



NandX: Bad Blocks -> Well Hidden

Josh 'm0nk' Thomas

(A DARPA CFT Project by MonkWorks, LLC)
RIP 4.1.13 - Long Live CFT

/whoami (m0nk)

- Applied Research Scientist @ Accuvant
- Engineer for the last ~12 years
- I like to _X_:
 - _X_ = { break, embed, repurpose, solder things into other things, reverse, stare at asm, find logic flaws, win, hustle }
- Find Me:
 - [@m0nk_dot](https://twitter.com/m0nk_dot)
 - jthomas@accuvant.com
 - m0nk.omg.pwnies@gmail.com

echo \$PROJECT_INFO

- Q: CAN I ALL THE THINGS?
 - Got tired of Air to Glass
 - Looking for a reliable way to hide files
 - Ooohh! Is that how NAND Works?
 - Really? So, I can probably reliably hide files?
 - Oh wow, That actually worked?
 - Wait, I can also do that... WTF?
- A: I CAN ALL THE THINGS!

Will he start already?

- Intro
- Defensive Postures (sorry)
- How NAND Flash Works (Hardware)
- How NAND Flash Works (Software)
- Options For How We Can Break It
- How I Broke It
- Forensics / Un-Breaking NAND (Defense Revisited)
- Now What?
- TL;DR:
 - <https://github.com/monk-dot/NandX>

Defensive Posture

- Don't Groan, This Will Be Short
- TL; DR:
 - This is elemental hardware design, there is no “fix”
 - Best bet until we get new tools?
 - Post Analysis, Logs and Forensics
 - Consider not using NAND?
 - Doubtful if you want to embed...

Science is hard, lets do
Science!

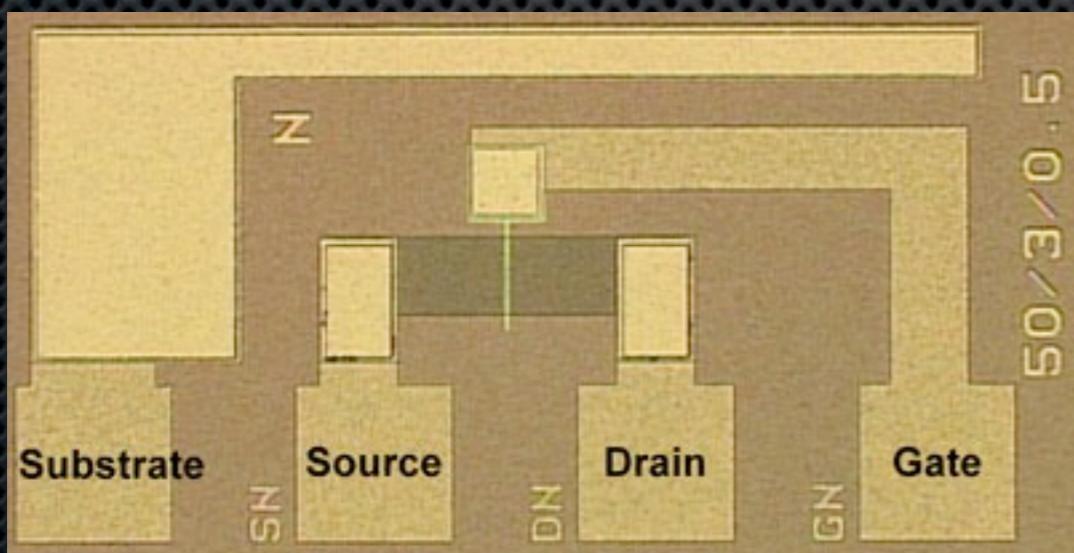
NAND: Hard It Works

- Buckets - Might not be the technical term
- Pages - Typically 512, 2048 or 4096 bytes
- Blocks - Typically 16kb - 512kb
- Initially set to 1 (0xFF)
- Shifting to 0 is easy
- Shifting to 1 is hard



NAND: It's a Trap!

- Gates are hard to build and somewhat fragile...
 - Things break normally after ~10 - 100k writes
- Because they wear out, we do wear leveling to disperse the headache across the full surface
 - Wear leveling leaves residue

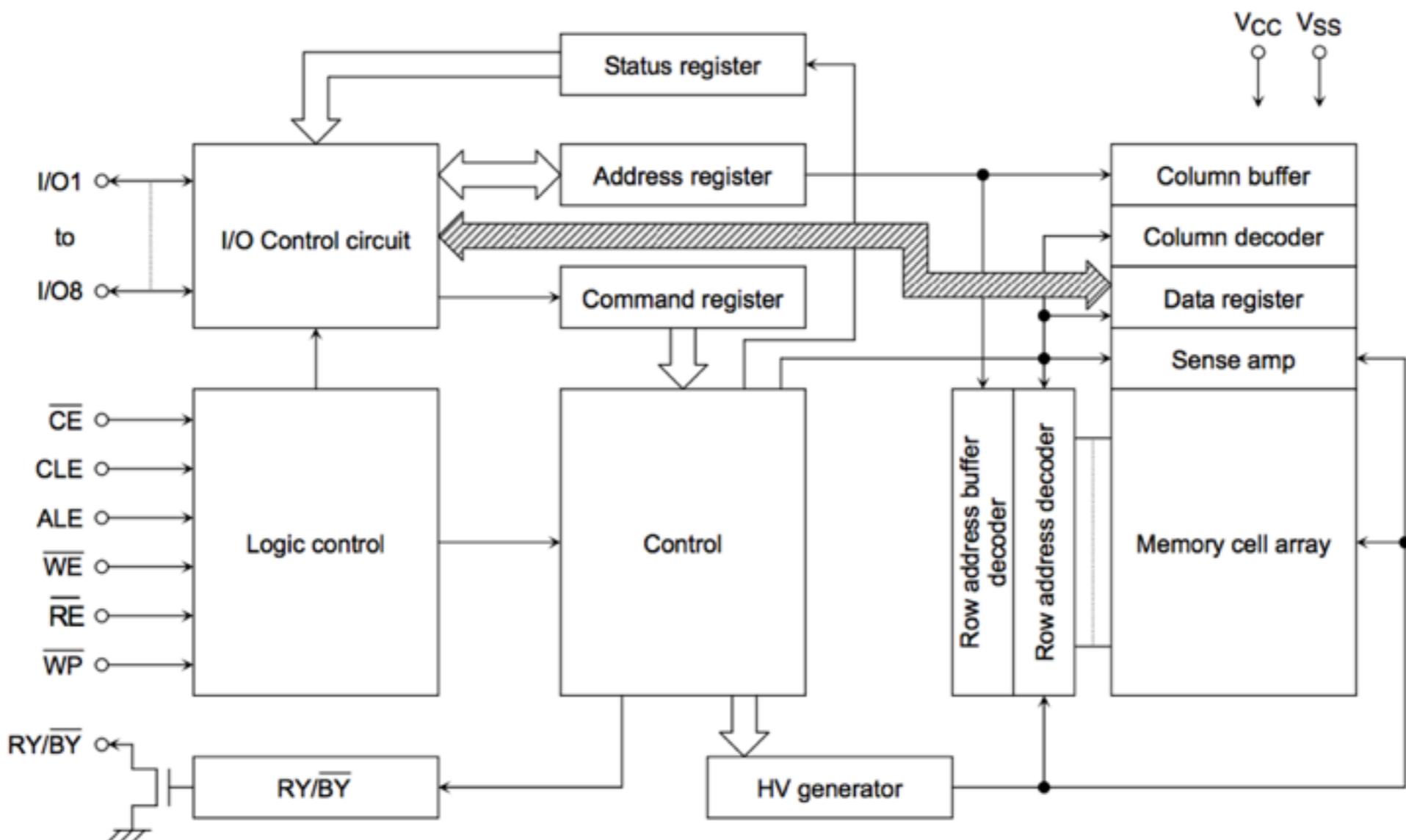


NAND: Hard It Works

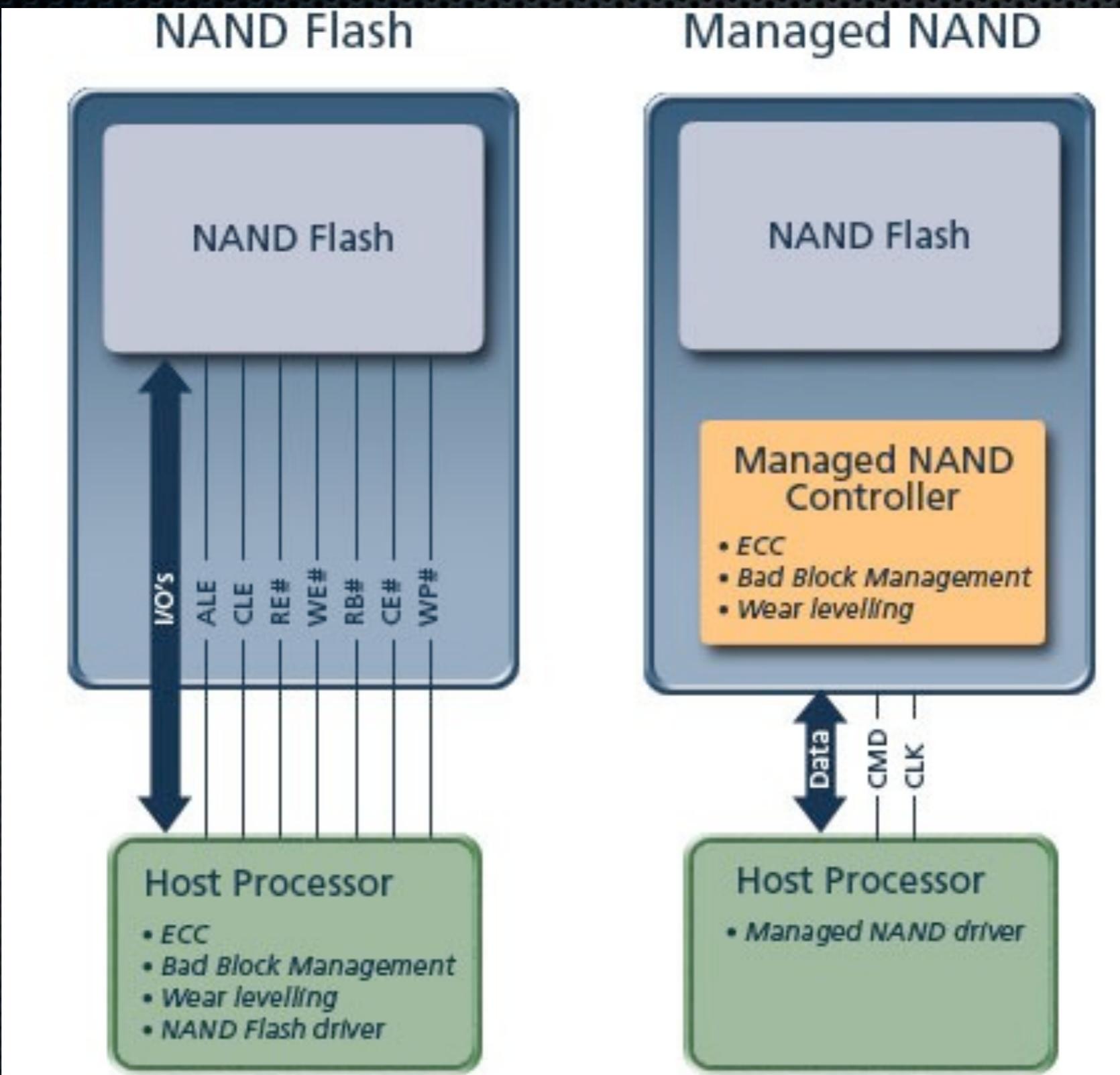
TOSHIBA

TC58DVM92A5TA00

BLOCK DIAGRAM



NAND: Hard It Works



NAND: Hard It Works

- ❖ When Bits go Bad:
 - ❖ BBT / OOB / ECC?

TIL: Notes from the Research

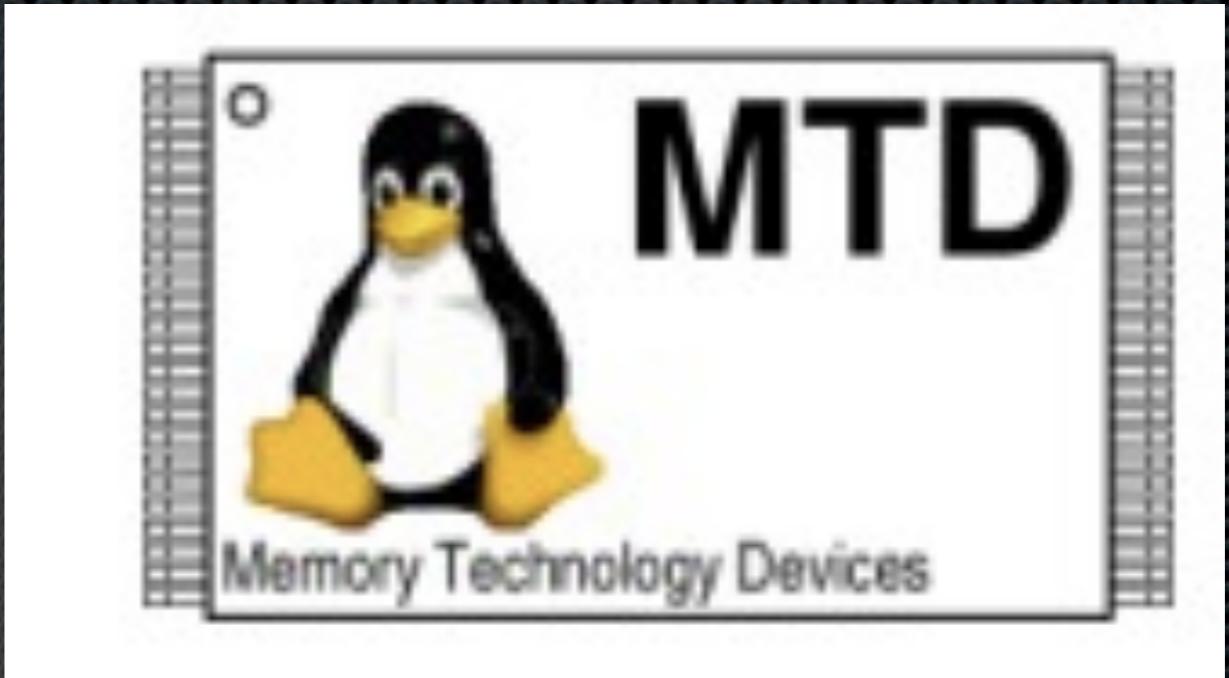
- Some systems fully manage the BBT in kernel memory (this is written back to disk as the “master” during reboot), so you wouldn’t even have to muck with the hardware
- Some systems use dual-page OOB markers for BBT & ECC (Sony!)
- Some systems use 1st or last block for the entire BBT & ECC

NAND: Soft It Works

- RAW NAND vs. MMC/eMMC
 - Complex Driver v. Simple Driver
- Proprietary (closed) wear leveling algorithms are normally embedded
- Still needs to interact with the kernel & the file system code (We can haz API!)

NAND: Soft It Works

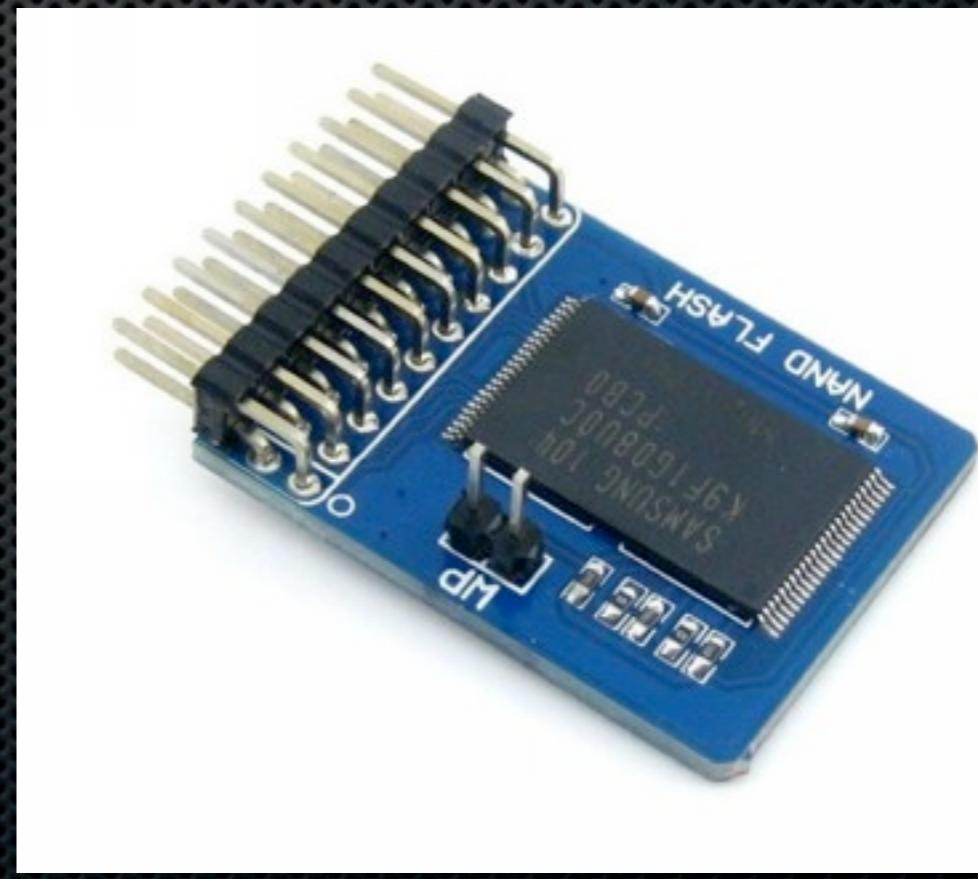
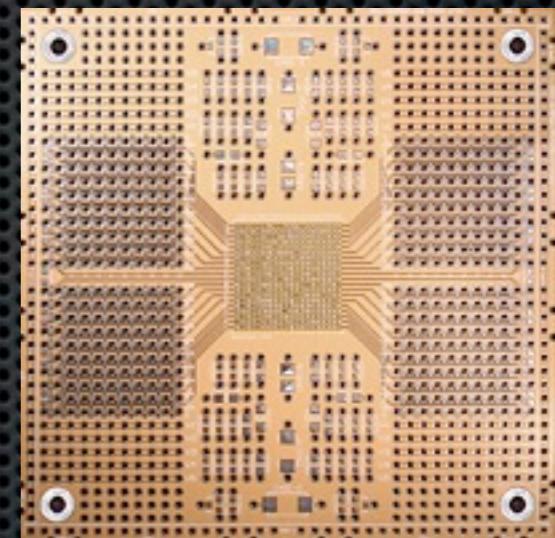
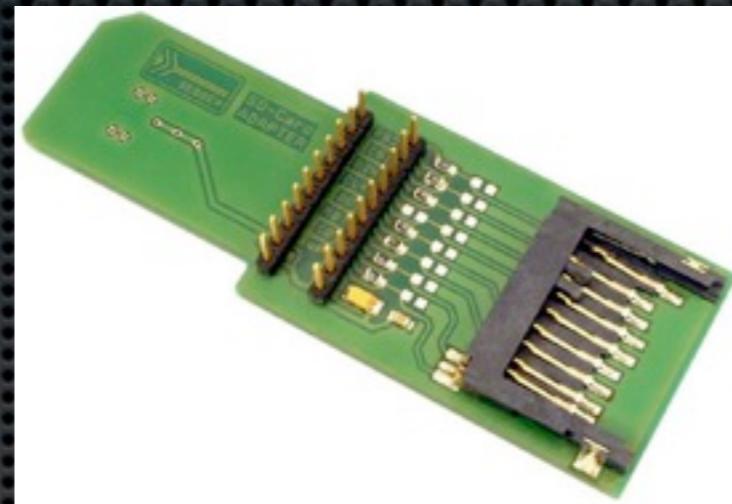
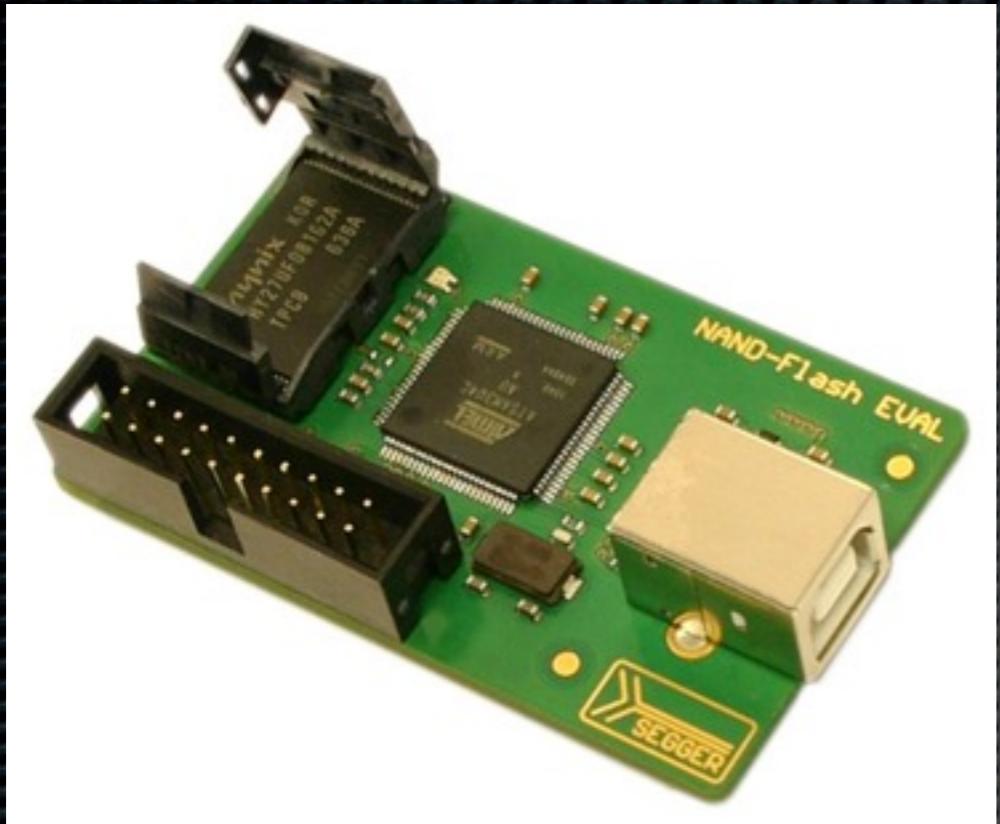
- MTD Subsystem
- Kind of a meta-driver
- Used heavily for boot partitions on Android



Options For Optimal Breakage

- YAFFS and other File Systems
- MTD at the Driver Level
- Android / Linux Kernel
- Flash Transition Layers and Reverse the Embedded Controllers
- Please don't re-de-invent the wheel, hit me up first!

What I expected.



Lets Go Shopping



Visual Palette Cleanse

My Path, And You Can Too!

```
$ cat /proc/partitions  
major minor #blocks name  
  
31      0    409600 mtdblock0  
31      1     6144 mtdblock1  
31      2   103936 mtdblock2  
31      3   430080 mtdblock3  
179     0  7778304 mmcblk0  
179     1  7777280 mmcblk0p1
```

```
$ cat /proc/mtd  
dev: size  erasesize name  
mtd0: 19000000 00020000 "system"  
mtd1: 00600000 00020000 "appslog"  
mtd2: 06580000 00020000 "cache"  
mtd3: 1a400000 00020000 "userdata"
```



My Path, And You Can Too!

- Kernel Modules: Side Loading Fun!
- Sure, I'll be a “test” case

```
<base kernel source>/kernel/drivers/mtd/tests/
```

```
obj-$(CONFIG_MTD_TESTS) += nandx_find_simple.o  
obj-$(CONFIG_MTD_TESTS) += nandx_find_complex.o  
obj-$(CONFIG_MTD_TESTS) += nandx_hide.o  
obj-$(CONFIG_MTD_TESTS) += mtd_oobtest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_pagetest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_readtest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_speedtest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_stresstest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_subpagetest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_torturetest.o  
obj-$(CONFIG_MTD_TESTS) += mtd_erasepart.o
```

My Path, And You Can Too!

- Almost everything I do is simply calling the API in the wrong order
 - The 1 exception is the OOB write
- Path to Winning?
 - Pick a block and wipe it
 - Cover the entire block in 0xDEADBEEF
 - Mark the Block as “Bad”
 - 0x00 out the OOB in the case of Sony
 - Watch the reboot from collision!

nandx_hide.c

```
7530 ►  /* */
7539 static void nandx_file_injector(int blockLocation, void *bufferToWrite)
7540 {
7541 ►  /* */
7554
7555     //TODO: Grab and check return values here!!!!
7556
7557 ►  /*
7564
7565     int err = 0;
7566
7567     //Moves all data out of the target block (no, it really doesn't)
7568     nandx_move_data_from_block( blockLocation );
7569
7570     //Erases the targeted block
7571     nandx_erase_block( blockLocation );
7572
7573     //Injects our buffer directly into the block
7574     nandx_buffer_write_to_block( blockLocation, bufferToWrite );
7575
7576     //Marks the target block as bad
7577     err = nandx_mark_bad_framework( blockLocation );
7578 ▼   if( !err ){
7579         printk(PRINT_PREF "First attempt at marking %d bad failed, going manual\n",
...         blockLocation);
7580         err = nandx_mark_bad_manual( blockLocation );
7581     }
7582
7583 ▾ }
```

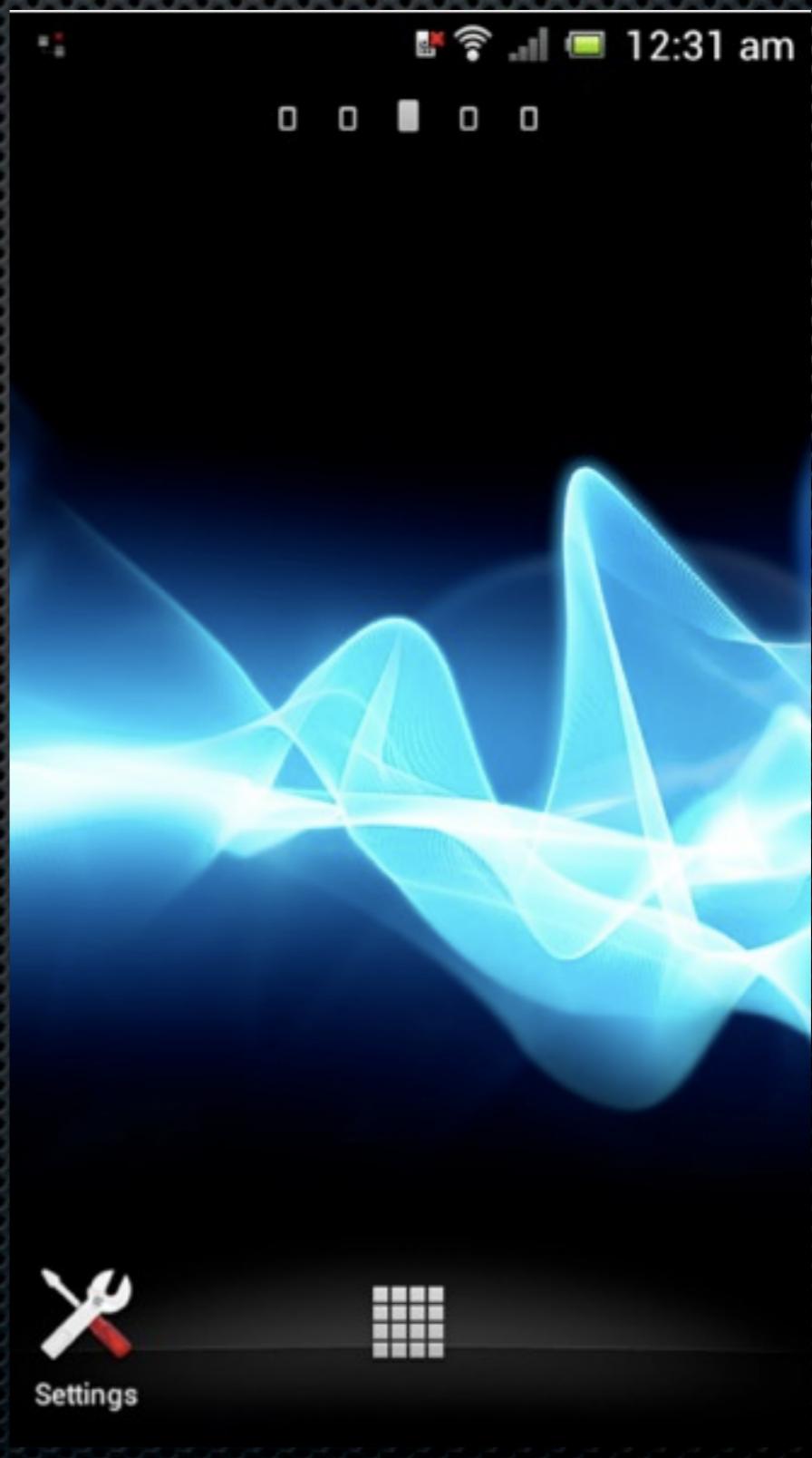
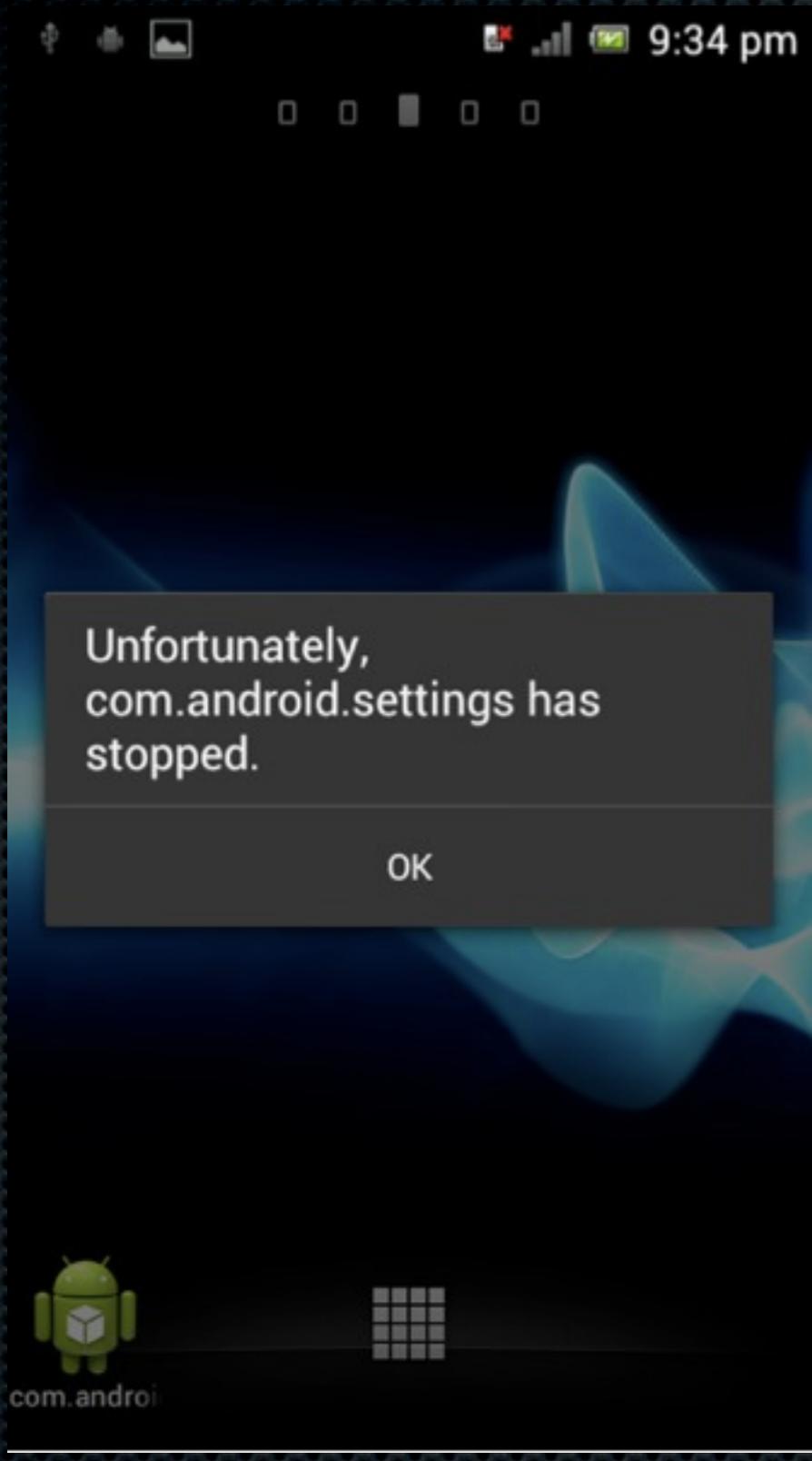
nandx_hide.c

```
7138 ►  /* */
7147 static int nandx_mark_bad_framework(int blockLocation)
7148 {
7149 ►  /* */
7168     int ret;
7169     loff_t addr = blockLocation * mtd->erasesize;
7170
7171     printk(PRINT_PREF "Marking the block %d as BAD\n", blockLocation);
7172
7173     ret = mtd->block_markbad(mtd, addr);
7174     if (ret)
7175         printk(PRINT_PREF "Success - block %d has been marked bad\n", blockLocation);
7176     else
7177         printk(PRINT_PREF "Failure - Why U no mark block %d as bad?\n", blockLocation);
7178
7179     return ret;
7180
7181 }
```

nandx_hide.c

```
7183 ►  /* */
7193 static int nandx_mark_bad_manual(int blockLocation)
7194 {
7195 ►  /* */
7219
7220     int ret;
7221     loff_t ofs = blockLocation * mtd->erasesize;
7222
7223     // THIS CALL IS THE ENTIRE MAGIC OF NANDX-HIDE
7224     ret = msm_nand_block_markbad(mtd, ofs);
7225
7226     if(ret)
7227         printk(PRINT_PREF "We call into the driver and make %d go away.\n", blockLocation);
7228     else
7229         printk(PRINT_PREF "Odd.. even a RAW write on the 00B doesn't kill block: %d\n",
... blockLocation);
7230     return ret;
7231 }
```

My Path, And You Can Too!



My Path, And You Can Too!

```
<6>[19359.863098]
<6>[19359.863098] =====
<6>[19359.863098] nandx_find_simple: NANDX Find for MTD device: 0
<6>[19359.863128] nandx_find_simple: MTD device
<6>[19359.863128]     size 419430400
<6>[19359.863128]     eraseblock size 131072
<6>[19359.863128]     page size 2048
<6>[19359.863128]     count of eraseblocks 3200
<6>[19359.863128]     pages per eraseblock 64
<6>[19359.863128]     OOB size 64
<6>[19359.863128]
<6>[19360.065277] nandx_find_simple: scanned 3200 eraseblocks, 1 are bad
<6>[19360.065277] =====
<6>[19360.065277] nandx_find_simple: MTD block MAP for device: 0
<6>[19360.065307] nandx_find_simple: block 37 is BAD
<6>[19360.065307]
<6>[19360.065307] =====
<6>[19360.065338] =====
```


My Path, And You Can Too!

<Live Demo> AND/OR <Canned Video>

http://youtu.be/AE_oUkKKaBY

My Path, And You Can Too!

- Once the block is bad, it's bad (unless you are me?)
- Flashing a new ROM doesn't reclaim it
- Factory Reset doesn't reclaim it
- 0xDEADBEEF is still there, just kickin' it
- If you are hungry you can just start eating 512kb blocks, one reboot at a time

My Path, And You Can Too!

- ECC stops running once we manipulate the BBT / OOB
- We can still manually run it from the MTD system

Un-Break It With Forensics?

- Start looking @ the Bad Blocks as well?
- Closed vendor secret wear leveling algorithms
- Interleave FTW

I Can All The Things

- “JT Just Went Full Oppenheimer” - Shawn Moyer
- I wanted to hide things in cell phones and all I got was the ability to brick beyond repair most embedded systems.
- You could hide, or just start breaking things in place...

Defensive Posture Revisited

- Education (Thanks for listening)
- TL; DR:
 - This is elemental hardware design, there is no “fix”
 - Best bet until we get new tools?
 - Post Analysis, Logs and Forensics
 - Attempt to force 0xFF on every bad block @ boot?
 - Consider not using NAND?
 - Doubtful if you want to embed...

I'm Bored, Lets Break things

- Kill data in place, wait for IT to wipe and trash the drive, physical exfil FTW

Break Responsibly & Be Cool

- ❖ @m0nk_dot
- ❖ <https://github.com/monk-dot/NandX>