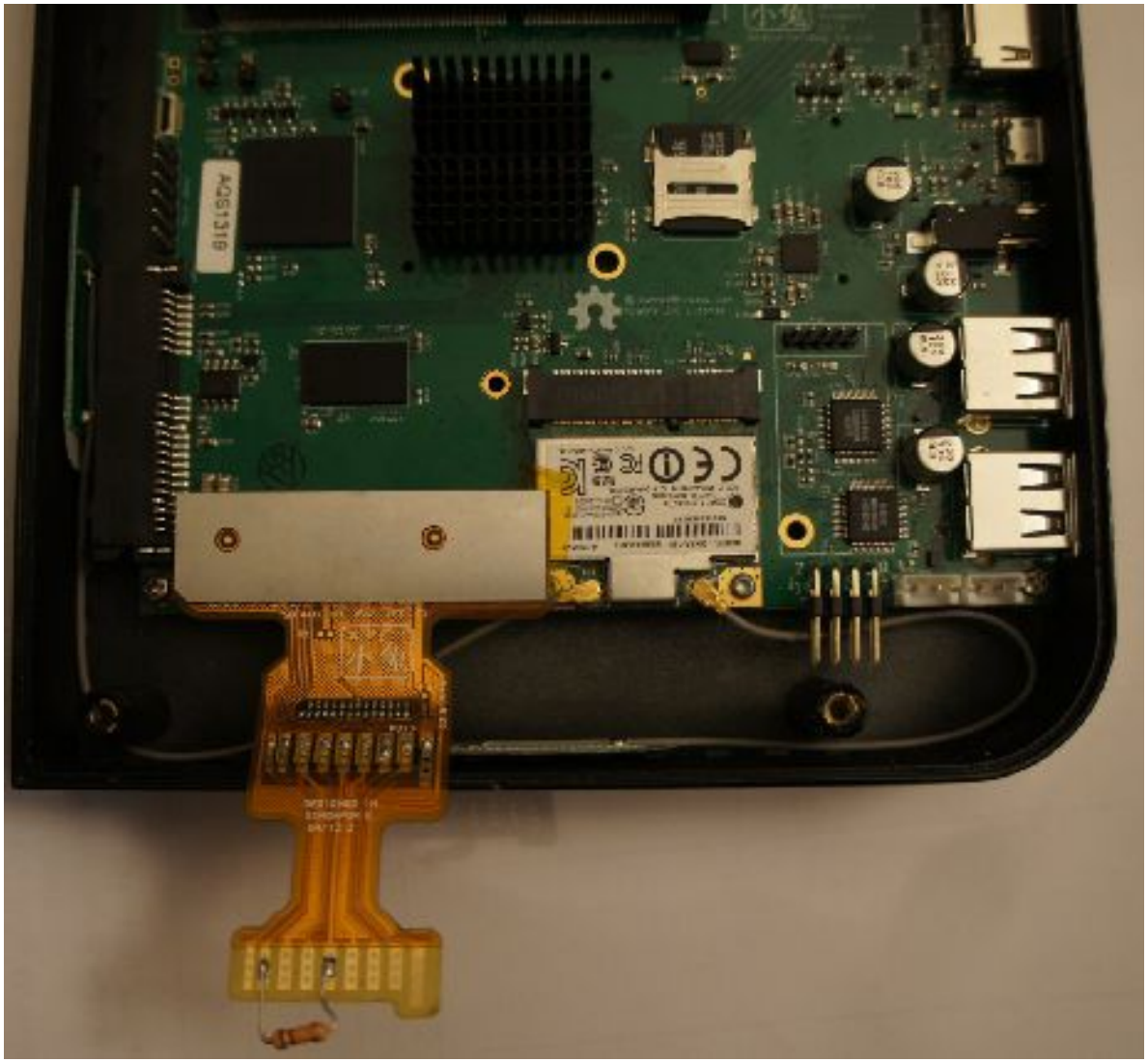
Taps Gen2


This header connects to the analysis computer
This region solders onto the SD card connector pins, allowing for injection of command stimuli
This region solders onto the TSOP flash pads, allowing for ROM emulation and logging

Taps: gen 2, monolithic and discrete

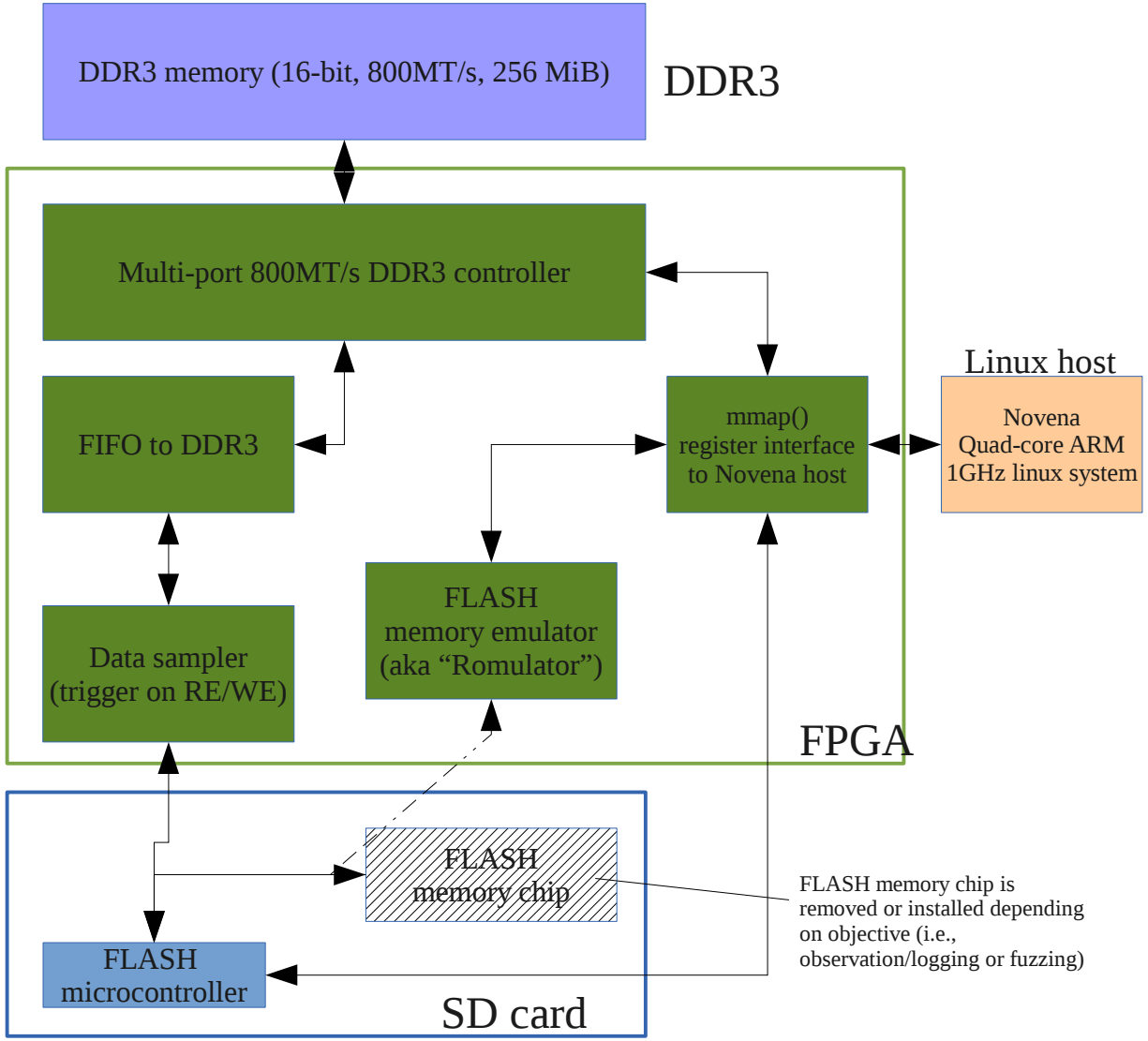


Tap in-system

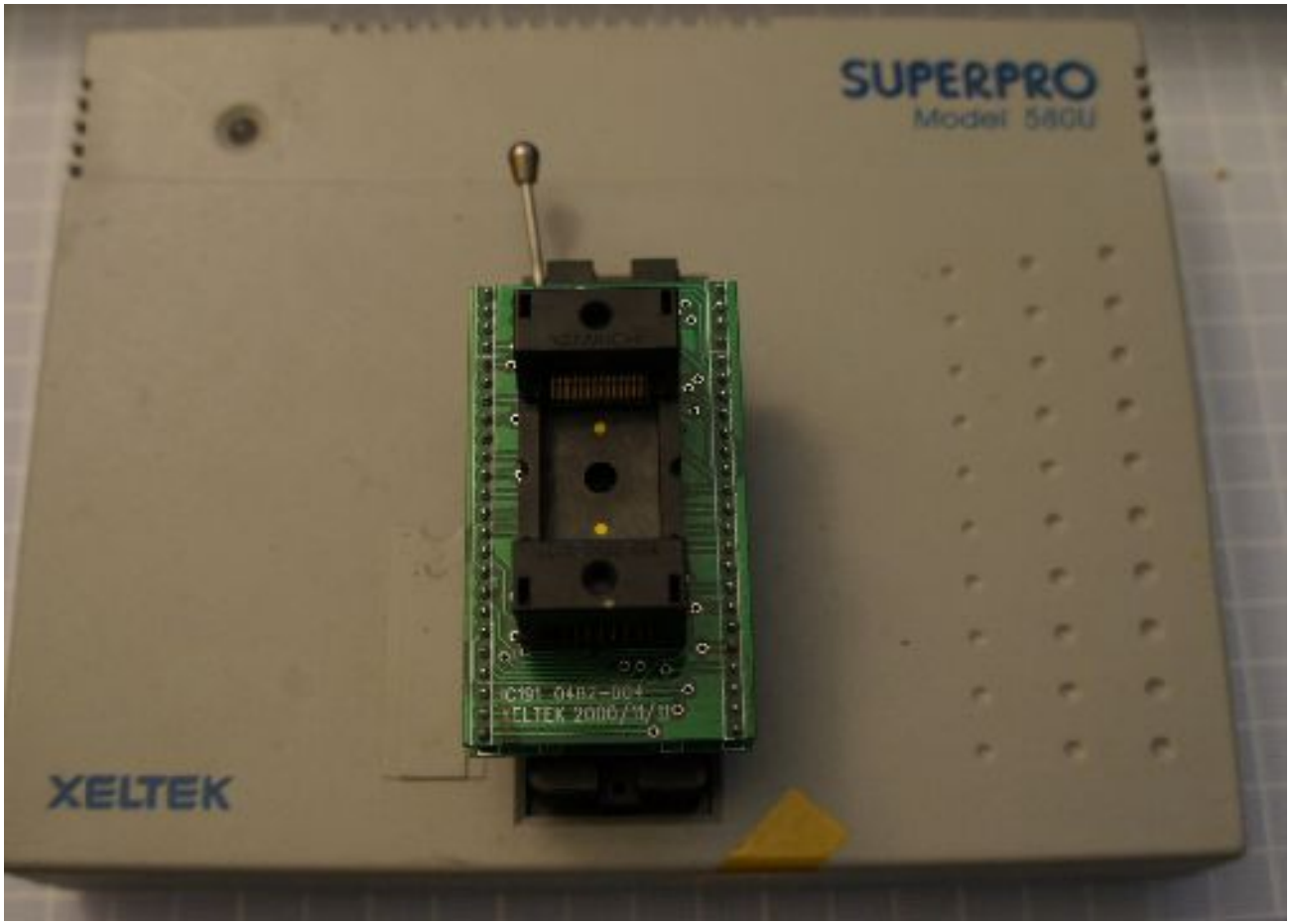


Tapping system diagram

- Capabilities:
  - Flash ROM emulation
    - DDR3 as Flash
    - Dual-port implementation, mod and read on the fly
  - Interface logging
    - Trace capture of SD and Flash interface transactions



ROM reader



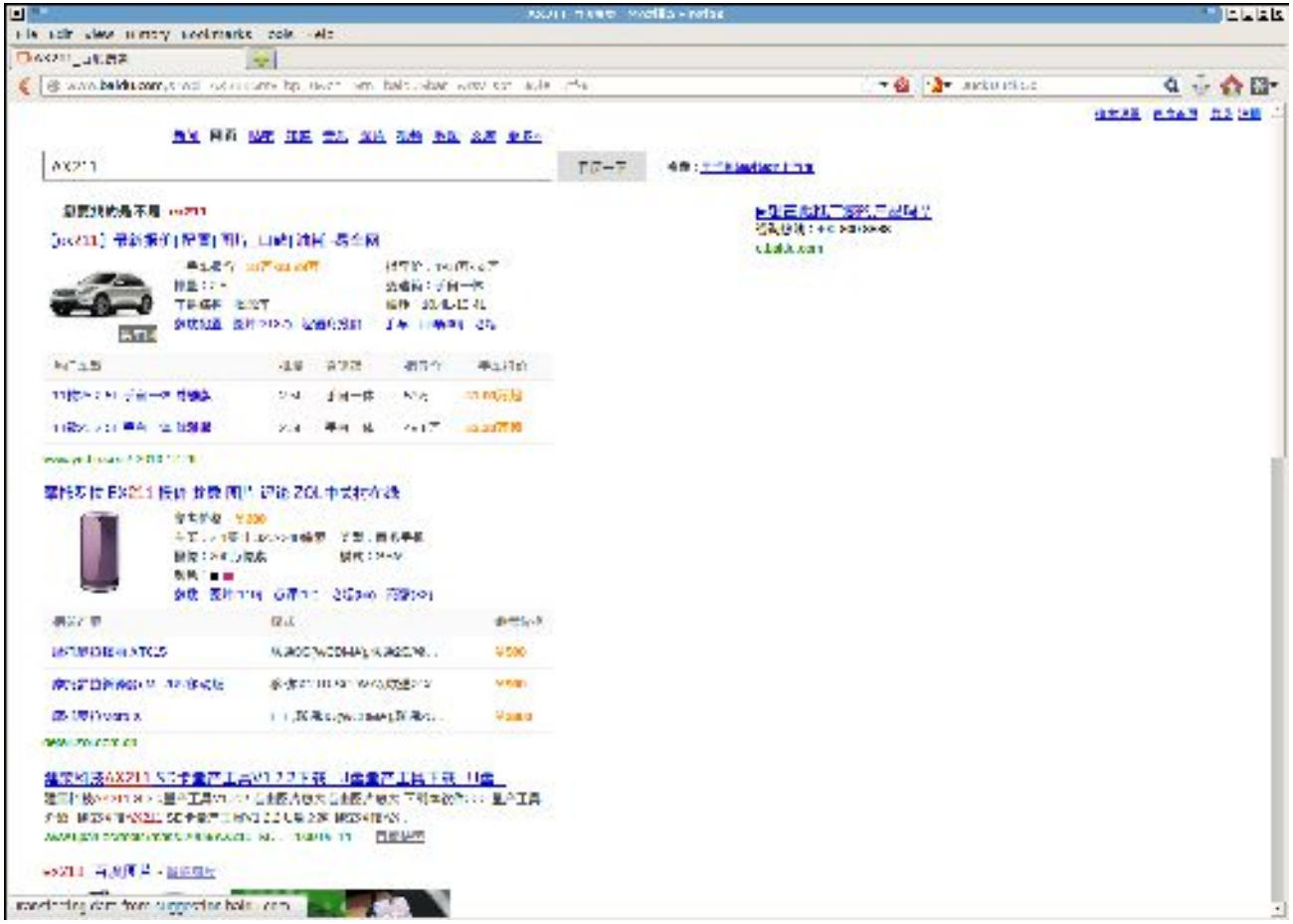
Identifying a target

- Discrete implementation – more hacking options than monolithic
- SLC memory (unscrambled, trivially readable)
  - Easy to check for strings:
    “`China Buildwin SD Controller,Anti-Japig,Author:Y/G/S/P/X Date:2008-7`”
  - Cross-check against google → Appotech controller, likely 8051
    - AX211

Factory Firmware

- Initial code had to get there somehow
  - Try to get ahold of the factory's flashing tool

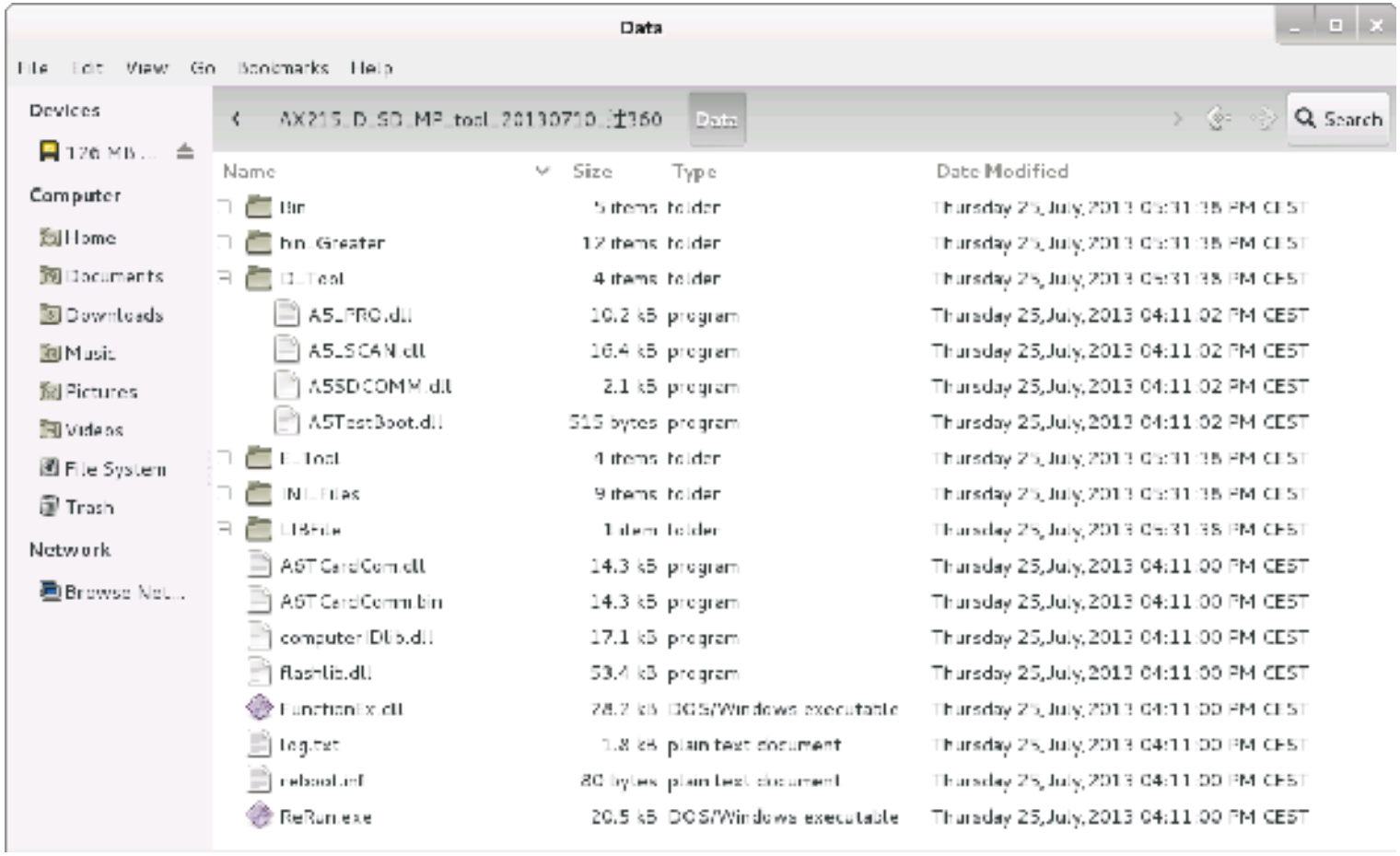
Obtaining software



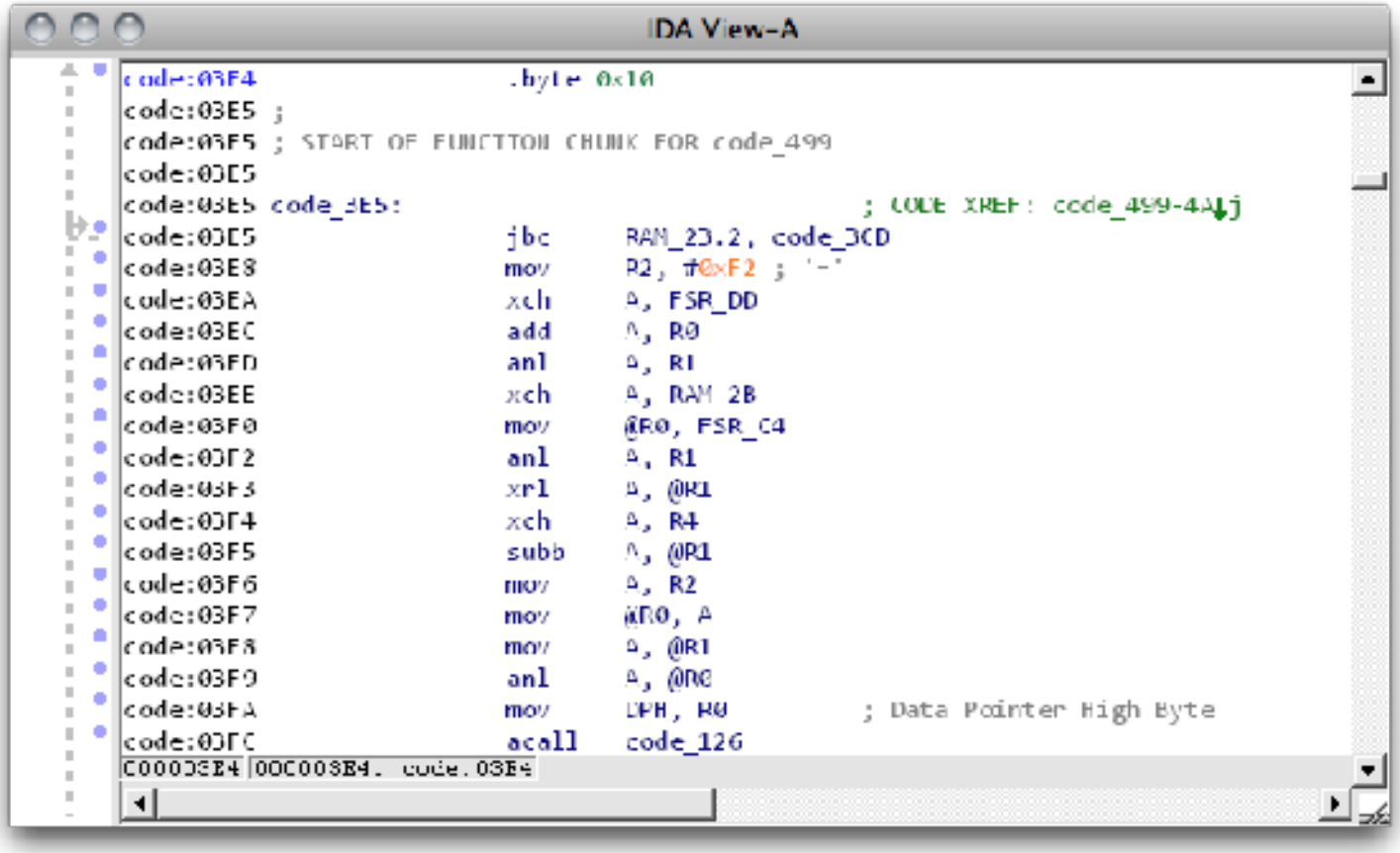
Obtaining software

Programming tool

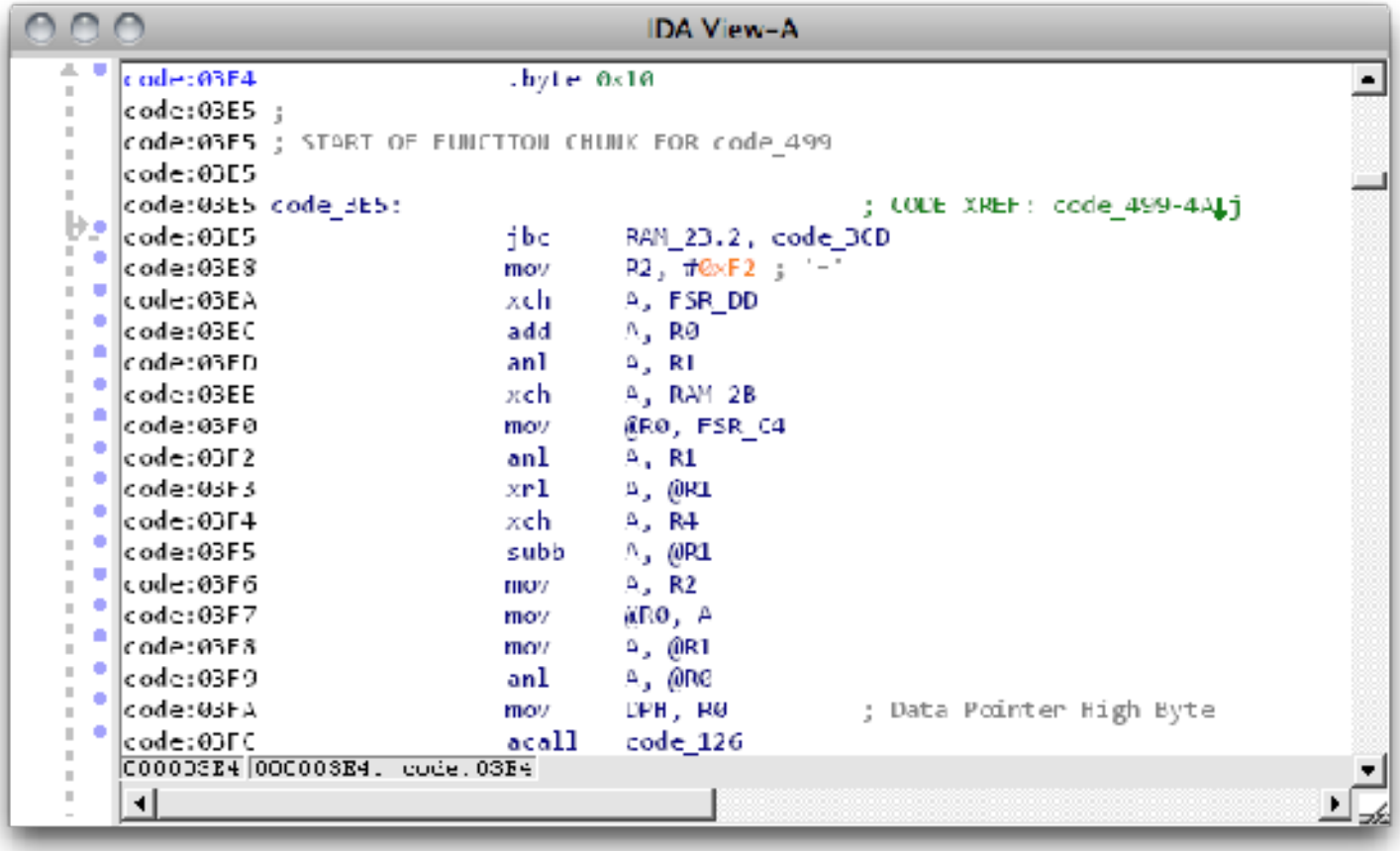
Strange filenames



About the 8051



About the 8051



dd if=/dev/urandom of=firmware.bin bs=2048 count=1

About the 8051

Instructions by opcode

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x00	NOP	AJMP	LJMP	RR	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC
0x10	JBC	ACALL	LCALL	RRC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
0x20	JB	AJMP	RET	RL	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD
0x30	JNB	ACALL	RETI	RLC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC
0x40	JC	AJMP	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL
0x50	JNC	ACALL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL
0x60	JZ	AJMP	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL
0x70	JNZ	ACALL	ORL	JMP	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x80	SJMP	AJMP	ANL	MOVC	DIV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x90	MOV	ACALL	MOV	MOVC	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB
0xa0	ORL	AJMP	MOV	INC	MUL	?	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xb0	ANL	ACALL	CPL	CPL	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE
0xc0	PUSH	AJMP	CLR	CLR	SWAP	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH
0xd0	POP	ACALL	SETB	SETB	DA	DJNZ	XCHD	XCHD	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ
0xe0	MOVX	AJMP	MOVX	MOVX	CLR	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xf0	MOVX	ACALL	MOVX	MOVX	CPL	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV

http://www.win.tue.nl/~aeb/comp/8051/set8051.html

About the 8051

Instructions by opcode

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x00	NOP	AJMP	LJMP	RR	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC	INC
0x10	JBC	ACALL	LCALL	RRC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
0x20	JB	AJMP	RET	RL	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD
0x30	JNB	ACALL	RETI	RLC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC	ADDC
0x40	JC	AJMP	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL	ORL
0x50	JNC	ACALL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL	ANL
0x60	JZ	AJMP	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL	XRL
0x70	JNZ	ACALL	ORL	JMP	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x80	SJMP	AJMP	ANL	MOVC	DIV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0x90	MOV	ACALL	MOV	MOV	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB	SUBB
0xa0	ORL	AJMP	MOV	INC	MUL	?	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xb0	ANL	ACALL	CPL	CPL	[illegible]	[illegible]	[illegible]	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE	CJNE
0xc0	PUSH	AJMP	CLR	CLR	SWAP	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH	XCH
0xd0	POP	ACALL	SETB	SETB	DA	DJNZ	XCHD	XCHD	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ	DJNZ
0xe0	MOVX	AJMP	MOVX	MOVX	CLR	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
0xf0	MOVX	ACALL	MOVX	MOVX	CPL	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV

http://www.win.tue.nl/~aeb/comp/8051/set8051.html

About the AX211

卓荣集成电路科技有限公司
AppoTech Limited

AX211 高性能 32位SD卡控制器

CPU 特性

- 单周期32位 RISC CPU，优化支持SD, NAND 应用
- 最大 50 MIPS 的性能，自带 RC 振荡器和PLL

SD接口特性

- 完全支持SD 卡 1.0/1.1/2.0标准
- 支持SD 高容量标准
- 支持 SPI 模式
- 支持Command class 0/2/4/5/6/7/8/10
- 支持CPRM
- 支持Speed class 6
- 支持主机时钟频率达50MHz
- 支持Bus宽度X1/X4
- 支持 SD 主机功能
- 加强的ESD 保护

NAND Flash 接口特性

- 支持小页或大页NAND flash
- 支持 SLC/MLC NAND flash
- 支持Two-Plane 或 Interleave NAND Flash
- 支持 X8/X16 NAND flash
- 支持并行模式
- 支持 ECC
- 内置6-54位/512B (528 字节) on-the-fly ECC
- 当断电操作时自动保护数据

低功耗支持

- 当数据传输时，工作电流约为10mA
- 支持睡眠模式，当芯片处于睡眠模式，工作电流低于200uA
- 支持快速唤醒

封装

- TQFP48或QFN48
- 裸片形式

CPU Features

- 1T 32-bit RISC CPU, optimized for SD, NAND Flash applications
- MAX 50 MIPS performance with on-chip RC oscillator and PLL

SD Interface Features

- Fully Supports SD card standard 1.0/1.1/2.0
- Supports SD high capacity standard
- Supports SPI mode
- Supports Command class 0/2/4/5/6/7/8/10
- Supports CPRM
- Speed class up to 6
- Supports host clock up to 50MHz
- Supports bus width X1/X4
- Supports SD host function
- Enhanced ESD protection

NAND Flash Interface Features

- Supports small or large page NAND flash
- Supports SLC or MLC NAND flash
- Supports Two-Plane or Interleave NAND flash
- Supports X8/X16 NAND flash
- Supports parallel mode
- Supports up to 8 CE
- Built in 6-54 bit (page/528 bytes) on-the-fly ECC
- Data protection during data transfer if unplugged /power off

Low Power Consumption

- Operating current is about 10mA during data transfer
- Supports sleep mode, current is less than 200uA during Sleep Mode
- Fast wake up during Sleep Mode

Package

- 48-pin TQFP or QFN package
- Die form

www.appotech.com
www.buildwin.com.cn

About the AX211

CPU Features

- 1T 32-Bit RISC CPU, optimized for SD, NAND Flash applications
- MAX 50 MIPS performance with on chip RC oscillator and PLL

SD Interface Features

- Fully Supports SD card standard 1.0/1.1/2.0
- Supports SD high capacity standard
- Supports SPI mode
- Supports Command class 0/2/4/5/6/7/8/10
- Supports CPRM
- Speed class up to 6
- Supports host clock up to 50MHz
- Supports bus width X1/X4
- Supports SD host function
- Enhanced ESD protection

NAND Flash Interface Features

- Supports small or large page NAND flash
- Supports SLC or MLC NAND flash
- Supports Two-Plane or Interleave NAND flash
- Supports X8/X16 NAND flash
- Supports parallel mode
- Supports up to 8 CE
- Built in 6-54bit/page(528 bytes) on-the-fly ECC
- Data protection during data transfer if unplugged /power off

Low Power Consumption

- Operating current is about 10mA during data transfer
- Supports Sleep Mode, current is less than 200uA during Sleep Mode
- Fast wake up during Sleep Mode

Package

- 48-pin TQFP or QFN package
- Die form

About the AX211

CPU Features

- 1T 32-Bit RISC CPU, optimized for SD, NAND Flash applications
- Up to 50 MIPS performance with on chip RC oscillator and PLL

SD Interface Features

- Fully Supports SD card standard 1.0/1.1/2.0
- Supports SD high capacity standard
- Supports SPI mode
- Supports Command class 0/2/4/5/6/7/8/10
- Supports CPRM
- Speed class up to 6
- Supports host clock up to 50MHz
- Supports bus width X1/X4
- Supports SD host function
- Enhanced ESD protection

NAND Flash Interface Features

- Supports small or large page NAND flash
- Supports SLC or MLC NAND flash
- Supports Two-Plane or Interleave NAND flash
- Supports X8/X16 NAND flash
- Supports parallel mode
- Supports up to 8 CE
- Built in 6-54bit/page(528 bytes) on-the-fly ECC
- Data protection during data transfer if unplugged /power off

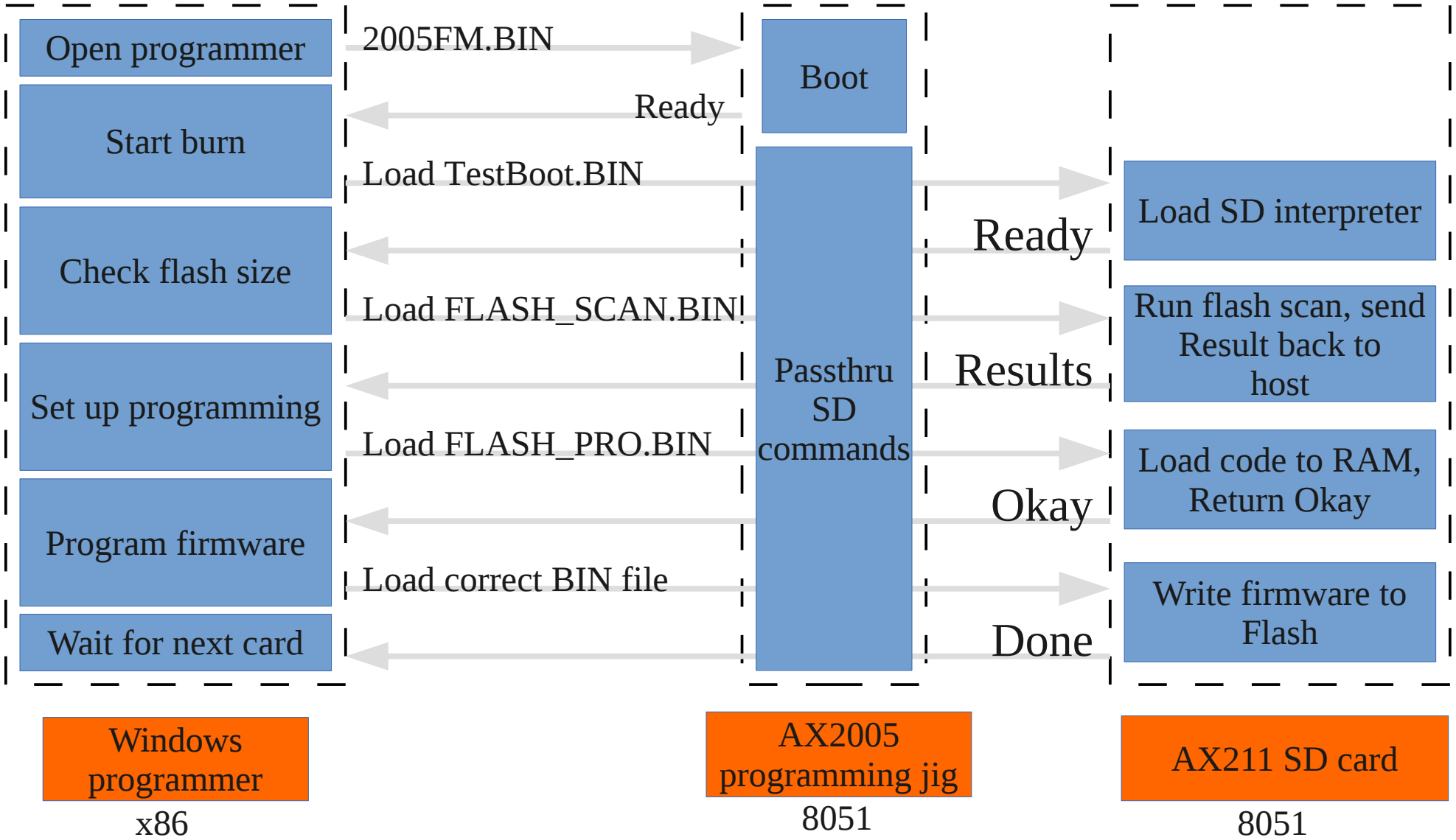
Low Power Consumption

- Operating current is about 10mA during data transfer
- Supports Sleep Mode, current is less than 200uA during Sleep Mode
- Fast wake up during Sleep Mode

Package

- 48-pin TQFP or QFN package
- Die form

Programming process



SD Protocol: Hardware

- Signals:
  - CMD
  - DAT0 – DAT3
  - CLK
- Signal integrity
  - Commands use CRC7
  - Data uses CRC16
- Also supports SPI mode

SD Protocol: Software

- 64 Possible Commands
  - CMD0: Reset / Go Idle
  - CMD10: Get CID
  - CMD41: ACMD “escape”
  - CMD60 – CMD63: Reserved for mfgr
- 32 bits of “argument” data

0	1	bit 5...bit 0	bit 31...bit 0	bit 6...bit 0	1
start bit	host	command	argument	CRC7[1]	end bit

[SanDisk Product Manual V1.9]

SD Protocol: Response

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	1
Value	'0'	'0'	x	x	x	'1'
Description	start bit	transmission bit	command index	card status	CRC7	end bit

Table 4-29: Response R1

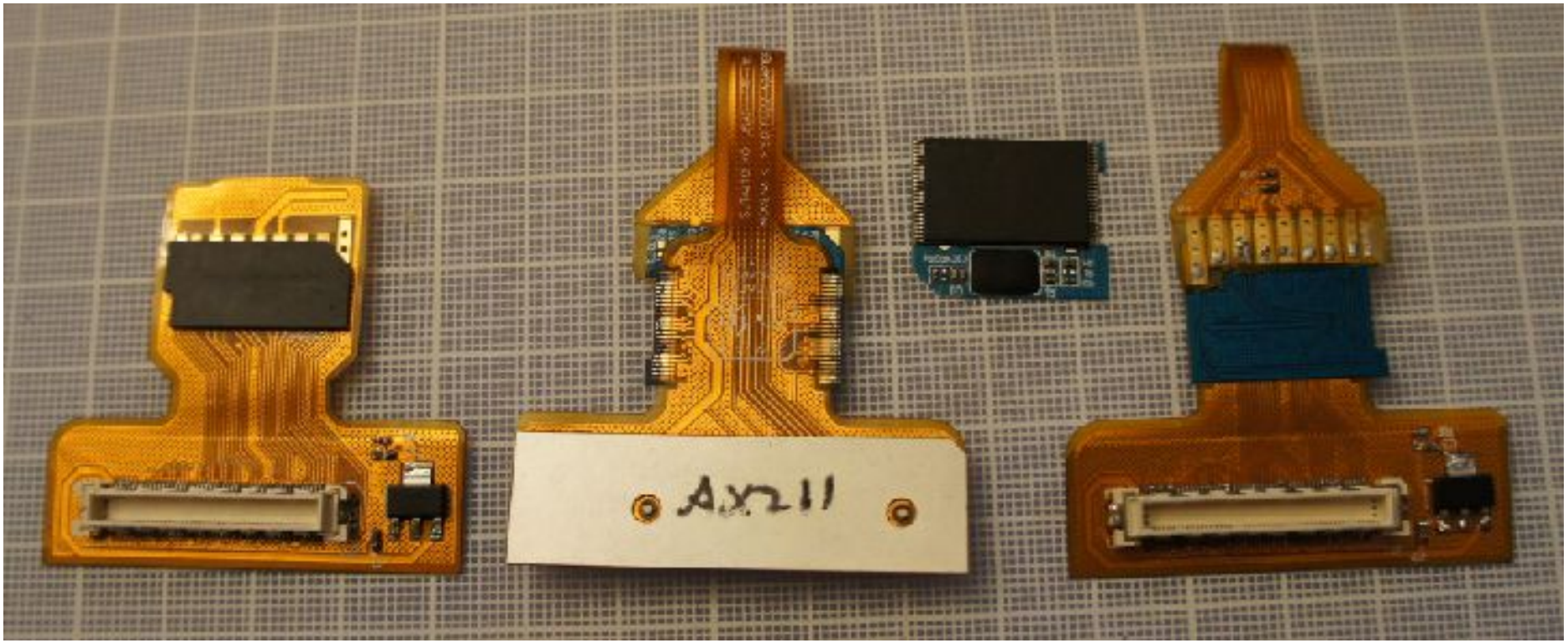
25	CARD_IS_LOCKED	S X	'0' = card unlocked '1' = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_FAILED	E R X	'0' = no error '1' = error	Set when a sequence or password error has been detected in lock/unlock card command.	C
23	COM_CRC_ERROR	E R	'0'= no error '1'= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	E R	'0'= no error '1'= error	Command not legal for the card state	B
21	CARD_ECC_FAILED	E R X	'0'= success '1'= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	E R X	'0'= no error '1'= error	Internal card controller error	C
19	ERROR	E R X	'0'= no error	A general or an unknown error	C

[SD Simplified Layer Spec]

Fuzzing knock sequence

- 64 possible commands
  - Only 4 “manufacturer” commands
  - 2^{32} possible arguments
- Fuzz sequence:
  - Reset card
  - Send random command/argument
  - Check for a response
  - No response means it may have crashed

Still works!



No success

- Huge number of possibilities
- Fuzzer can run non-interactively
- Try a different approach
  - Look at the firmware burner

Programming jig

- AX2005
- Bit-banged SD

IDA View-A

```
ROM:4DE3 ; =============== S U B R O U T I N E =======================================
ROM:4DE3
ROM:4DE3
ROM:4DE3 toggle_clk_once:                        ; CODE XREF: ROM_2B76:ROM_2B81↑p
ROM:4DE3                                         ; ROM_2D7G:ROM_2D9C↑p ...
ROM:4DE3                 jb      RAM_227, toggle_clk2
ROM:4DE6                 anl     FSR_80, #0xEF
ROM:4DE9                 nop
ROM:4DEA                 orl     FSR_80, #0x10
ROM:4DED                 ret
ROM:4DEE ; ---------------------------------------------------------------------------
ROM:4DEE
ROM:4DEE toggle_clk2:                            ; CODE XREF: toggle_clk_once↑j
ROM:4DEE                 anl     FSR_C0, #0xEF
ROM:4DF1                 nop
ROM:4DF2                 orl     FSR_C0, #0x10
ROM:4DF5                 ret
ROM:4DF5 ; End of function toggle_clk_once
00004DED 0C004DE3: toggle_clk_once+A
```

Running code

- Noticed 'APPO' in AX2005 firmware
- Preceeded by #63
- Maybe the knock is “CMD63 APPO”
- Card seems to respond
  - Doesn't say “invalid command”
  - Doesn't respond at all for 130 cycles
  - If CRC16 is valid, card stops responding at all

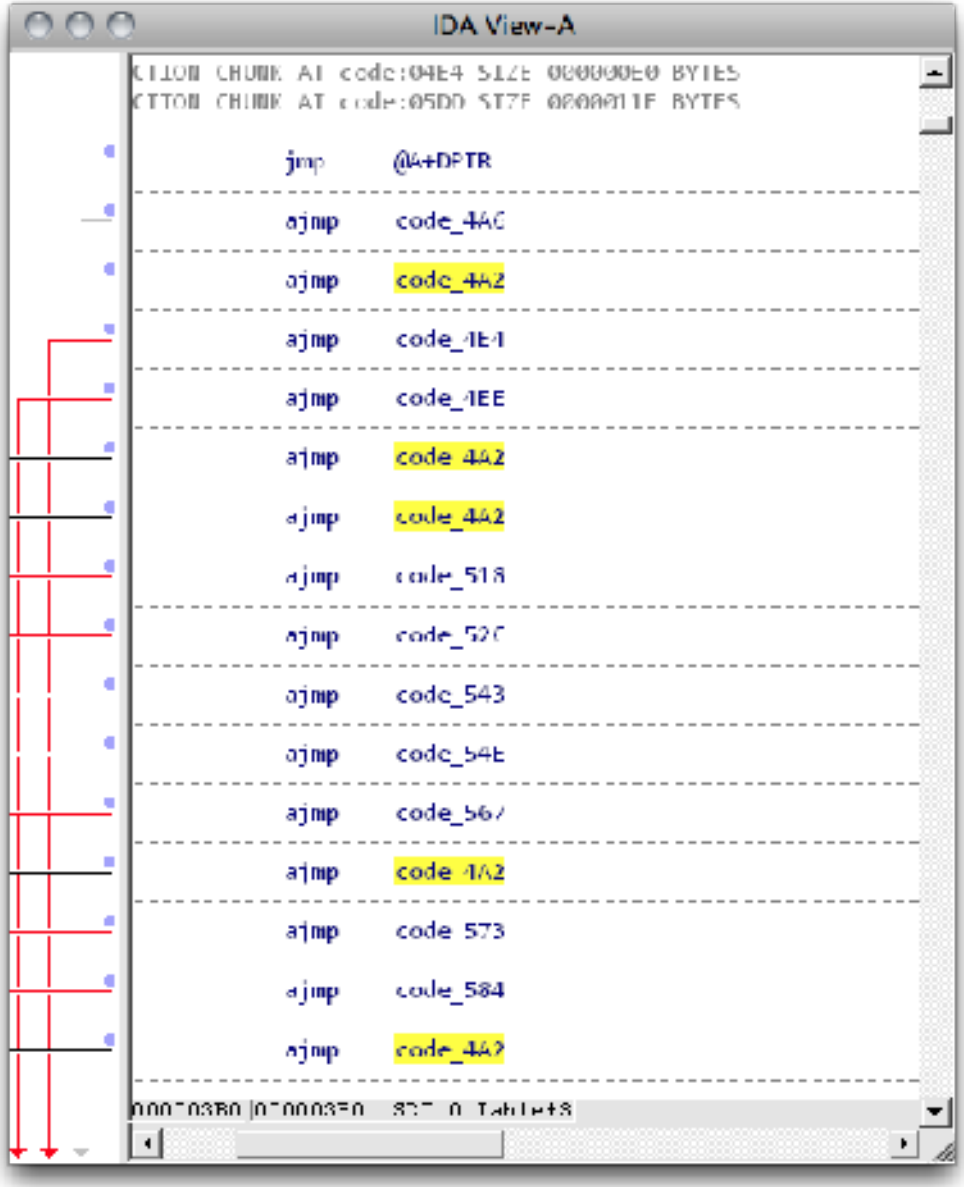
Writing a debugger

- We can run code. Great!
- We don't know what to run! Darn.
- Debugger can go over SD
- We have example code

TestBoot.bin

- 512 bytes
- Easy to analyze
- Tells us entry point
- Contains SD state machine

Also, Original Card Firmware Dump



CMD INDEX	type	argument	resp	abbreviation	command description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all cards to idle state
CMD1	reserved				
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks any card to send the CID numbers on the CMD line (any card that is connected to the host will respond)
CMD3	bcr	[31:0] stuff bits	R6	SEND_RELATIVE_ADDR	Ask the card to publish a new relative address (RCA)
CMD4	bc	[31:16] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards
CMD5	reserved for SDIO cards (refer to the "SDIO Card Specification")				
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1b (only from the selected card)	SELECT/DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address; address 0 deselects all. In the case that the RCA equals 0, then the host may do one of the following: - Use other RCA number to perform card de-selection. - Re-send CMD3 to change its RCA number to other than 0 and then use CMD7 with RCA=0 for card de-selection.
CMD8	bcr	[31:12]reserved bits [11:8]supply voltage(VHS) [7:0]check pattern	R7	SEND_IF_COND	Sends SD Memory Card interface condition, which includes host supply voltage information and asks the card whether card supports voltage. Reserved bits shall be set to '0'.
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card identification (CID) on CMD the line.
CMD11	reserved				
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission
CMD13	ac	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	reserved				

[SD Simplified Layer Spec]

Writing a debugger

- Borrow TestBoot.bin
  - Code doesn't work out of the box
- No debugger whatsoever
  - Maybe we can wiggle a pin?

GPIO hunting

- Probably 1 – 3 registers
  - Set/Clear register value
  - Set/Clear pullup
  - Set pin function
- Toggle them with some frequency

Fuzzer

- Generate an 8051 program that:
  - Pokes value to a random SFR
  - Delays a while
  - Changes SFR value
  - Delays again
  - Repeat
- Read GPIO input values on host
  - Watch for toggling pins

“Hello, World” that finally worked!

```
fuzz:
        mov     0xef, #0x00
        acall   sleep
        mov     0xef, #0xff
        acall   sleep
        sjmp    fuzz

sleep:
        mov     R5, #0xff
        mov     R6, #0x20
top_of_pause:
        djnz    R5, top_of_pause
        djnz    R6, top_of_pause
        ret
```

“Hello, World”

```
Observed 65 changes:
00000000  57 57 57 57 57 57 57 47  47 47 47 47 47 57 57 57  |WWWWWWWGGGGGGWWW|
00000010  57 57 57 57 57 57 47 47  47 47 47 47 47 57 57 57  |WWWWWWGGGGGGGWWW|
00000020  57 57 57 57 57 57 47 47  47 47 47 47 47 57 57 57  |WWWWWWGGGGGGGWWW|
00000030  57 57 57 57 57 57 47 47  47 47 47 47 47 57 57 57  |WWWWWWGGGGGGGWWW|
00000040  57 57 57 57 57 57 47 47  47 47 47 47 57 57 57 57  |WWWWWWGGGGGGWWWW|
00000050  57 57 57 57 57 47 47 47  47 47 47 47 57 57 57 57  |WWWWWGGGGGGGWWWW|
00000060  57 57 57 57 57 47 47 47  47 47 47 47 57 57 57 57  |WWWWWGGGGGGGWWWW|
00000070  57 57 57 57 57 47 47 47  47 47 47 47 57 57 57 57  |WWWWWGGGGGGGWWWW|
00000080  57 57 57 57 57 47 47 47  47 47 47 47 57 57 57 57  |WWWWWGGGGGGGWWWW|
00000090  57 57 57 57 57 47 47 47  47 47 47 47 57 57 57 57  |WWWWWGGGGGGGWWWW|
000000a0  57 57 47 47 47 47 47 47  47 57 57 57 57 57 57 57  |WWGGGGGGGWWWWWWW|
000000b0  57 57 47 47 47 47 47 47  47 57 57 57 57 57 57 57  |WWGGGGGGGWWWWWWW|
000000c0  57 57 47 47 47 47 47 47  47 57 57 57 57 57 57 57  |WWGGGGGGGWWWWWWW|
000000d0  57 57 47 47 47 47 47 47  47 57 57 57 57 57 57 57  |WWGGGGGGGWWWWWWW|
000000e0  57 57 47 47 47 47 47 47  47 57 57 57 57 57 57 57  |WWGGGGGGGWWWWWWW|
000000f0  57 57 47 47 47 47 47 47  57 57 57 57 57 57 57 57  |WWGGGGGGWWWWWWWW|
00000100  57 47 47 47 47 47 47 47  57 57 57 57 57 57 57 57  |WGGGGGGGWWWWWWWW|
00000110  57 47 47 47 47 47 47 47  57 57 57 57 57 57 57 57  |WGGGGGGGWWWWWWWW|
00000120  57 47 47 47 47 47 47 47  57 57 57 57 57 57 57 57  |WGGGGGGGWWWWWWWW|
00000130  57 47 47 47 47 47 47 47  57 57 57 57 57 57 57 57  |WGGGGGGGWWWWWWWW|
00000140  57 47 47 47 47 47 47 47  57 57 57 57 57 57 57 57  |WGGGGGGGWWWWWWWW|
00000150  57 47 47 47 47 47 47 57  57 57 57 57 57 57 57 57  |WGGGGGGWWWWWWWWW|
00000160  47 47 47 47 47 47 47 57  57 57 57 57 57 57 57 57  |GGGGGGGWWWWWWWWW|
00000170  47 47 47 47 47 47 47 57  57 57 57 57 57 57 57 57  |GGGGGGGWWWWWWWWW|
00000180  47 47 47 47 47 47 47 57  57 57 57 57 57 57 57 57  |GGGGGGGWWWWWWWWW|
00000190  47 47 47 47 47 47 47 57  57 57 57 57 57 57 57 57  |GGGGGGGWWWWWWWWW|
000001a0  47 47 47 47 47 47 47 57  57 57 57 57 57 57 57 57  |GGGGGGGWWWWWWWWW|
000001b0  47 47 47 47 47 47 57 57  57 57 57 57 57 57 57 57  |GGGGGGWWWWWWWWWW|
000001c0  47 47 47 47 47 47 57 57  57 57 57 57 57 57 57 47  |GGGGGGWWWWWWWWWG|
000001d0  47 47 47 47 47 47 57 57  57 57 57 57 57 57 57 47  |GGGGGGWWWWWWWWWG|
000001e0  47 47 47 47 47 47 57 57  57 57 57 57 57 57 57 47  |GGGGGGWWWWWWWWWG|
000001f0  47 47 47 47 47 47 57 57  57 57 57 57 57 57 57 47  |GGGGGGWWWWWWWWWG|
```

Writing a Debugger

- Bidirectional SD communications
  - Send CMD with four 8-byte arguments
  - Get CMD back with four 8-byte responses
- Basic commands
  - peek/poke
  - GPIO control
  - IRQ status
  - NAND emulator
  - 32-bit opcodes?
- https://github.com/xobs/ax2xx-code

0xa5 “Escape” opcode

- Undefined in standard 8051
- All over the place in AX211 code
- 0xa5 0xXY
- 0xa5 0x7Y 0xWZ

8 bit or 32 bit?

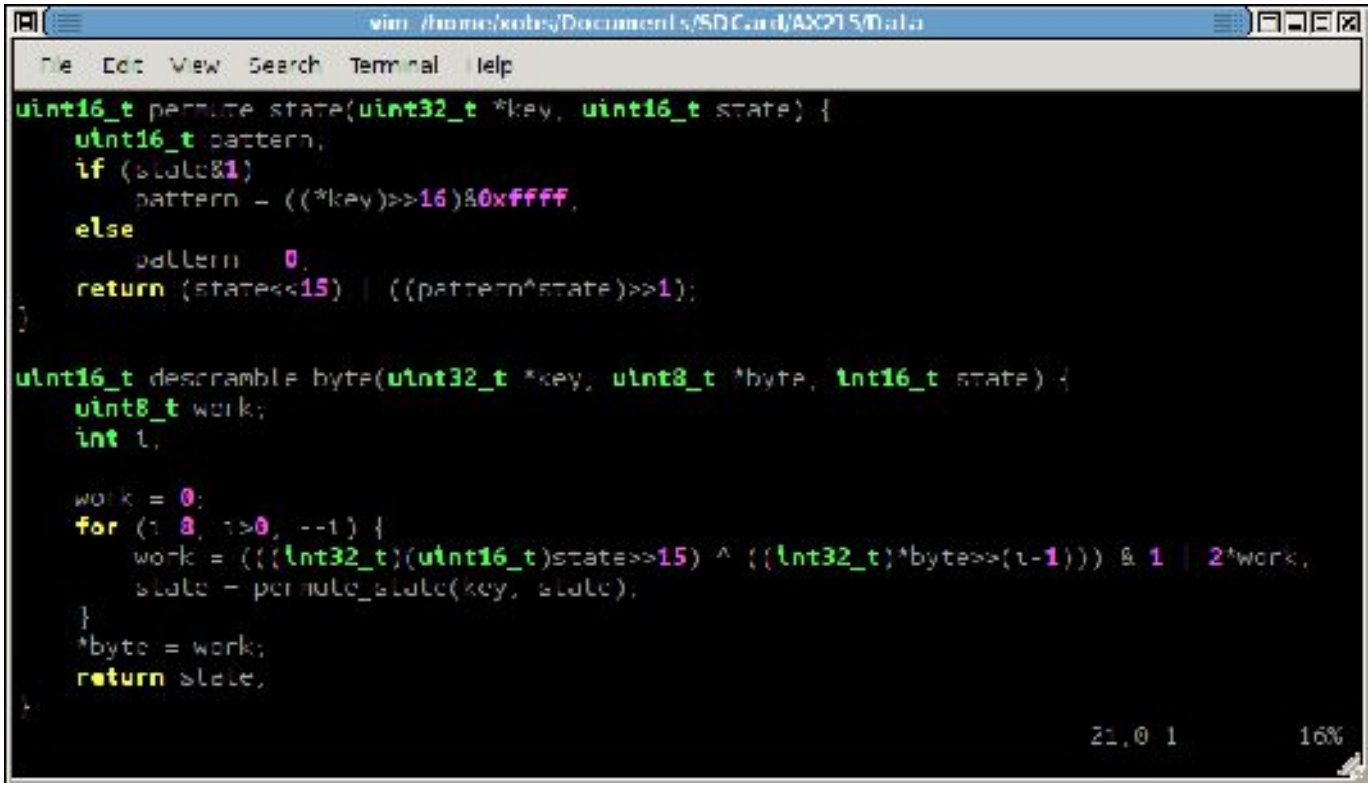
- Four 32-bit registers

```
00000080  10 80 31 47 00 00 00 00  00 00 00 00 00 00 c6 80  |..1G............|
00000090  10 00 38 ff 00 00 00 b5  0f 00 00 fd 74 2f f5 00  |..8.........t/..|
000000a0  78 83 00 00 05 00 00 00  ff 00 00 00 00 00 00 00  |x...............|
000000b0  73 2e ff 00 00 00 00 d7  80 00 03 00 ff 01 1d 00  |s...............|
000000c0  ed 04 00 00 60 ff 3f 02  52 05 00 00 00 13 00 00  |....`.?.R.......|
000000d0  40 00 50 41 00 00 00 00  00 00 00 00 80 ff 00 00  |@.PA............|
000000e0  12 00 00 02 00 10 06 f0  00 00 00 00 00 ff ff 00  |................|
000000f0  00 00 20 ff ff 3f 00 1f  00 00 00 00 ff ff 00 ff  |....?...........|
```

- “extop” debugger command
- Discovered 32-bit clr, not, inc, dec
- Many undiscovered opcodes

AX215

- Similar to AX211
- Faster, more GPIOs, different SFR map



Time for Tin Foil Hats

- Attack scenarios:
  - Eavesdropping
    - Report smaller than actual capacity
    - Data is sequestered to hidden sectors that are uneraseable
  - ToC/ToU
    - Present one version of file for verification, another for execution
    - Bootloader manipulation, etc.
  - Selective-modify
    - Scan for assets of interest, e.g. security keys, binaries, and replace with insecure versions

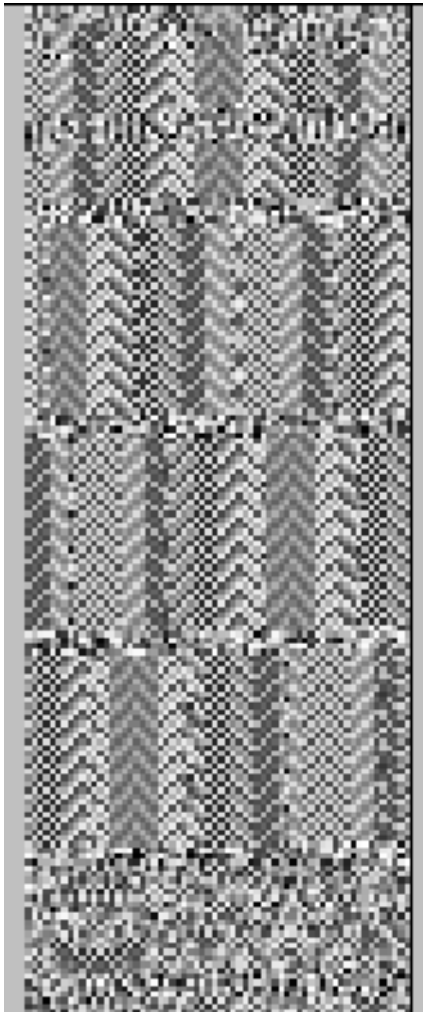
Other Direction: Samsung MMC

- Samsung pushed firmware patch to eMMC cards in Android
- Contains ARM7 code
  - http://forum.xda-developers.com/showthread.php?t=2096045
  - Uses “class 8” instructions reserved for manufacturer

“By inspecting some code, it seems that we know how to dump the eMMC RAM:
Look at the function mmc_set_wearlevel_page in line 206. It patches the RAM (using the method mentioned before), then it validates what it has written (in lines 255-290). Seems that the procedure to read the RAM is as following:
1. CMD62(0xEFAC62EC) CMD62(0x10210002) to enter RAM reading mode
2. MMC_ERASE_GROUP_START(Address to read) MMC_ERASE_GROUP_END(Length to read) MMC_ERASE(0)
3. MMC_READ_SINGLE_BLOCK to read the data
4. CMD62(0xEFAC62EC) CMD62(0xDECCEE) to exit RAM reading mode ”

Other Direction: TLC

- TLC Flash has scrambling applied to avoid “read-disturb” and “program-disturb” issues
  - Scrambling is a proprietary algorithm, as of yet unknown
  - Highly structured



Wrap-up

- SD cards contain fully programmable microcontrollers
- Controller program modifiable via special host commands
  - Potential for MITM attack scenarios ☻
  - Potential for extremely cheap microcontroller for fun projects ☺

Special Thanks

- Shout out to .mudge for creating CFT which enabled this research, and many other good things (some yet to come!)

Q&A

- Demo (time allowing)
- Thanks for your attention!

About the 8051