Performing Open Heart Surgery on a Furby

Michael Coppola

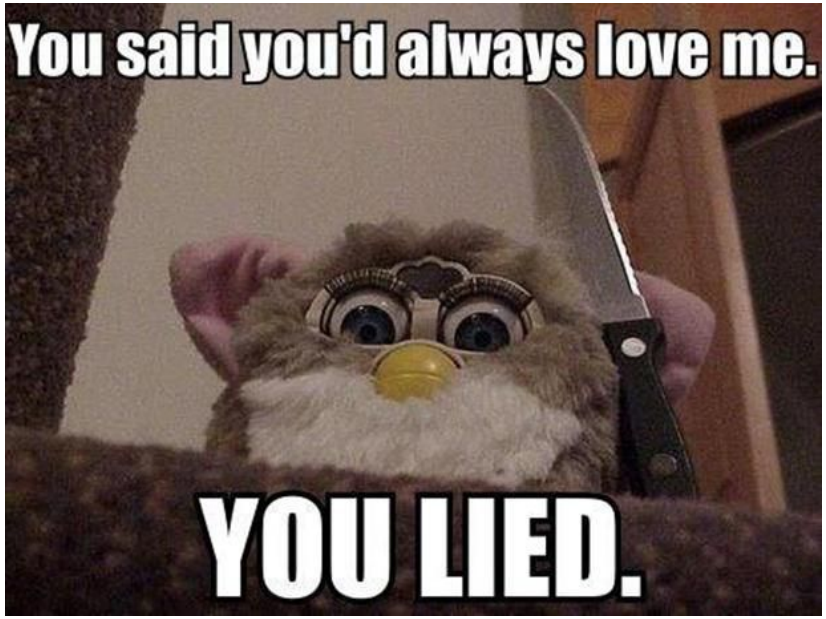
Recon 2014

Who am I?

- #
- Student at Northeastern University
- CTF every now and then
- http://poppopret.org/

So.. What is this thing?

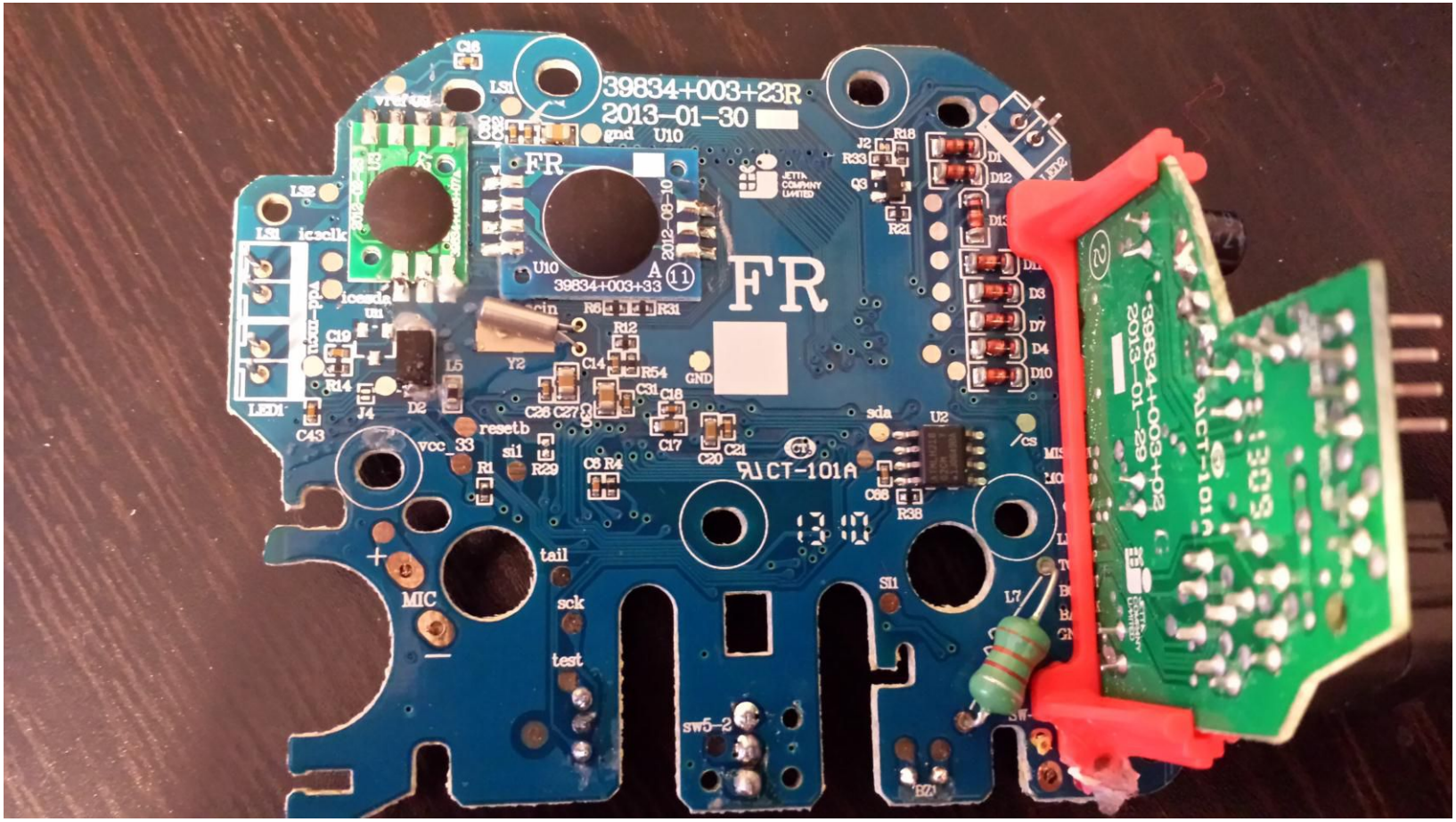
- Furby 2012
- Animatronic toy made by Hasbro (originally Tiger)
- Responds to stimuli
- Speaks “Furbish”, but learns English over time
- Interacts with other nearby Furbies



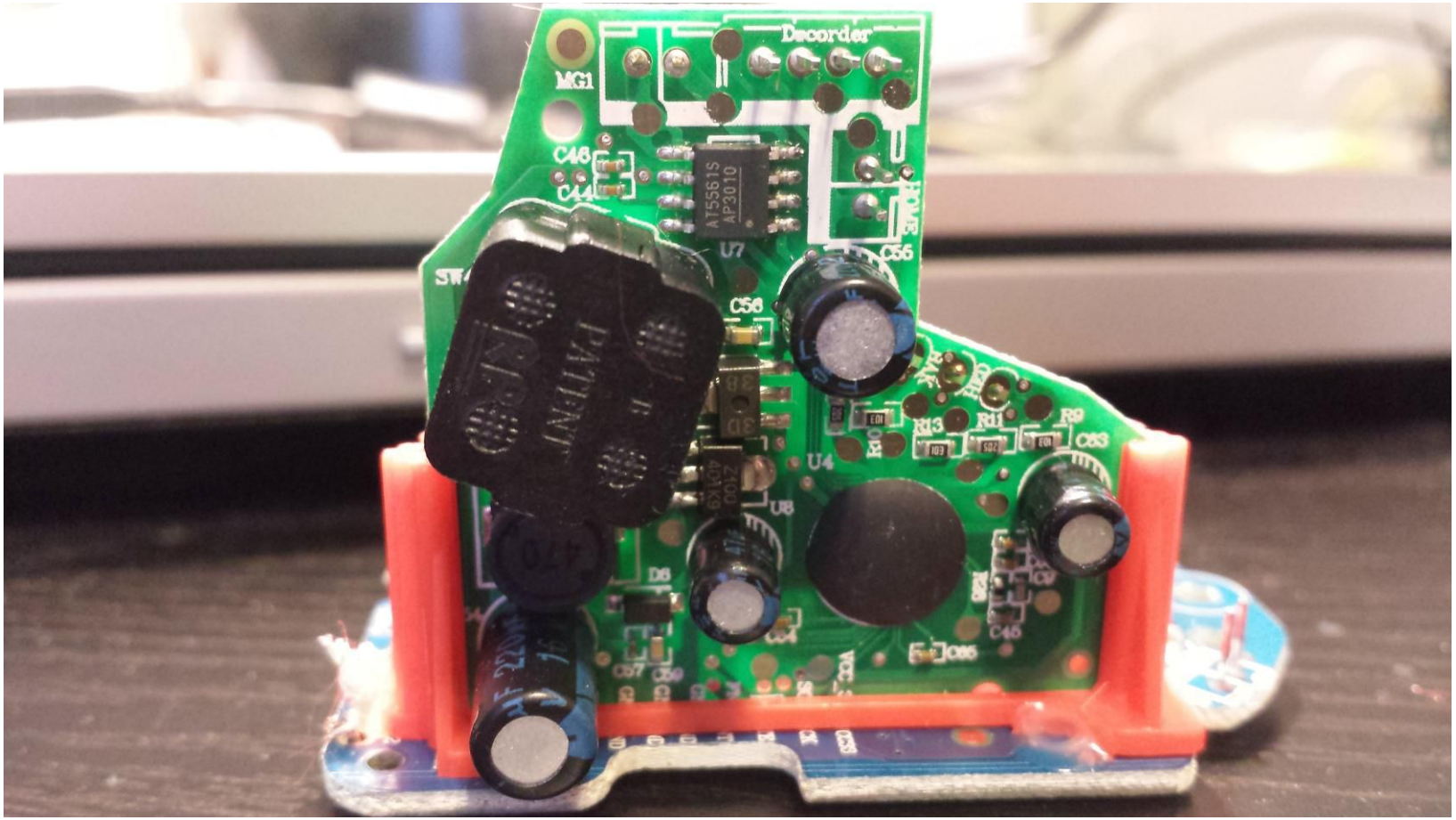
This thing communicates?

- Originally over IR, now over a #badBIOS-esque protocol
- Pulses a high-pitched tone and decodes through the microphone
- github.com/iafan/Hacksby

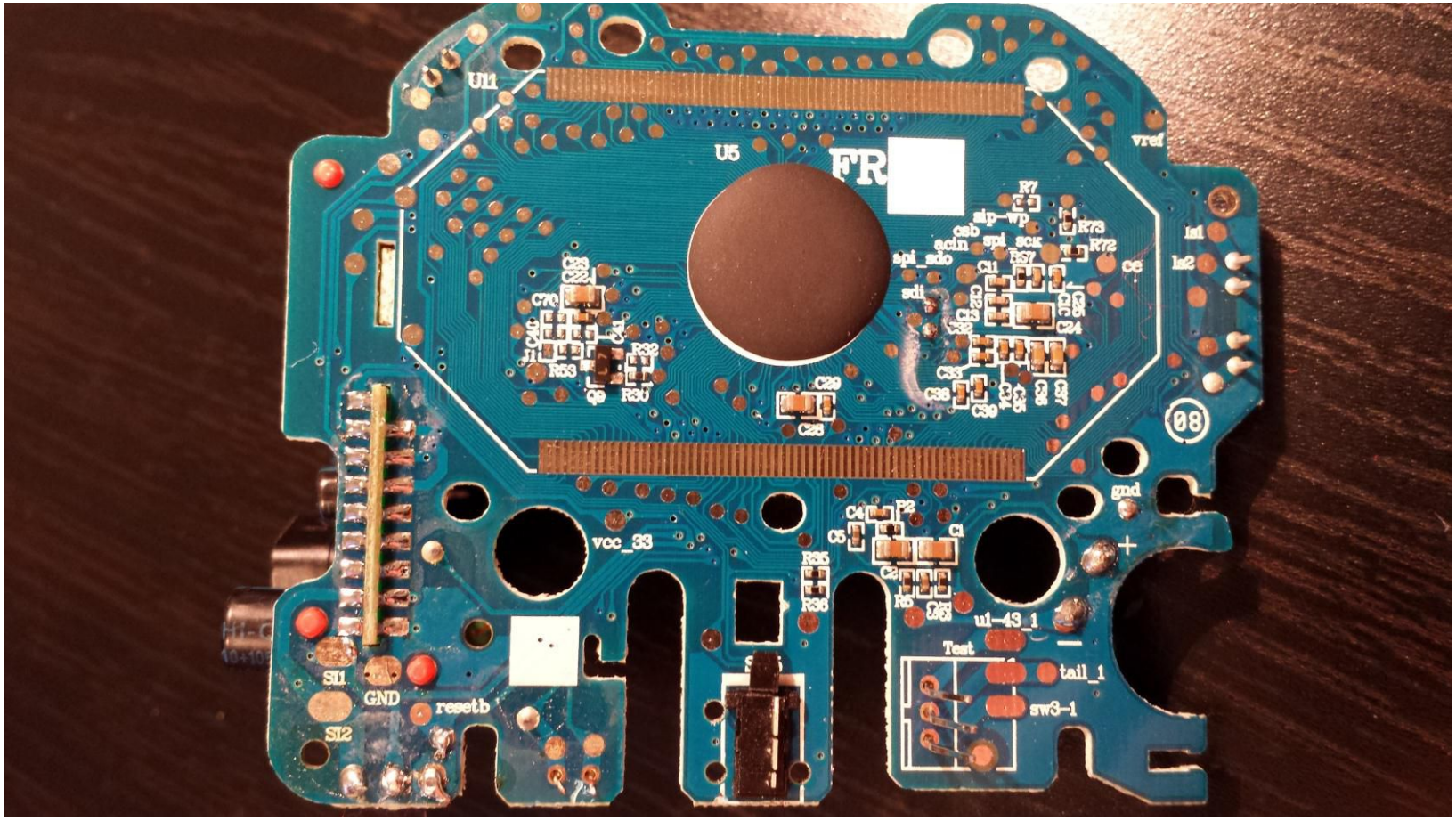
The circuit board



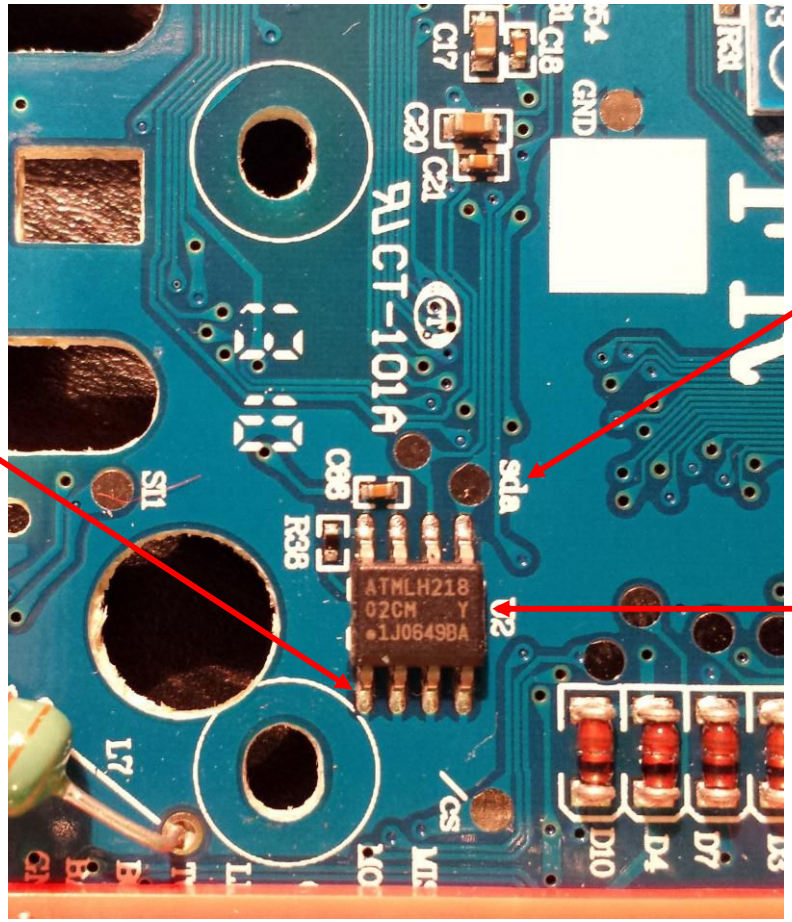
The circuit board



The circuit board



Identifying components



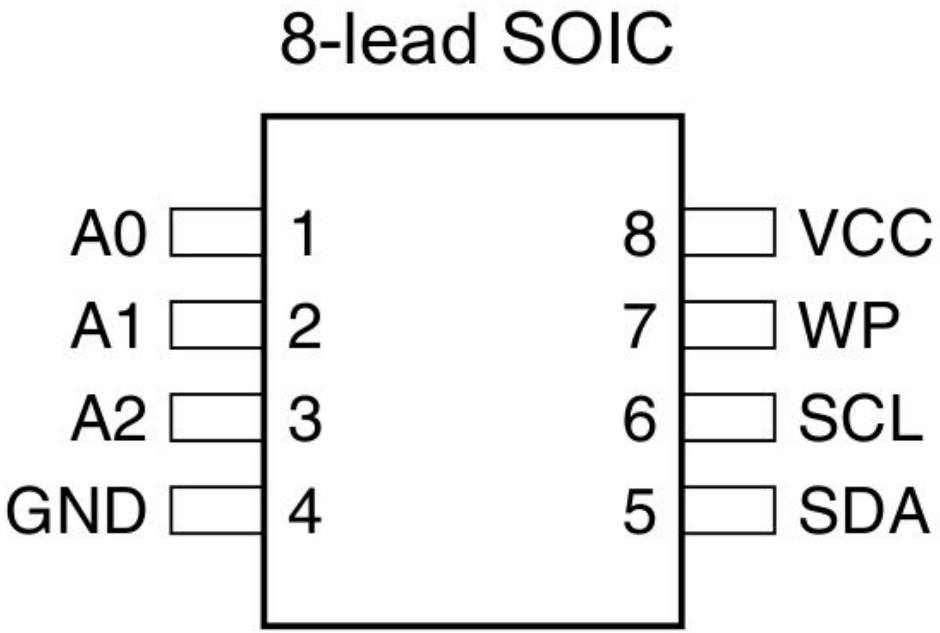
Yup, it's EEPROM

Desoldering components

- Heat gun + tweezers
- Cheap rework station
  - Sparkfun $100
- Solder wick
- Soldering iron blade tip

Interfacing with EEPROM

- I2C protocol
- A0-2 address pins
- WP – write protect
- SCL – clock
- SDA – data

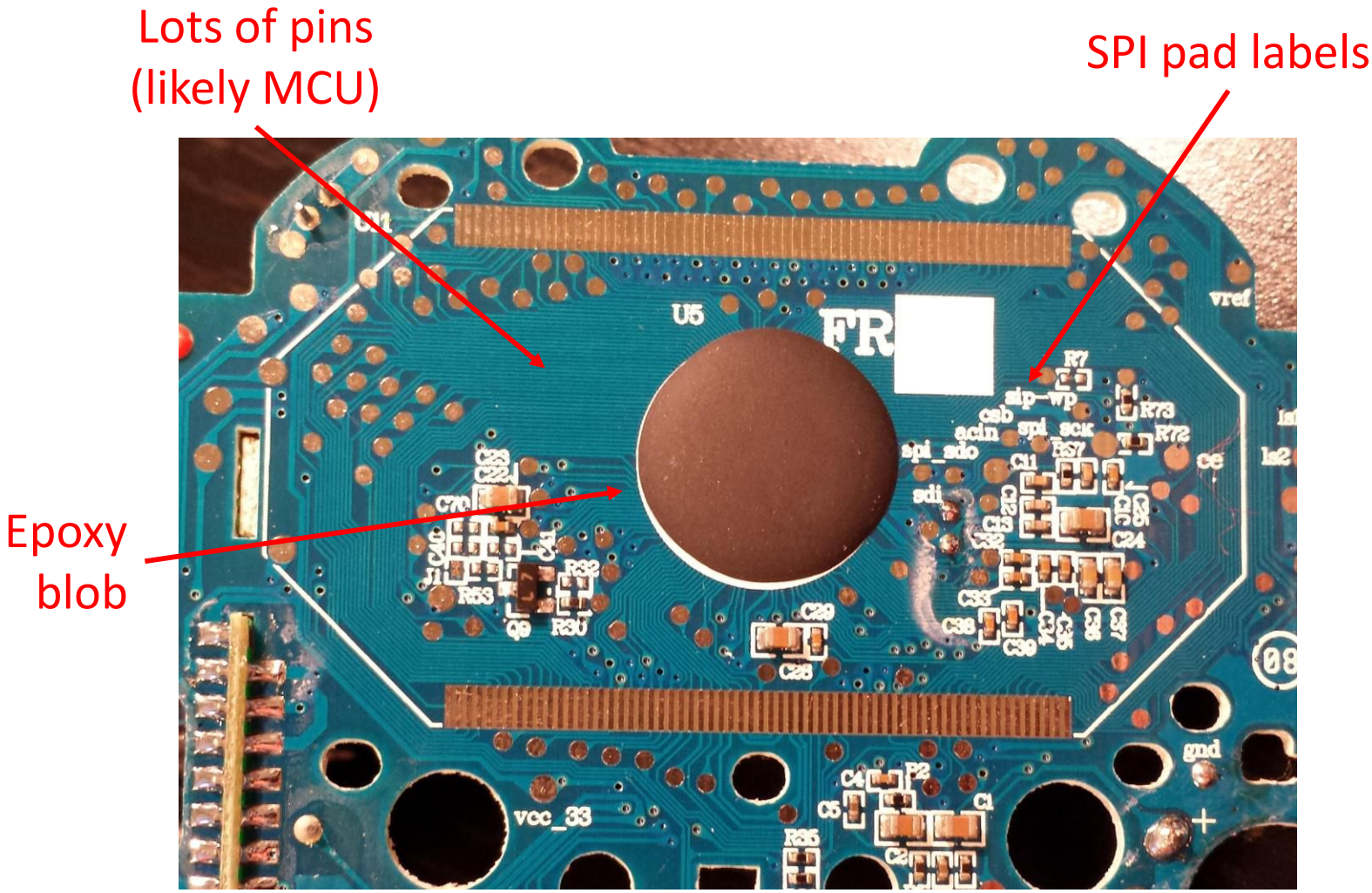


Dumped EEPROM

```
2F 64 00 00 00 00 5A EB 2F 64 00 00 00 00 5A EB
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
05 00 00 04 00 00 02 18 05 00 00 04 00 00 02 18
0F 00 00 00 00 00 18 18 0F 00 00 00 00 00 18 18
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F8
```

(Likely runtime settings of some sort)

Chip-on-board is annoying

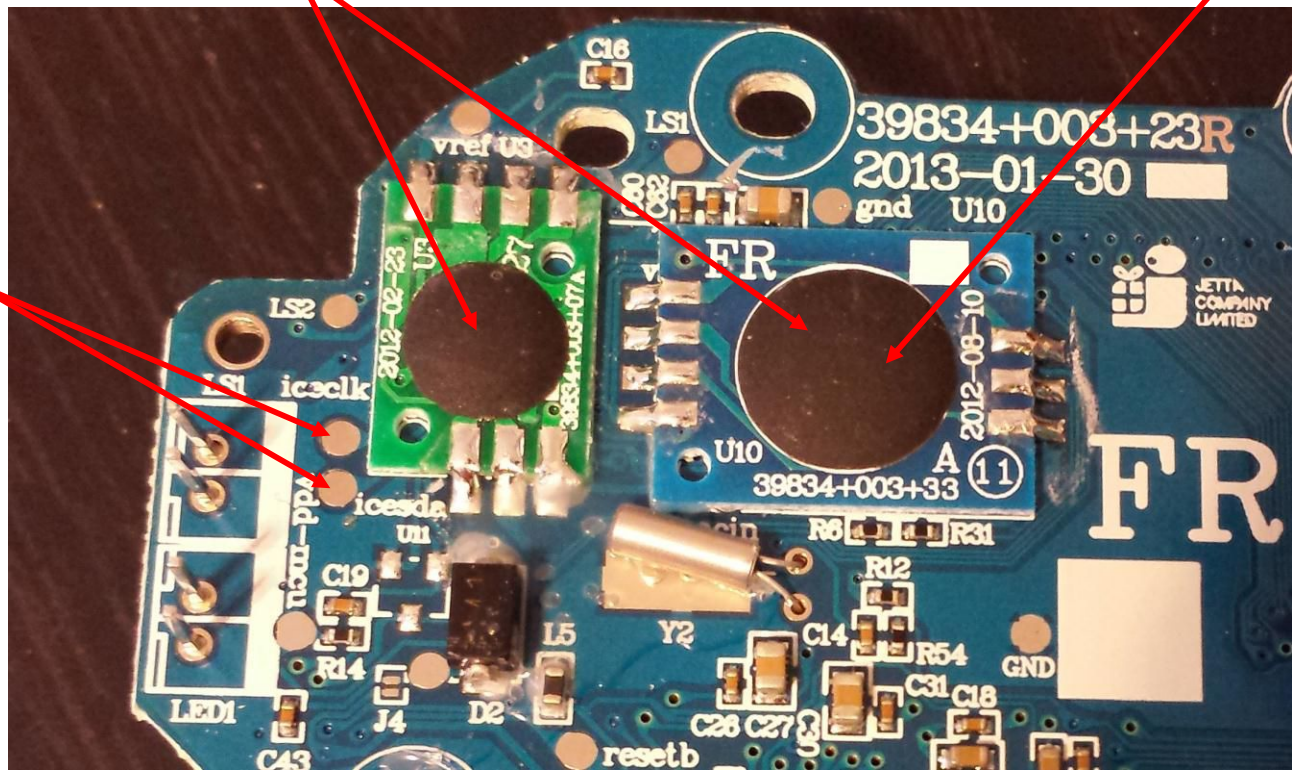


Chip-on-board is annoying

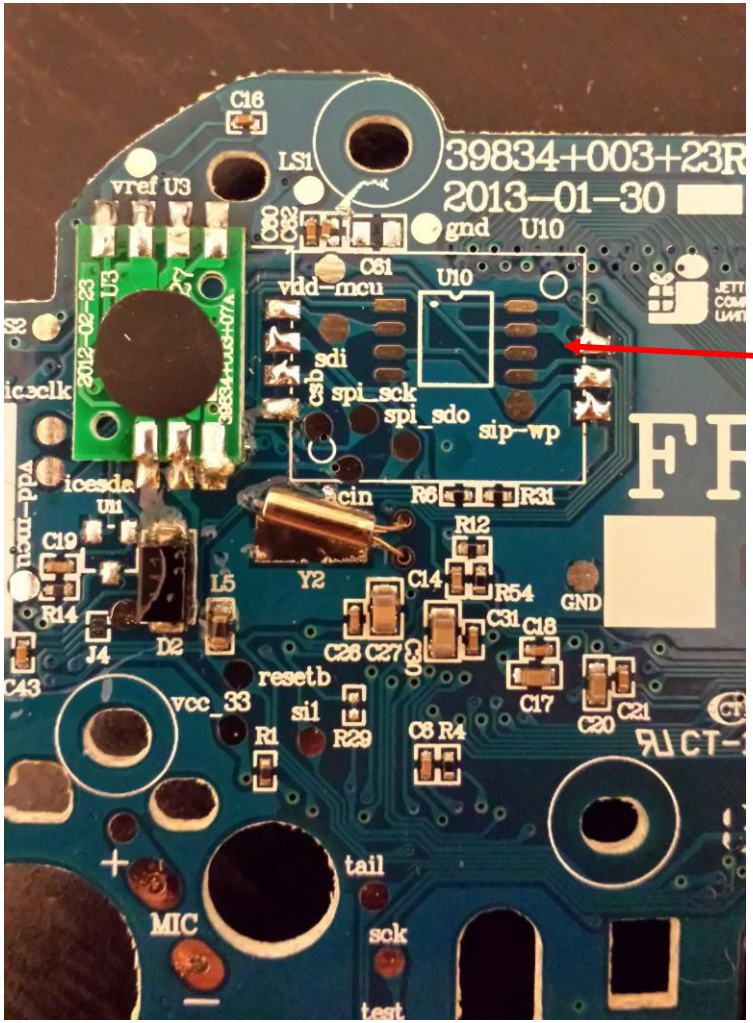
Possibly connected to SPI vias

More epoxy bullshit

ICECLK?
ICESDA?



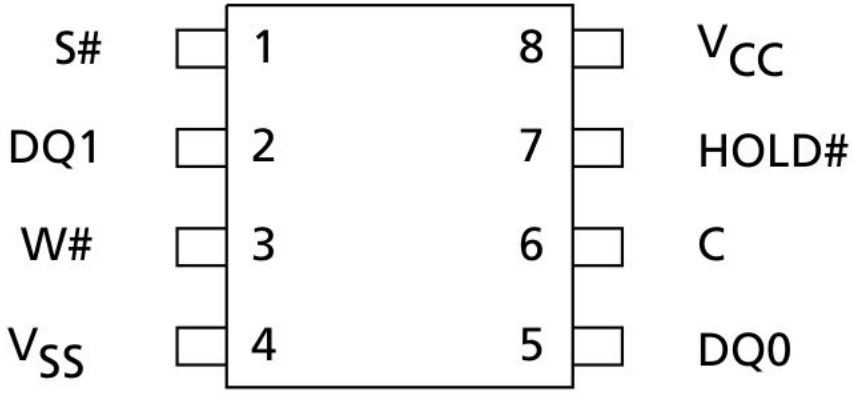
…That's convenient



Full pinout
(SPI memory)

Interfacing with SPI component

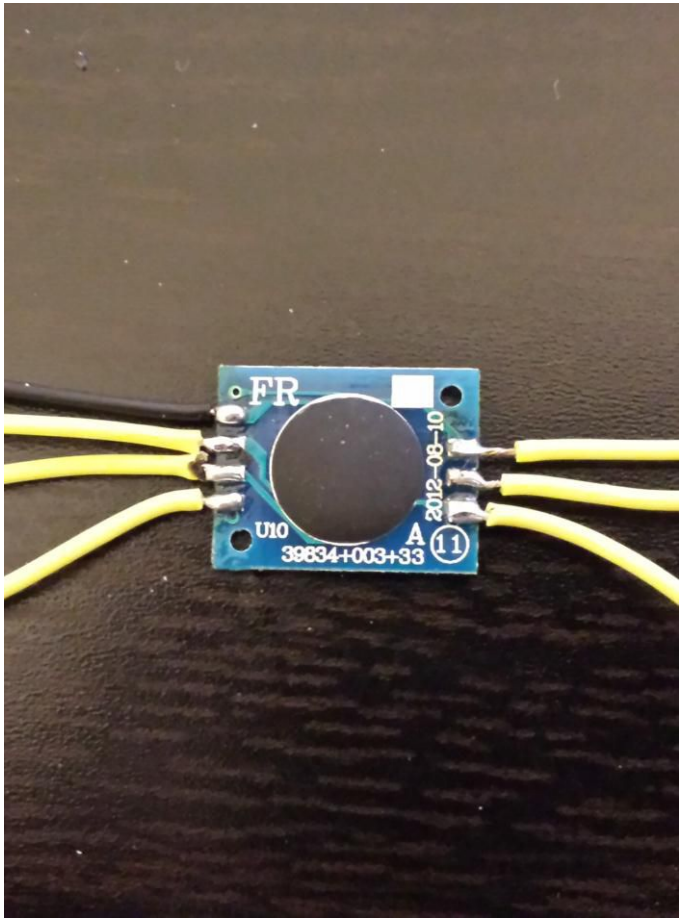
- Shift registers – exchange bytes
- MISO – Master In Slave Out
- MOSI – Master Out Slave In
- CS – Chip select
- CLK - Clock
- WP# - Write protect (inv)
- HOLD# - Hold (inv)



Interfacing with SPI component

- Arduino is too slow for SPI
- Bus Pirate?
  - Adafruit $37

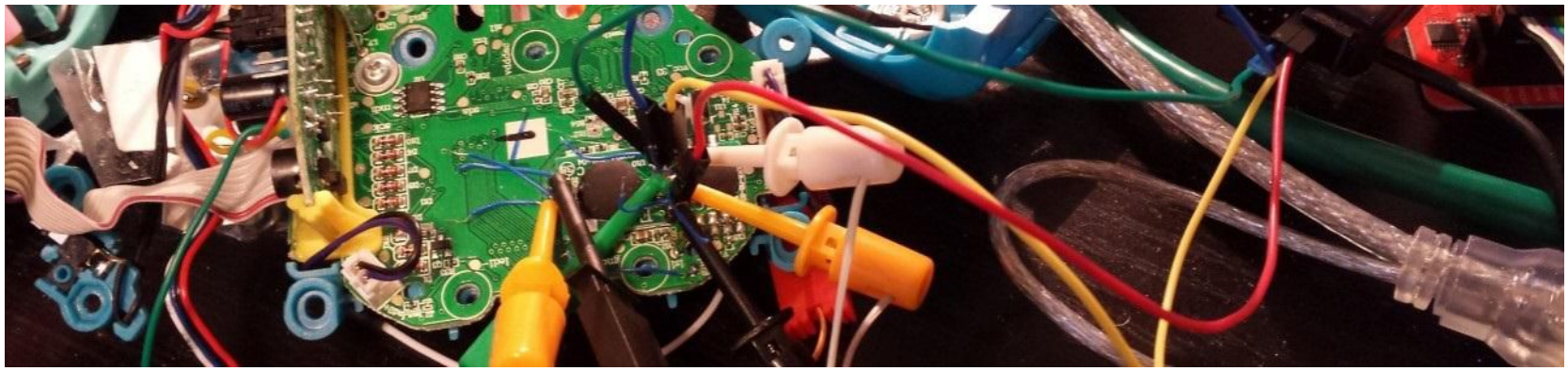
- Chip not recognized by flashrom
- But spitool seemed to return some kind of data



Dumping with spitool

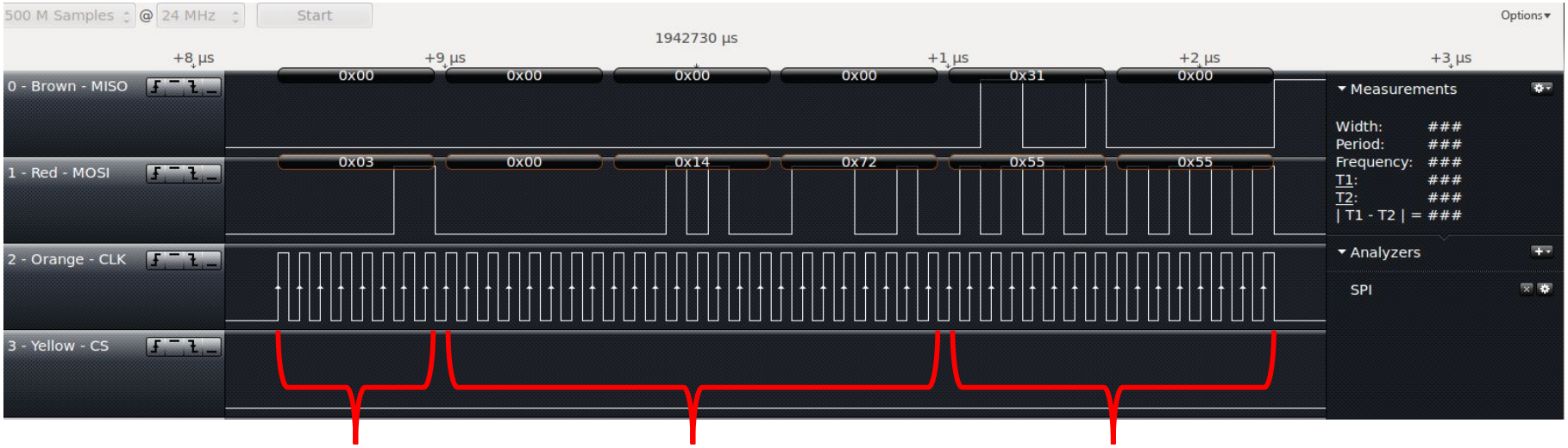
- Returned valid looking data but… it would repeat every 0x4000 bytes
- Bought a knockoff Saleae logic analyzer to verify the read process ($10)

- Probes on MISO, MOSI, CLK, and CS



Debugging with a logic analyzer

Sample capture from boot:



0x03 READ

3-byte address

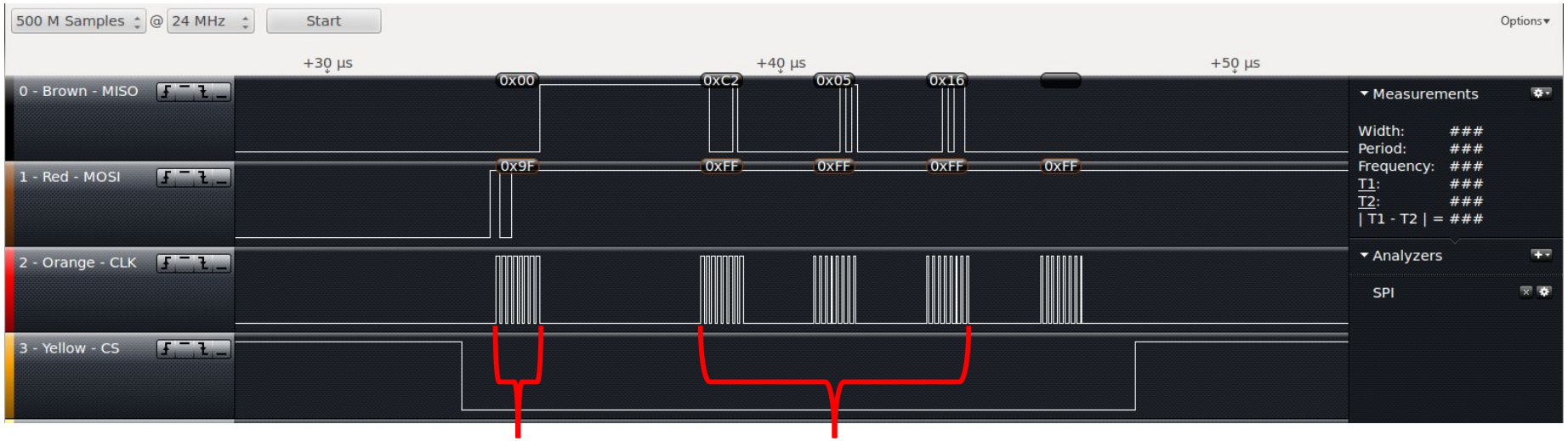
Retrieved data

Debugging spitool

- spitool sent well-formed SPI commands… just the wrong ones
- Incremented through the entire 24-bit address space and wrapped around multiple times
- Back to trying flashrom

Dumping with flashrom

- flashrom couldn’t recognize the chip, but maybe it just doesn’t support it yet
- Sniffed the flashrom PROBE operation:



0x9F RDID 3-byte JEDEC ID

Identifying the SPI component

- JEDEC ID: 0xC2 0x05 0x16

MXIC

MACRONIX
INTERNATIONAL CO., LTD.

MX23L3254

COMMAND DESCRIPTION

(1) Read Identification (RDID)

The RDID instruction is for reading the manufacturer ID of 1-byte and is followed by Device ID of 2-byte. The MXIC Manufacturer ID is C2h, the memory type ID is 05h as the first-byte device ID, and the individual device ID of second-byte ID is:16h.

The sequence of issuing RDID instruction is: CS# goes low-> sending RDID instruction code -> 24-bits ID data is sent out on SO -> to end RDID operation which can use CS# to be high at any time during data out. (see Figure 3) When CS# goes high, the device is at standby stage.

Table of ID Definitions:

RDID	manufacturer ID	memory type	memory density
9Fh	C2h	05h	16h

Identifying the SPI component

- Chip is a Macronix MX23L3254
- 4MB (32Mbit)
- Mask ROM (read only)
- 16 pins, but 8 are disconnected internally

Dumping with flashrom

- Wrote a new config, identifies chip, and dumps contents successfully

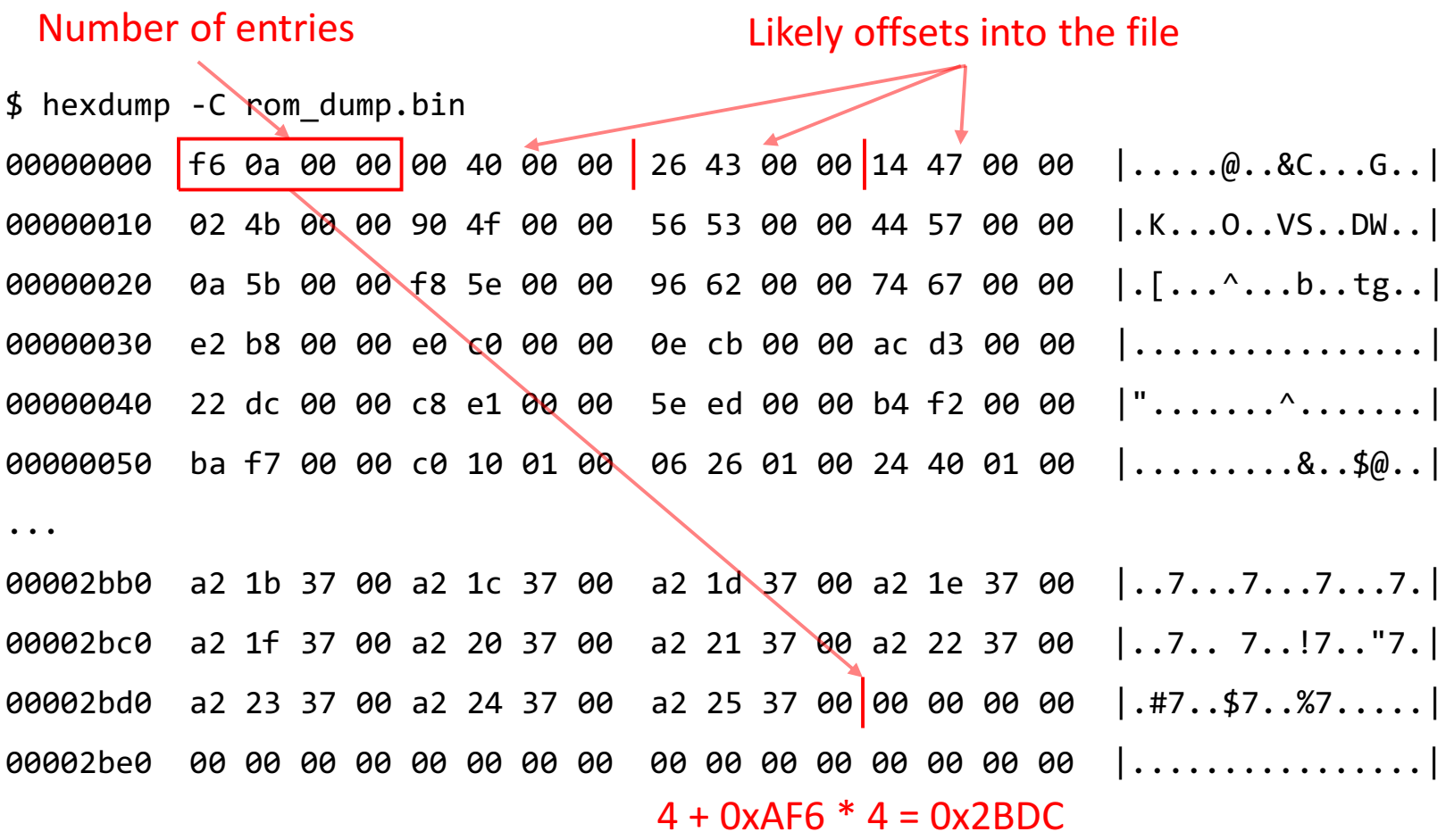
```
$ ./flashrom -p buspirate_spi:dev=/dev/ttyUSB0 -r out.bin
flashrom v0.9.7-r1767 on Linux 3.8.0-37-generic (x86_64)
flashrom is free software, get the source code at http://www.flashrom.org

Calibrating delay loop... OK.
Found Macronix flash chip "MX23L3254" (4096 kB, SPI) on buspirate_spi.
```

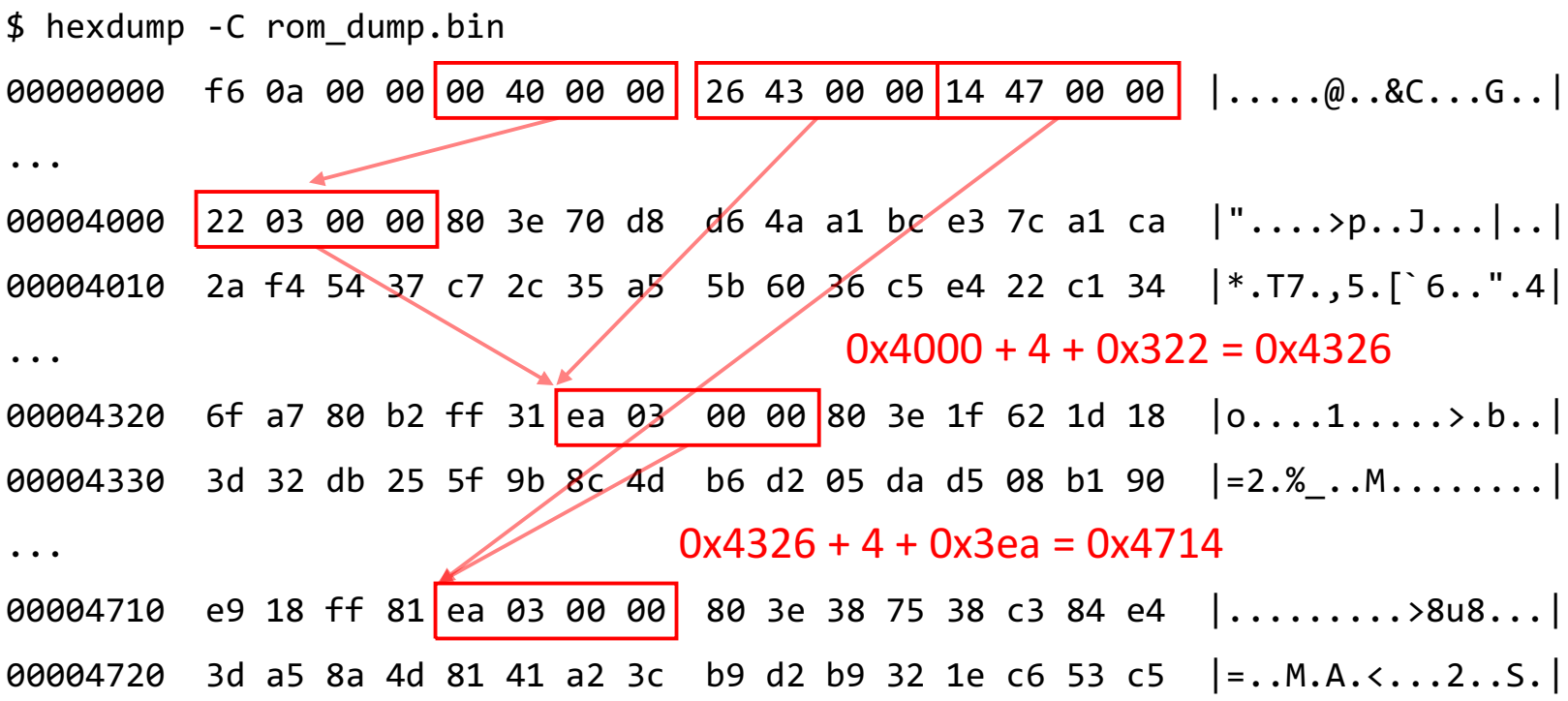
Analyzing the ROM

- 4MB binary image
- No results from binwalk
- No strings
- Two sections joined by null padding

Analyzing the ROM header



Analyzing the ROM body



ROM format

Header:

[number of offsets] [offset to record] …

Variable records:

[size of record] [record data] …

Constant records: 256 bytes

So what kind of data is it?

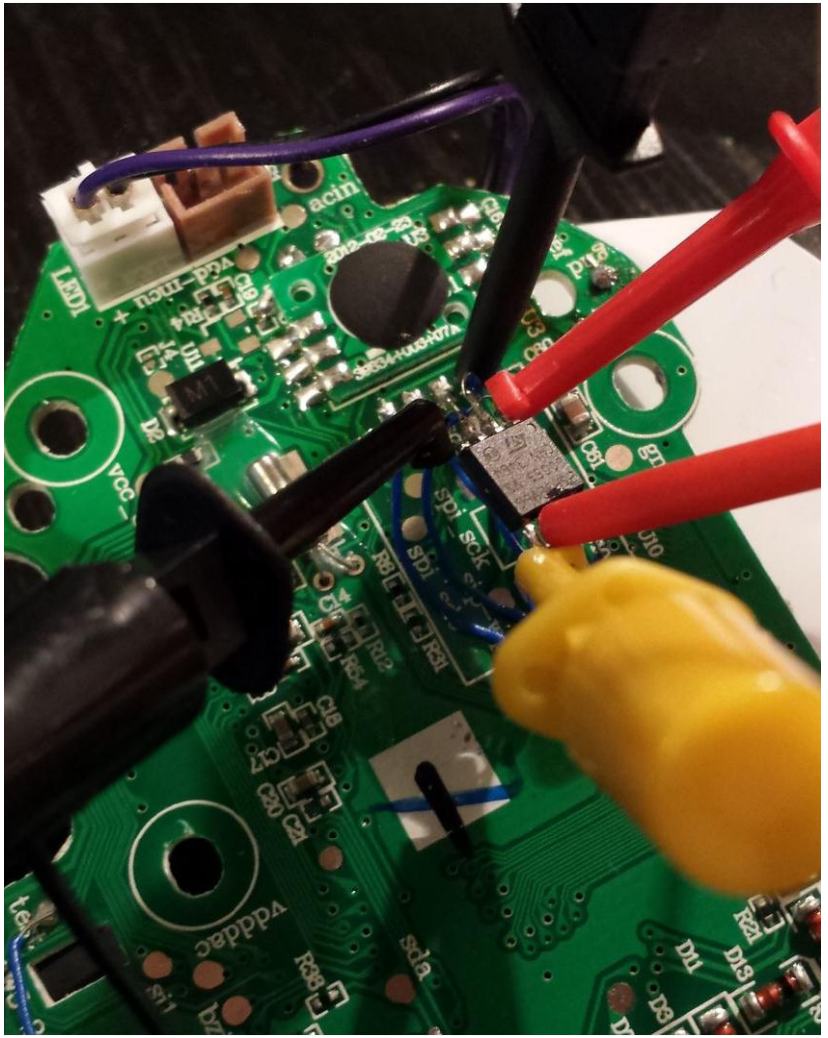
- Guesses:
  - Code? Probably not, weird format
  - Audio data? Maybe, the variable size records
  - Image data? Maybe, the consistent size records

- Manipulate data on the chip, see how system behavior changes

- Mask ROM is read-only, so we can’t reprogram it

Let’s fuzz a bit

- The COB mask ROM is... on a desolderable board
- Remove mask ROM, replace with similar read/write flash memory
- Program chip with fuzzed data, observe



Observing system behavior

- Clobber all records with ‘AAAAAAAAAAAAAAA’
  - No audio
  - LCD eyes are messed up
- Point all offsets in header to same record
  - Produces only one sound
  - LCD eyes are messed up
- Our guesses were correct

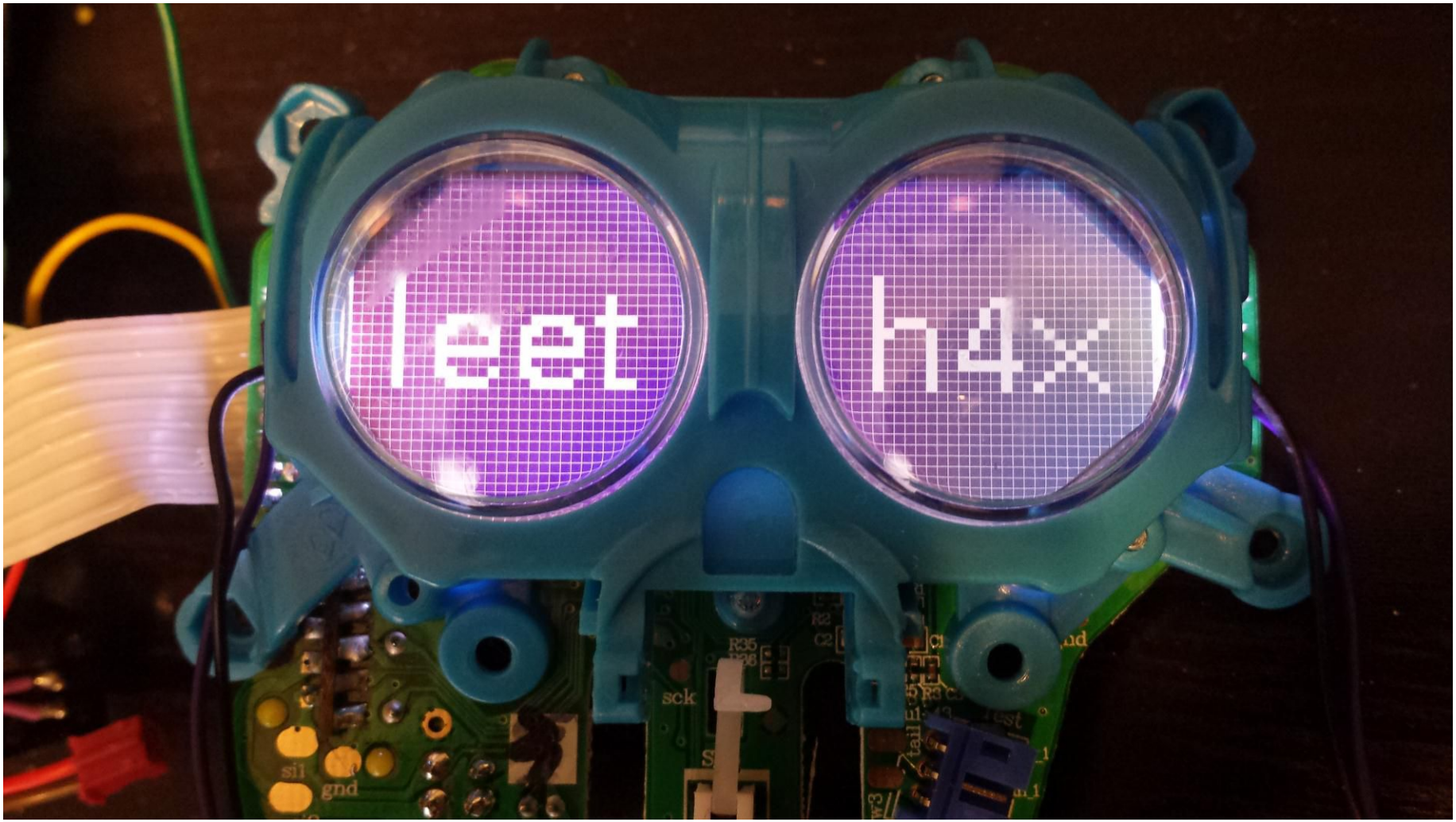
Let’s start with image data

- Each record is 256 bytes
- LCD is 64x32 pixels = 256 * 8
  - 1 pixel = 1 bit
- Need to find mapping between data ↔ LCD pixels

Let’s start with image data

- Flashed unique patterns and recorded pixel locations, but took way too long
- Got help from Olivier Galibert (a MAME dev), derived x-y offsets

Arbitrary control over the LCD



What about the audio data?

- Can we craft arbitrary audio too?
- Tried (mostly) every format/codec could think of
- No idea what it is
- Common first two bytes: 0x80 0x3e
- Some code / more info would be nice

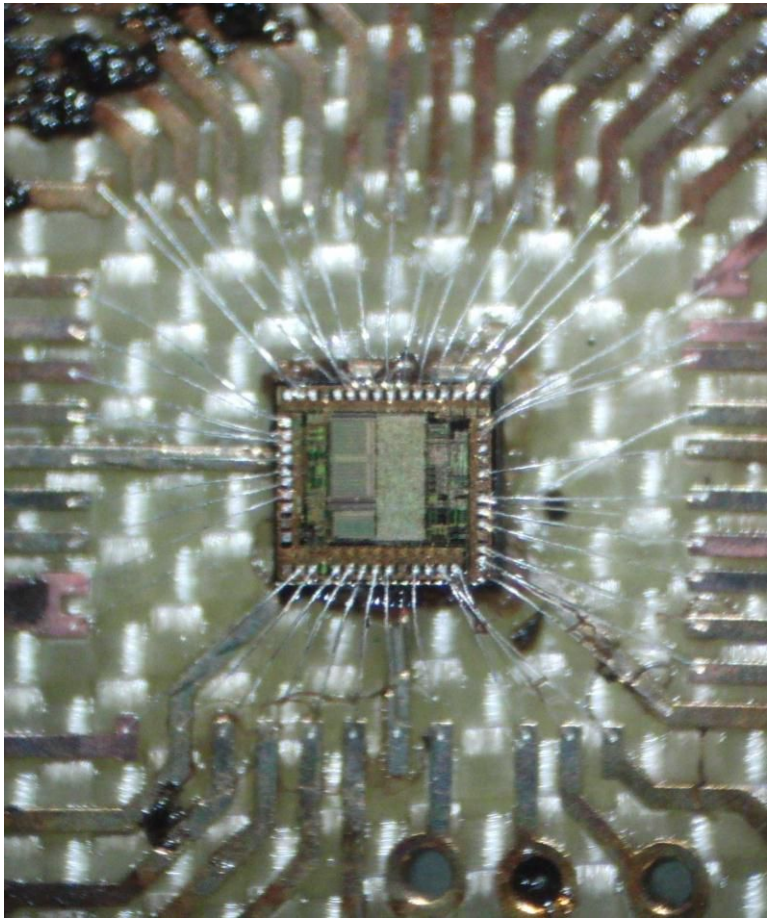
Microcontroller?

- No idea what it is, or which architecture
- Possible to read code off it?
- Traced pads to/from
- No JTAG, but seriously… WTF is ICE?
  - Google mentions something about “Generalplus”
- Enough with the guessing…

BOIL EVERYTHING IN ACID

Chip decapsulation

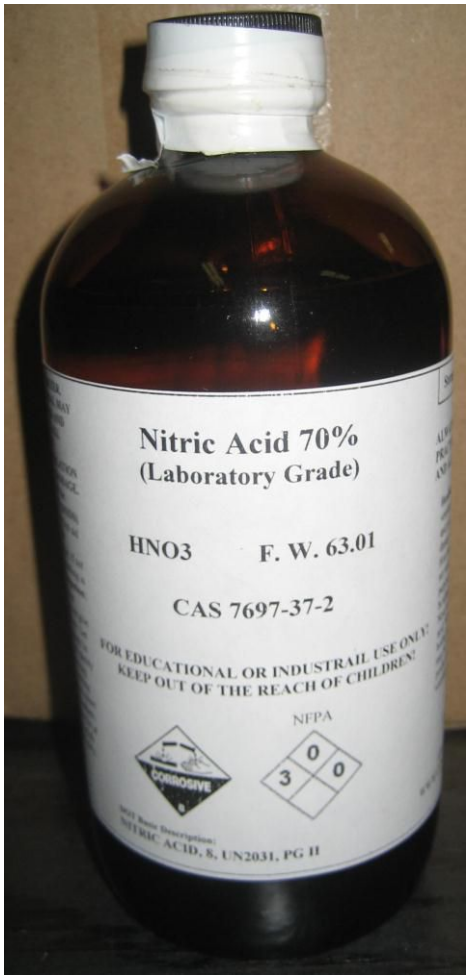
- (aka chip “decapping”)
- Exposes die for analysis
- Many creative techniques
  - Mechanical
  - Thermal
  - Chemical
- Live analysis possible



siliconpr0n.org

Nitric acid

- HNO_3
- Concentrated (68%)
  - Requires high temp
  - Degrades bond pads
- Fuming (>86%)
  - Reacts at room temp
  - Permits live decap
- Really nasty stuff



siliconpr0n.org

Nitric acid

- Requires a **fume hood**
  - *Cu (s) + 4 HNO_3 (aq) = $Cu(NO_3)_2$ (aq) + 2 H_2O (l) +* ***2 NO_2 (g)***
  - a.k.a. you're going to be an unhappy camper
- Requires **proper disposal**
- Reasonable to obtain concentrated acid
- Nobody's going to sell you fuming acid
- You'll probably be put on a watch list

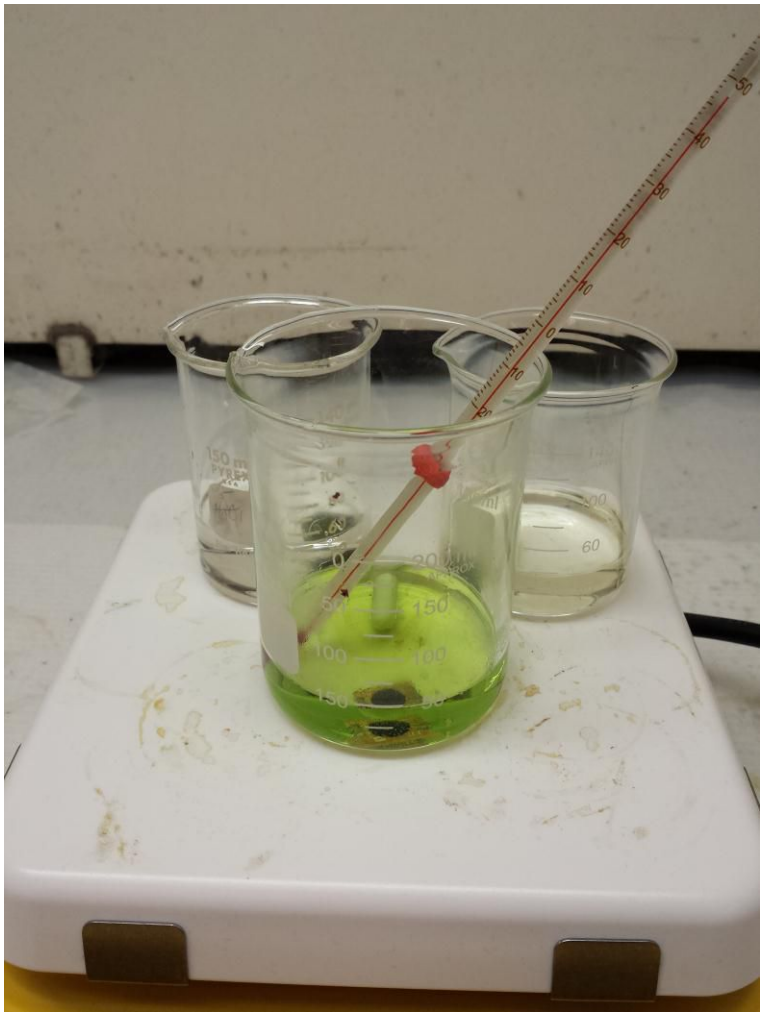
Sulfuric acid

- H_2SO_4
- Commercial drain cleaner
- Produces black sludge
- Leaves bond wires intact
- Also really nasty stuff

siliconpr0n.org

Decapping with nitric acid

- Isolate samples as much as possible
- 70% nitric acid
- Heat to 80°C
- 5 – 60 minutes



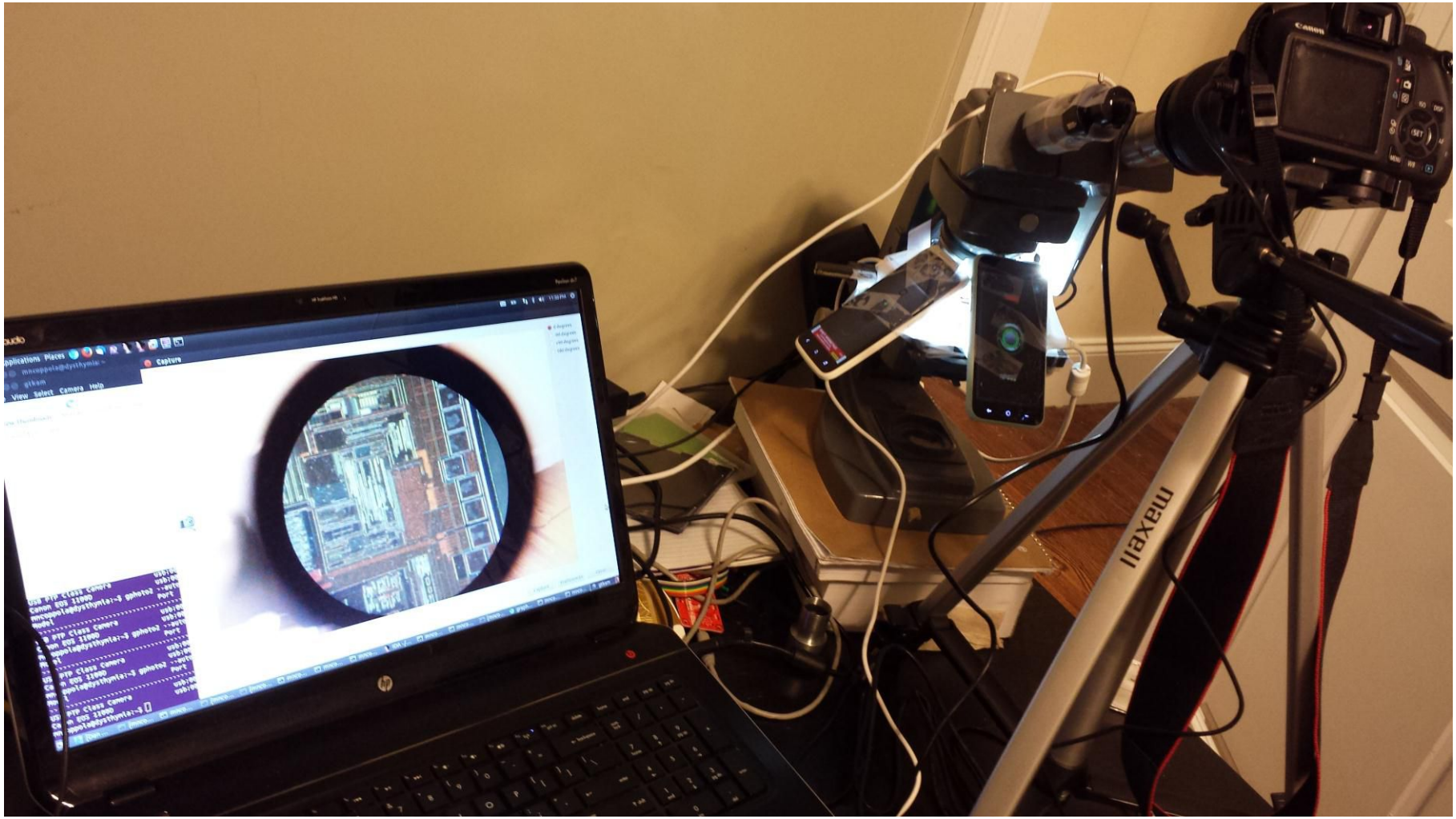
Recovering samples

- Decant + soft tweezers
- Rinse with deionized water, then acetone
- No, not nail polish remover
- Ethanol also works

Optical microscope

- Regular bio microscopes won't work
  - Need illumination from above
- Stereo / inverted / metallurgical microscope
  - Olympus BH(2) series highly recommended
- Likely able to see lower metal layers
- Image quality highly dependent on camera and objectives

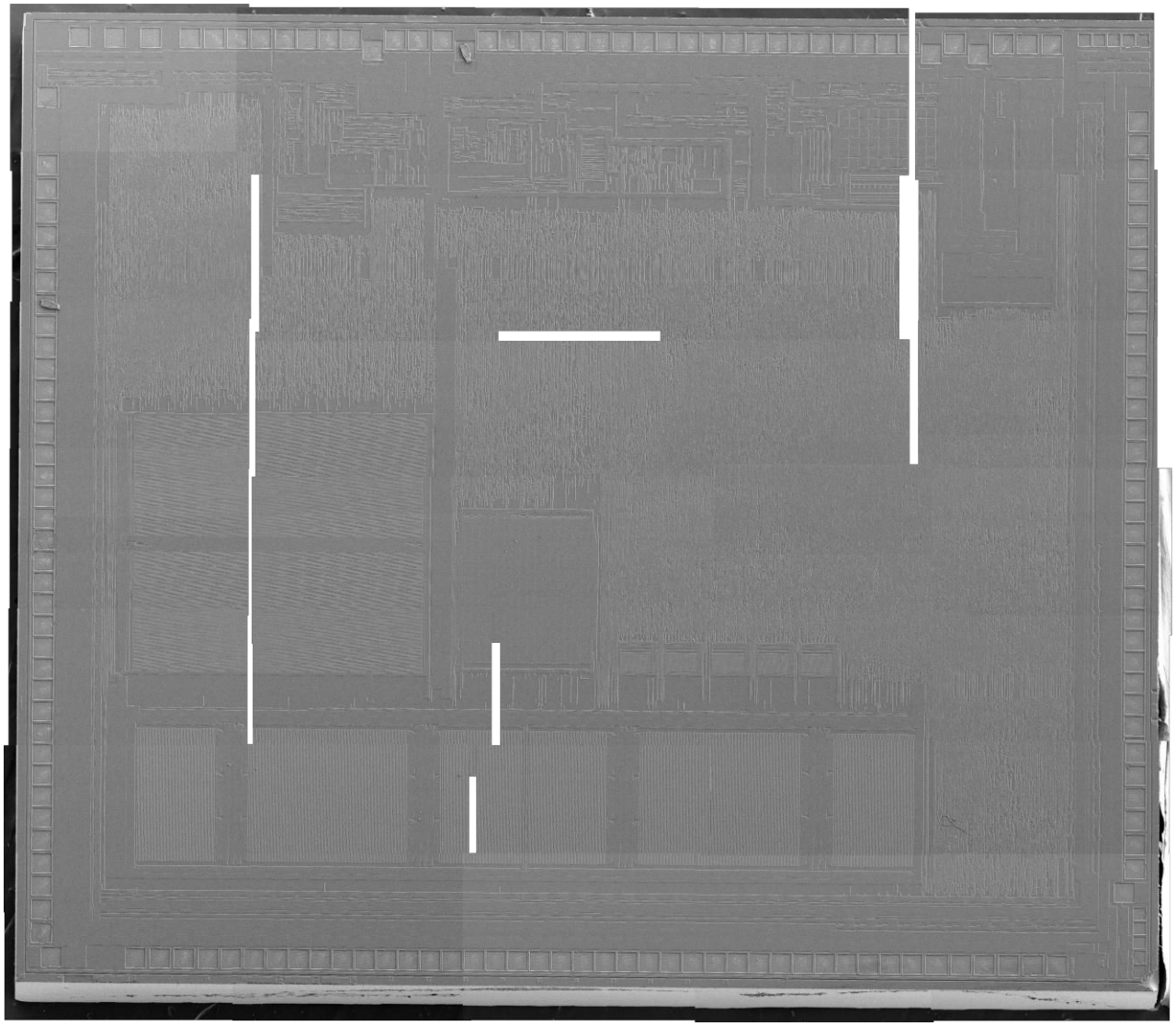
Work with what you've got



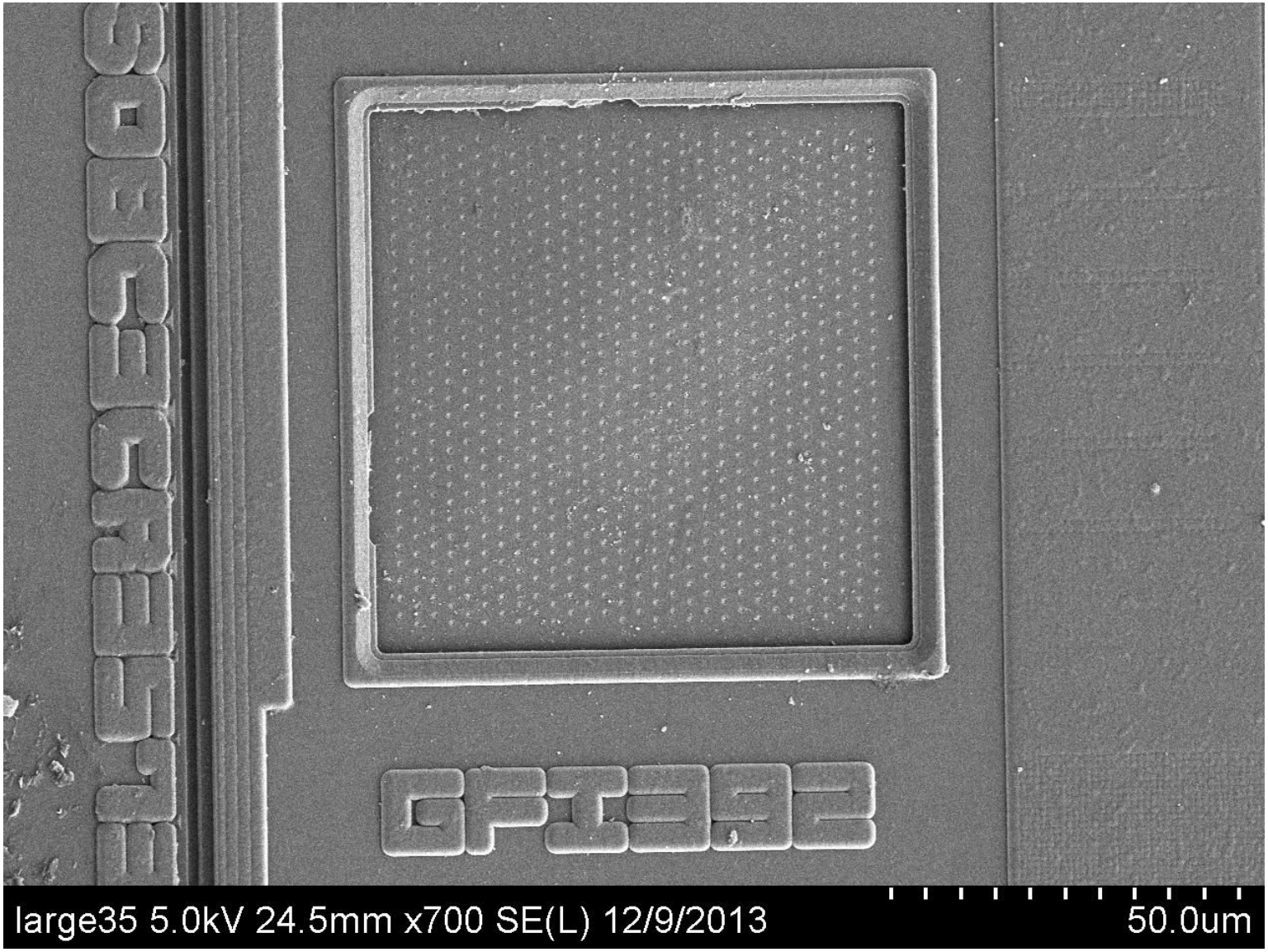
Scanning Electron Microscope

- Provides the highest resolution image at insane zoom levels
- Black & white image only
- Big problem: can only view topography of passivation layer (overglass)

Scanning Electron Microscope

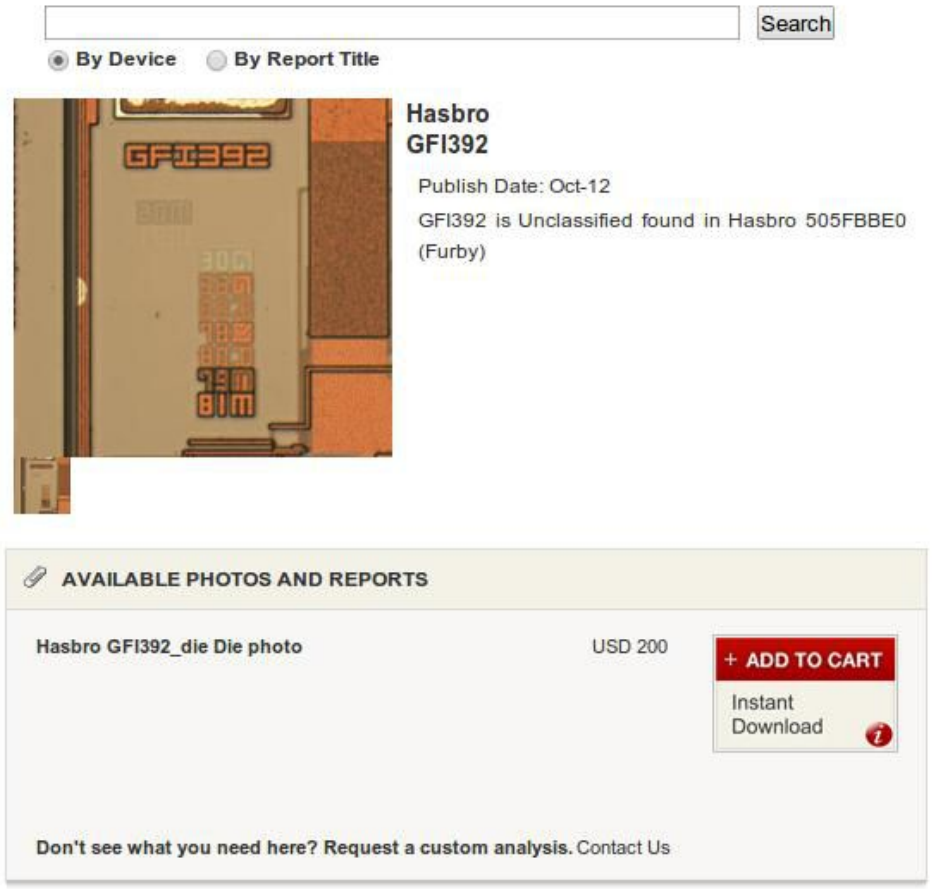


Scanning Electron Microscope



GFI392

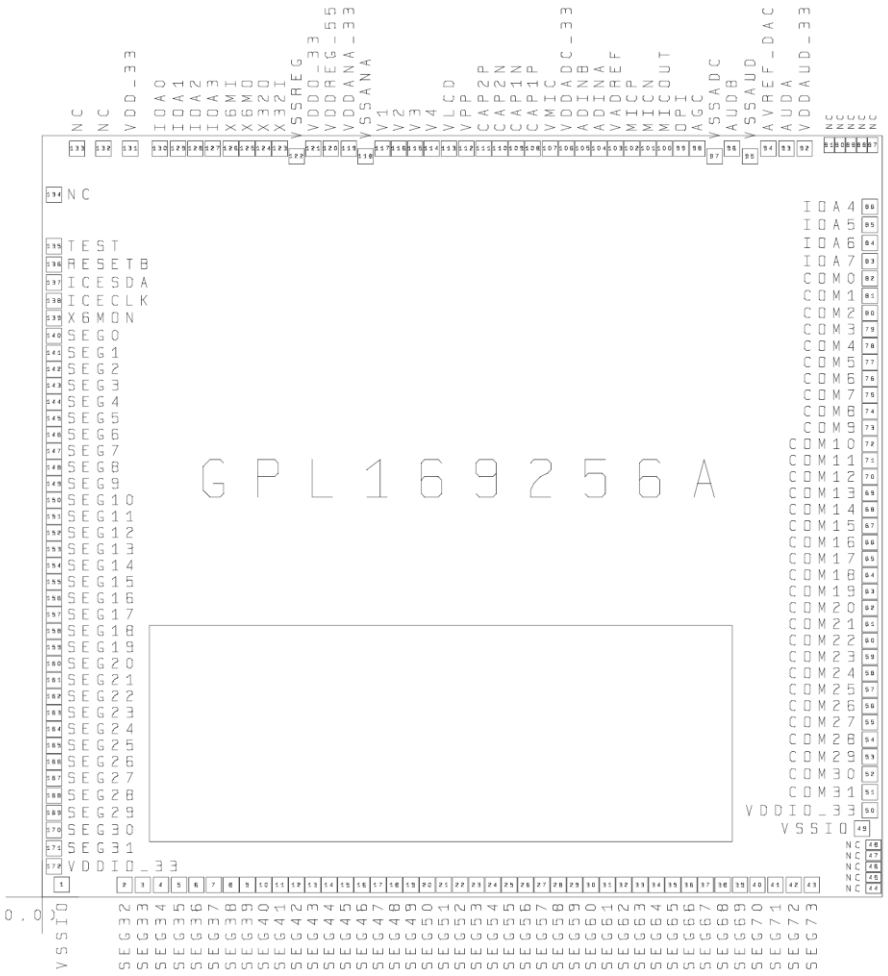
- No info on Google
- Might be rebranded
- Chipworks decapped this chip as well

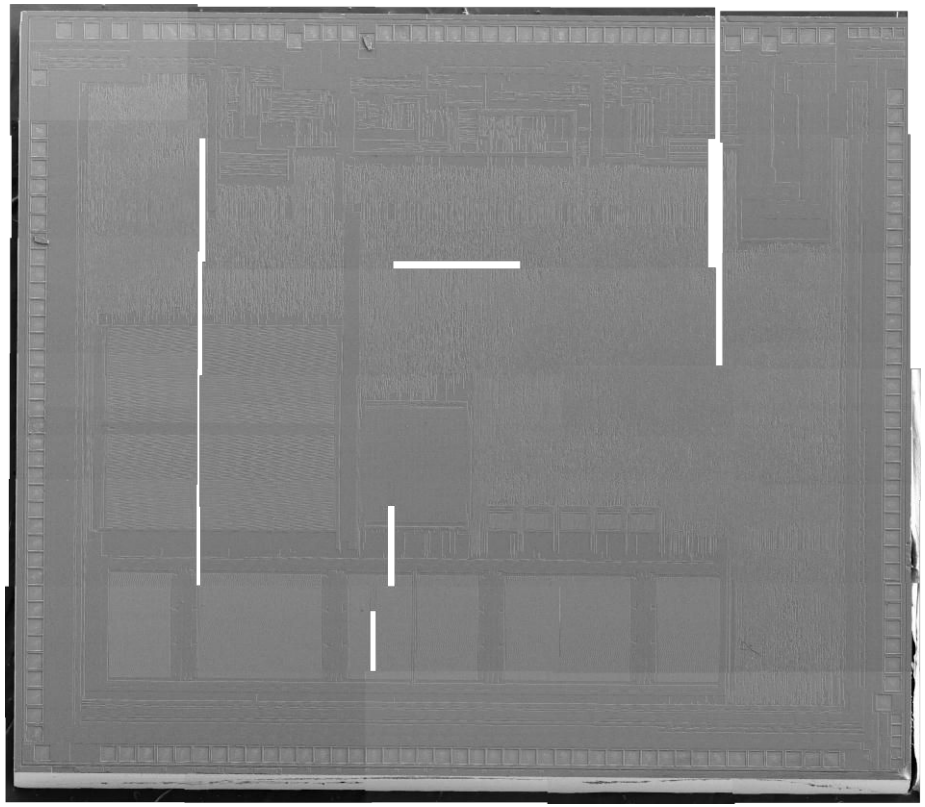


What about Generalplus?

- Company in China, mass produces low-cost ICs
- Commonly found in video games, toys (Tamagotchi)
- Same as Natalie, browsed datasheets until…

Matching pad layout





GPL169256A

- 16-bit u'nSP MCU
- LCD controller
- 256K mask ROM
- ICE debug interface
  - Tried to get a debug probe
  - They didn't fall for it.
  - Probably disabled anyways

MCU audio format support

- Datasheet lists supported audio formats
- Google everything
- Found a GitHub repo with compiled u’nSP libraries
- Matched byte pattern
  - SACM_DVR1800

6.16. Audio Algorithm

The following speech types can be used in GPL169256A: PCM, LOG PCM, SACM_A1600, SACM_1601, SACM_S200, SACM_S480, SACM_S530, SACM_S720, SACM_S320, SACM_S880, SACM_DVR1800, SACM_DVR520, SACM_DVR1600, SACM_DVR4800, and SACM_DVR3200. For melody synthesis, the GPL169256A provides a SACM_MS01 (FM synthesizer) and SACM_MS02 wave-table synthesizer.

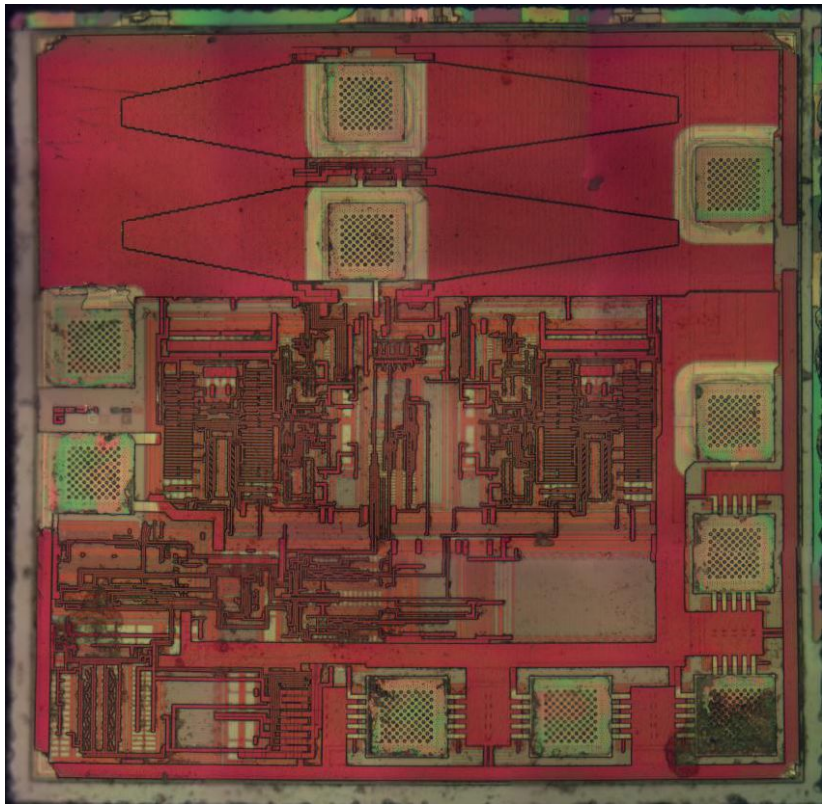
SACM_DVR1800

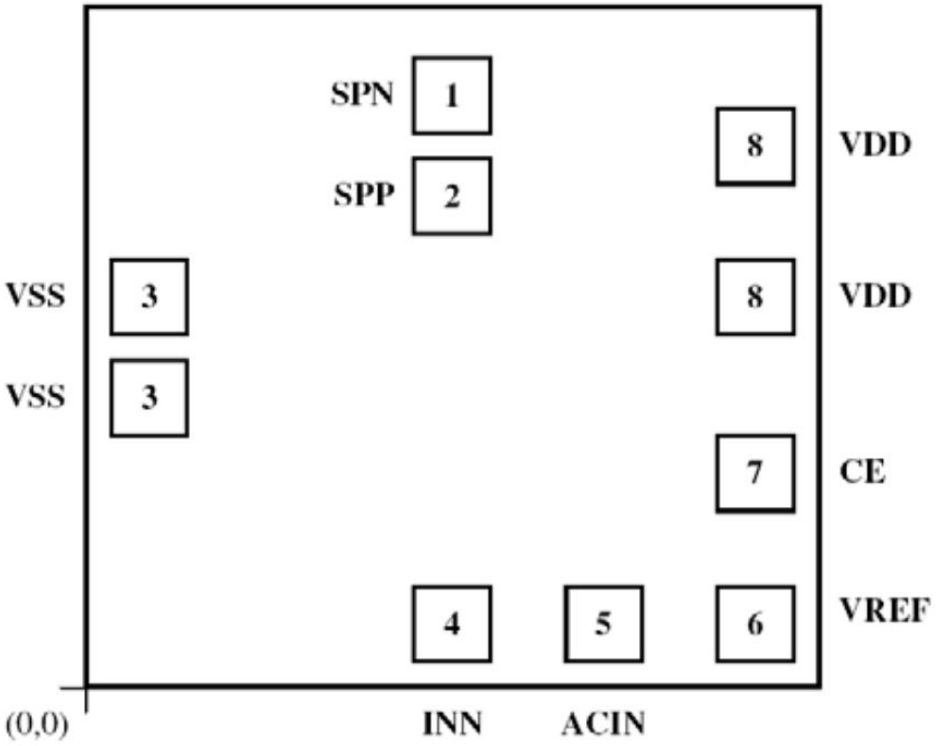
- u'nSP library created with unSPIDE LibMaker

- Library format reverse engineered by David Carne
  - Tools to unpack object files
  - IDA Pro loader with symbol support
  - http://github.com/davidcarne/unsp_tools

```
SACM_DVR1800_IM_BLOCK:00000DAD F_DVR1800_Decode:
SACM_DVR1800_IM_BLOCK:00000DAD                 fir_mov  on
SACM_DVR1800_IM_BLOCK:00000DAE                 r1  = [$15AA]
SACM_DVR1800_IM_BLOCK:00000DB0                 r2  = $1658
SACM_DVR1800_IM_BLOCK:00000DB2                 r3  = [$4422]
SACM_DVR1800_IM_BLOCK:00000DB4
SACM_DVR1800_IM_BLOCK:00000DB4 loc_DB4:                                ; CODE XREF: F_DVR1800_Decode+A↓j
SACM_DVR1800_IM_BLOCK:00000DB4                 r4  = [r3++]
SACM_DVR1800_IM_BLOCK:00000DB5                 [r2++]  = r4
SACM_DVR1800_IM_BLOCK:00000DB6                 r1 -= 1
SACM_DVR1800_IM_BLOCK:00000DB7                 jne     loc_DB4
SACM_DVR1800_IM_BLOCK:00000DB8                 [$4422]  = r3
SACM_DVR1800_IM_BLOCK:00000DBA                 r1  = $1658
```

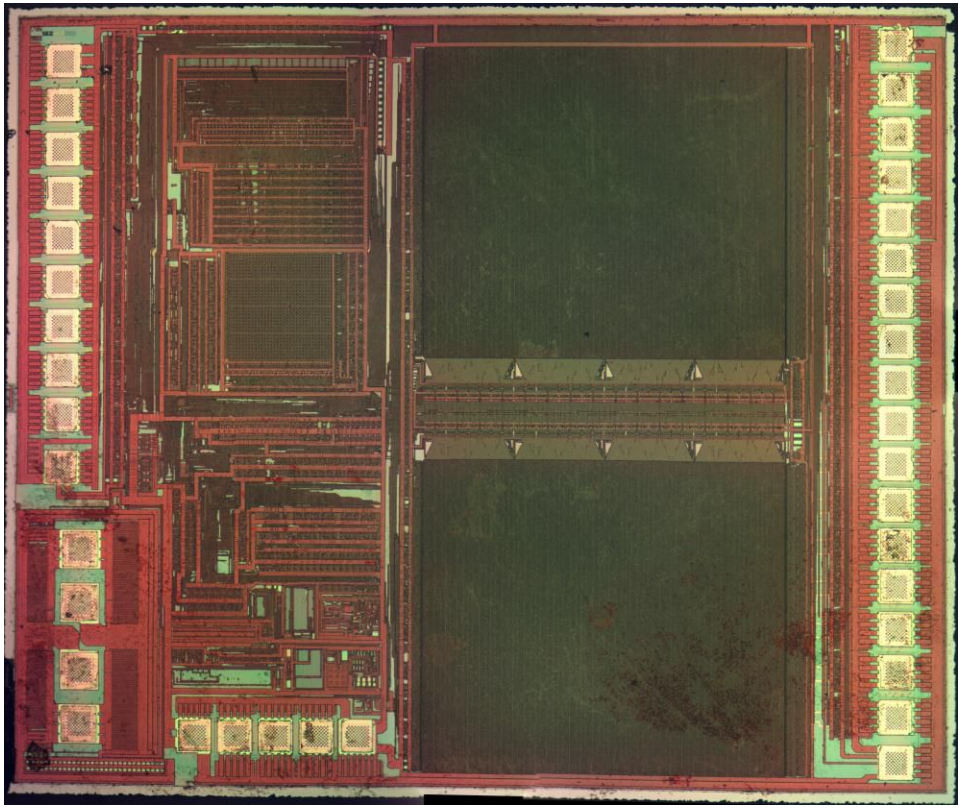
G+ GPY0030x audio driver





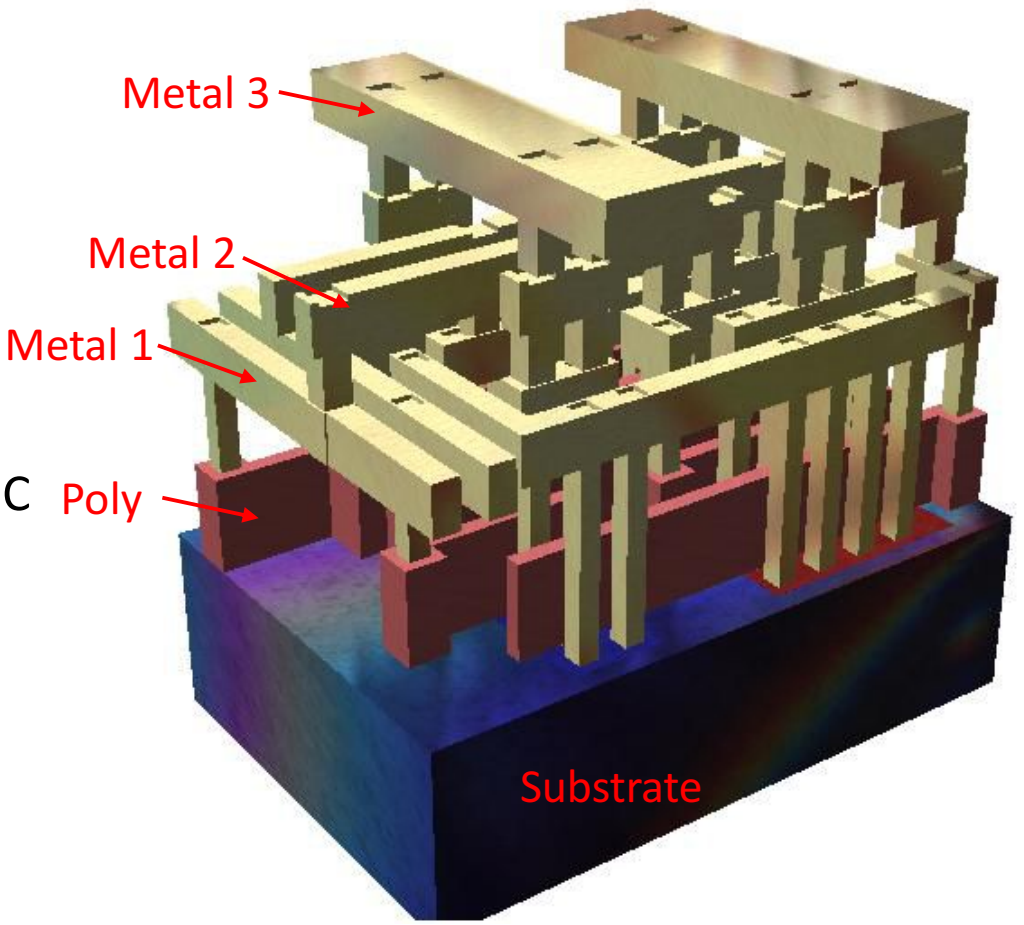
Unknown chip on daughterboard

- GHH393
- Couldn’t match pad layout to datasheet
- Likely still Generalplus
- Microcontroller?
  - Internal clock
  - Connected to peripherals
- Memory chip?
  - Huge memory banks
  - Not much logic



Delayering the chip

- Submerge chip in hydrofluoric acid (3%)
- Commercial rust remover
- Heated in water bath for 1.5 minute intervals
  - Limits temperature to 100°C
- Remove overglass + layers
- 1 metal, 1 poly, substrate (active layer)



Wikipedia

Close up analysis

TODO.txt

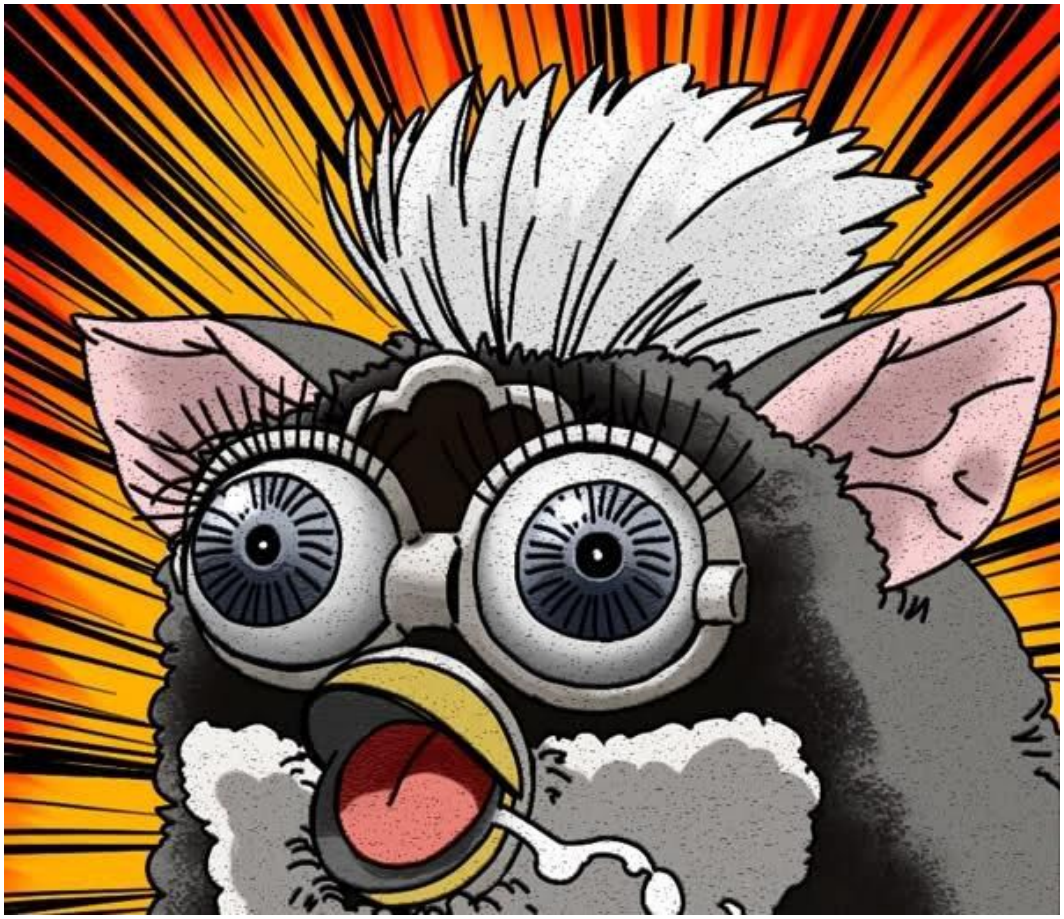
- Extract ROM from daughterboard microcontroller
  - Explore programming-related pads
- Extract ROM from main microcontroller
  - Delayer chip → optical reading?
  - Code exec via power glitching, or fuzzed memory chip?
- Decode audio data
  - Reverse engineer u’nSP implementation
- Perform VR on extracted firmware
  - Delicious Furby 0day

github.com/mncoppola/Furby-2012/

Thanks

- Andrew Zonenberg
- Olivier Galibert
- David Carne
- Segher Boessenkool
- Dr. Geoffrey Davies
- Dr. William Fowle
- Dr. Chuck DiMarzio
- Dr. Wil Robertson
- Kaylie DeHart
- Molly White

Questions?



@mncoppola
poppopret.org