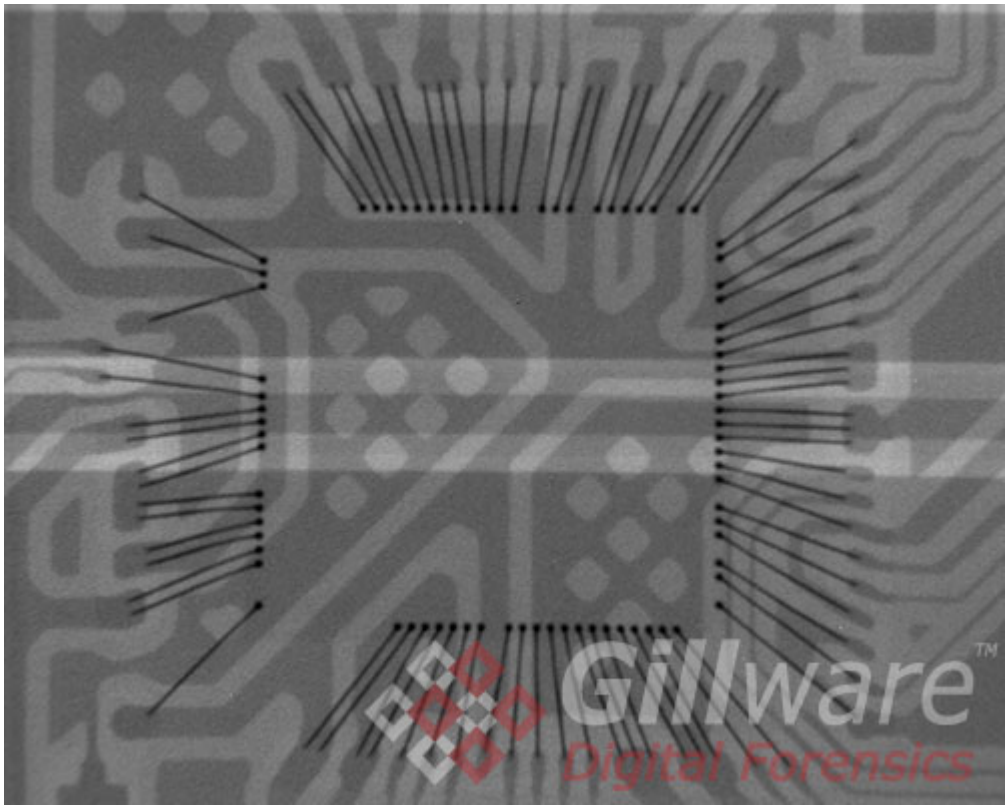
Flash Memory Data Recovery: Get Data through Direct Read of NAND Memory

Gillware Data Recovery Articles: 21 ⋮ 12-15 minutes



"*Anything that can go wrong, will go wrong*."

That's Murphy's Law. How we deal with the fallout afterwards is what determines whether Murphy's Law works for us or against us.

In my first blog post, I reflected on the overlap and ripe potential for synergy between data recovery and digital forensics. During one of my early visits to Gillware, Greg Andrzejewski (Head of Research and Development and co-owner of GDF) told me about a data recovery "win" related to an artifact I'll refer to as *"Flash Memory Amnesia"* that illustrated this point really well.

A Flash Memory Data Recovery Success Story

A professional photographer hired Gillware Data Recovery to attempt to recover data from an SD Card used in her photography business. A happy couple hired her to take wedding photos, and she diligently and artfully captured the first and most memorable moments of the newlywed couple's lives together. Moments that would never happen again. Moments intended to be captured for posterity. Then, she accidentally reformatted the SD Card using the camera it was installed in. Talk about a Murphy's Law moment!

The photographer attempted to recover the images on her own, to no avail. Next, she sent the card to a Gillware competitor, realizing that recovering the data would require professional assistance, and hoping they would be successful. The competitor attempted data recovery and failed. The photographer was told that the data on the card was corrupted and was irretrievable. Murphy strikes again!

Luckily, our intrepid photographer wasn't deterred by the competitor's answer. She called Gillware for a second opinion.

Greg approached this problem the same way most experienced forensicators would. He used a write blocker and previewed the data stored on the card. And he saw all zeros, and an empty FAT table. The card looked the same as it would if it was brand new. Greg explained to the photographer that the reformatting process performed by her camera hadn't just been a simple format. The camera's memory management system had applied a TRIM process as well, readying the card to accept new data.

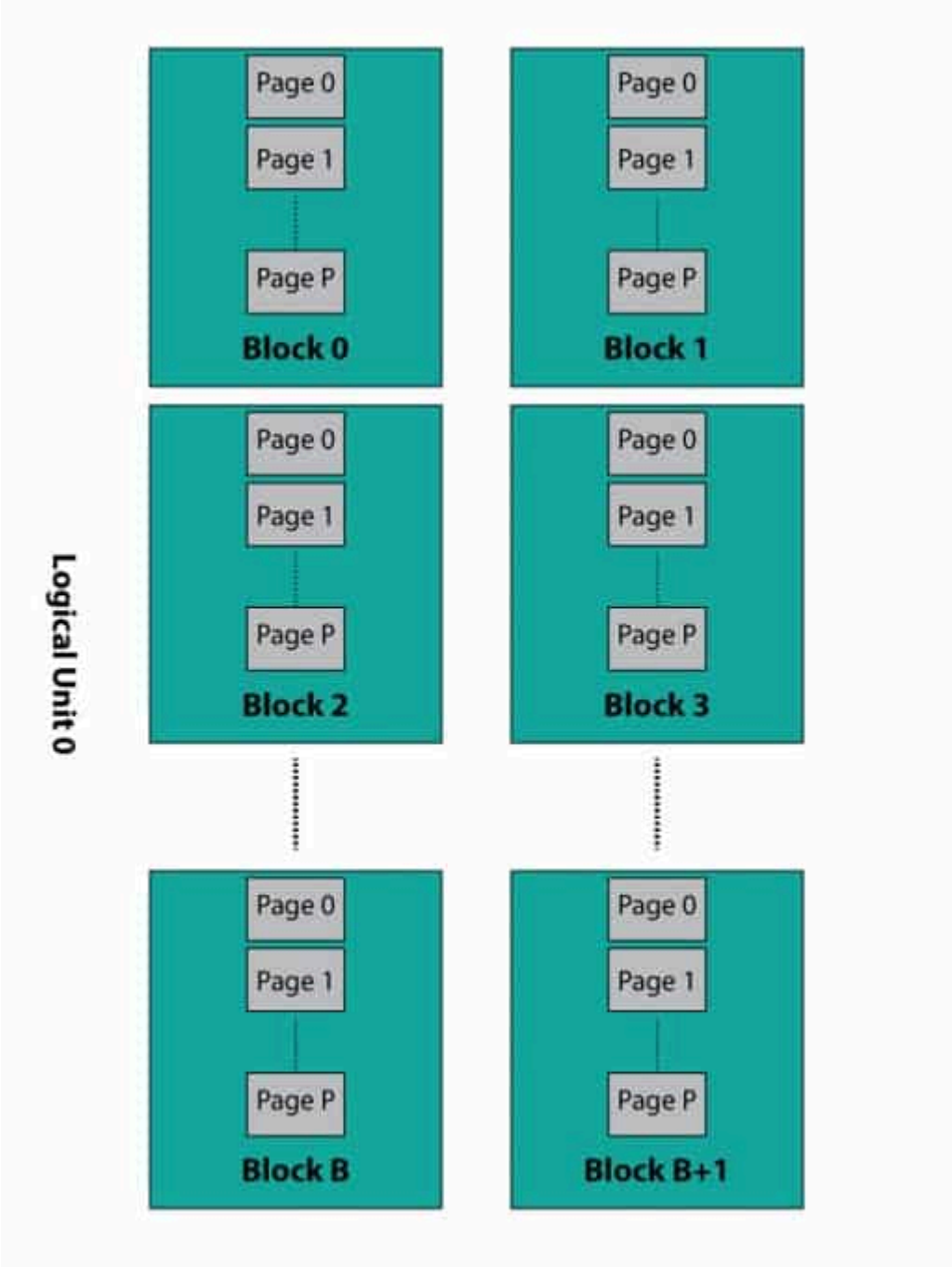
If this flash memory card came into most forensic labs, that would be the end of it. If a flash memory card shows us all zeros, it MUST be empty, right?

But Greg, who pioneered solid state data recovery methods used by Gillware in 2008 knew that while forensic tools and hex editors might show all zeros, with flash memory that *might not be the end of the story*. He told the photographer that there might still be a possibility to get the newlyweds' pictures back.

Here's where the story blew my mind and sparked my imagination…

Flash Memory Amnesia

Previously existing data overwritten at the LBA level where it is accessible to forensic tools *remains in the NAND flash array for indeterminate amount of time.* ***This can represent a potential goldmine for forensicators.*** But that potential hidden treasure of "stale data" that might reside on the NAND waiting to be cleaned up, is inaccessible through normal means. Forensic tools can't see it in order to show it to you. All the tools see is empty memory… *Flash Memory Amnesia.*



Flash Memory Storage

Flash Memory Basics:

NAND flash memory is divided up into equal-sized units, known as **"pages"**. A page is the smallest unit of information that can be read from or written to memory, analogous to a sector on a hard disk drive platter. Chunks of sequential pages are grouped together into **"blocks"**. This is important because while individual pages within a block can be read or written in any order, in order to **re**write a page the entire block must first be erased.

Consider a flash memory device with a page size of 512 bytes and 128 pages per block. A simple way of constructing a camera card would be to linearly map a logical block address (LBA) to each physical page of the device. A problem with this approach, however, is that a change to a single sector would require the entire 64KB (512*128) block be erased and rewritten. Aside from the obvious performance penalty, NAND flash memory blocks have precious few program-erase cycles. LBA ranges containing frequently updated data (FAT table, Master File Table, etc.) would quickly wear out and become unusable.

So, what's a manufacturer to do?

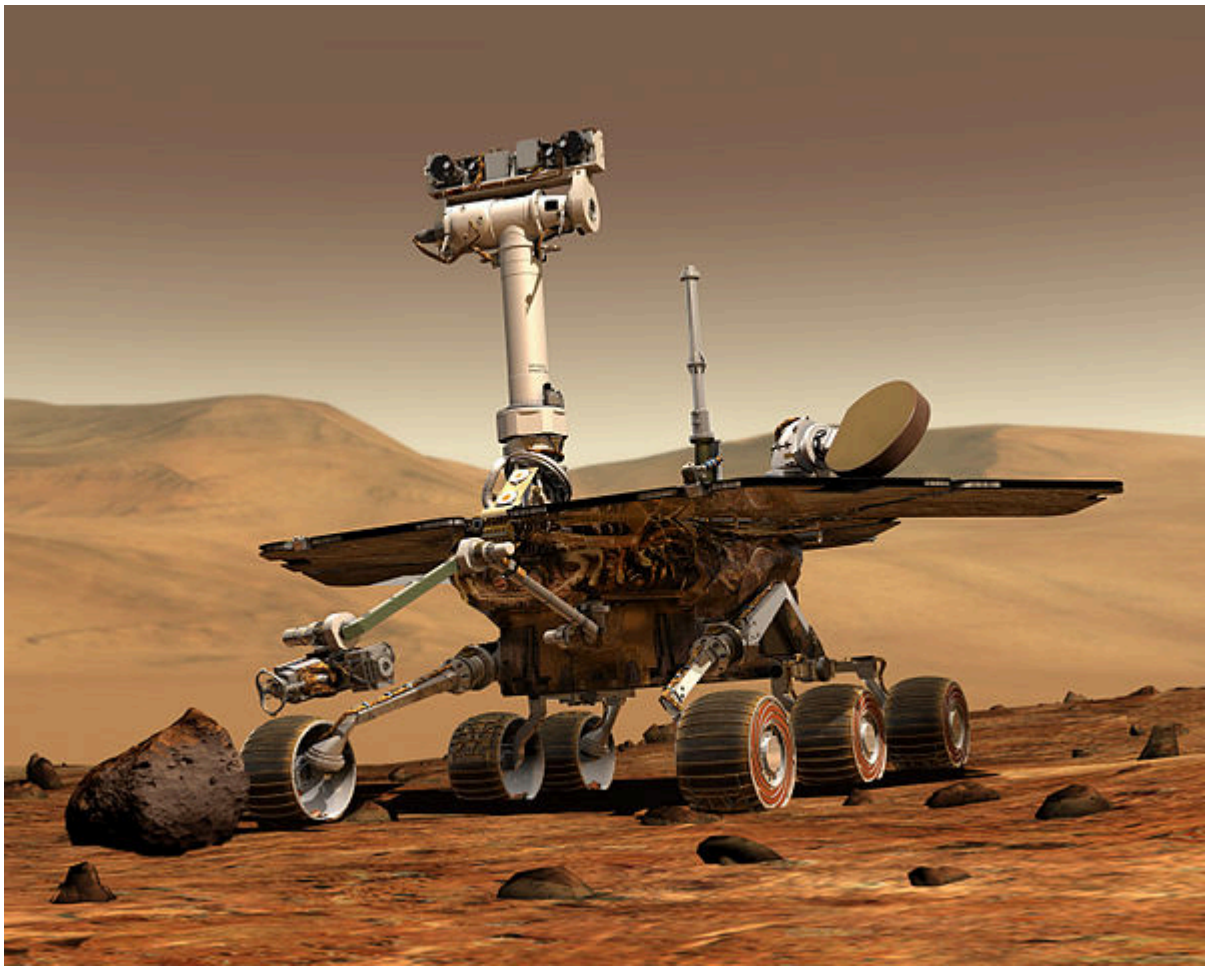
To work around the limitations of NAND flash memory, solid state storage devices employ a powerful microprocessor to manage the underlying storage. Instead of a static logical-to-physical mapping, manufacturers develop sophisticated firmware tuned to maximize both performance and usable life. An integral technique for the latter is known as **wear leveling**. Because each flash memory block has a finite number of program-erase cycles, firmware seeks to spread writes evenly across the flash array.

For forensicators who work in mobile device forensics, wear leveling artifacts are not a new concept. We regularly see multiple versions of data laid down as new data is added, before garbage collection and other clean-up mechanisms kick in.

As a result of wear leveling, new data written to an LBA is practically never stored to the same physical location. Rather, the new data is written wherever the firmware sees fit and the Flash Translation Layer (FTL) is updated with the new physical location. The FTL is the firmware component responsible for logical-to-physical mapping, keeping track of where the data for a particular LBA is actually stored.

But what about the previous data stored at that LBA? If the new data is written somewhere else, does that mean the old data is left just sitting there? In a word, yes. Well, until the firmware gets around to erasing it. *Pause and let that sink in a bit.* Previous data, data overwritten at the LBA level, say by a reformatting event or by writing zeros over all the data on the card, *remains in the NAND flash array for an indeterminate amount of time*.

Exactly how long previous data hangs around depends on a variety of factors, but particularly the aggressiveness of the firmware **garbage collection** routine. Part of the internal process of updating data stored at a particular LBA is to mark the previous data as no longer needed. During periods of inactivity, device firmware will perform garbage collection, which seeks to erase blocks containing unneeded data so that they may be reused. Unbeknownst to the garbage collector, it may be inadvertently destroying evidence critical to an investigation. Such phenomena are sometimes referred to as SSD “self-corrosion”.



Mars Rover Opportunity – Flash memory amnesia, and major Murphy's Law incident.
Picture from NASA/JPL/Cornell University, Maas Digital LLC

Flash Memory Self Corrosion Mechanisms

Solid state memory uses a number of mechanisms to clean up its own act so that it can operate efficiently. These mechanisms are sometimes referred to as "***self-corrosion mechanisms.***" In the forensics world, things that make changes to other things in ways that are beyond our control are problematic. Some even claim that such things spell the end of forensics. I have read a lot of papers and talked to a lot of people about work related to these issues and had accepted self-corrosion as a difficult barrier to overcome in forensics. It's not the first, it won't be the last. Thankfully, so far our field hasn't imploded in the face of these various challenges.

Another self-corrosion concept is **TRIM**, which acts as a hint from the OS to the device that data contained in specific LBAs is no longer needed (i.e. after a file is deleted, or when an SD Card is formatted like the one in our story was). The data from TRIMmed LBAs ***may remain visible*** for a while on some devices (non-deterministic TRIM), but on most devices (including our photographer's), it will vanish immediately.

Thus the concept of ***flash memory amnesia***. Data might still exist on the NAND but the device appears to have forgotten about it. More specifically, the device has been instructed to lose the data by a TRIM command issued by the OS. This is actually quite beneficial to the device, as the FTL can become increasing convoluted over time. Following a TRIM command, the logical-to-physical mapping for the entire LBA range can be purged from the FTL, easing the burden on the firmware of remembering where everything is actually stored. A read request for an LBA not in the FTL can be filled by simply returning all zeros. The OS won't have any idea that there's "stale" data content in NAND, and neither will your forensic tools.

Flash Memory Data Extraction Options

With Flash Memory Devices we have two avenues to data extraction:

1) Through the device's normal device interface (SATA, eMMC, SDIO, etc…), or

2) Via direct read of NAND Flash memory of one or more chips.

Individual NAND flash memory chips use industry-standard interfaces, and hardware adapters for reading memory chips are available for a couple hundred dollars. Directly reading the chip seems pretty straight forward.

Unfortunately, it's extremely difficult (…maybe near-impossible) to make use of raw data read directly from NAND for a number of reasons:

- Most modern devices use AES-128/256 encryption
- The examiner must manually determine FTL logical-to-physical mapping
- Wear leveling can leave behind hundreds of previous revisions for any given LBA
- Proprietary error-correction routines must be performed to make data readable
- Removal of the NAND chip is physically destructive to the original device

That's the "why not" of direct read of NAND flash memory. But sometimes pushing into that extremely difficult territory yields amazing results.

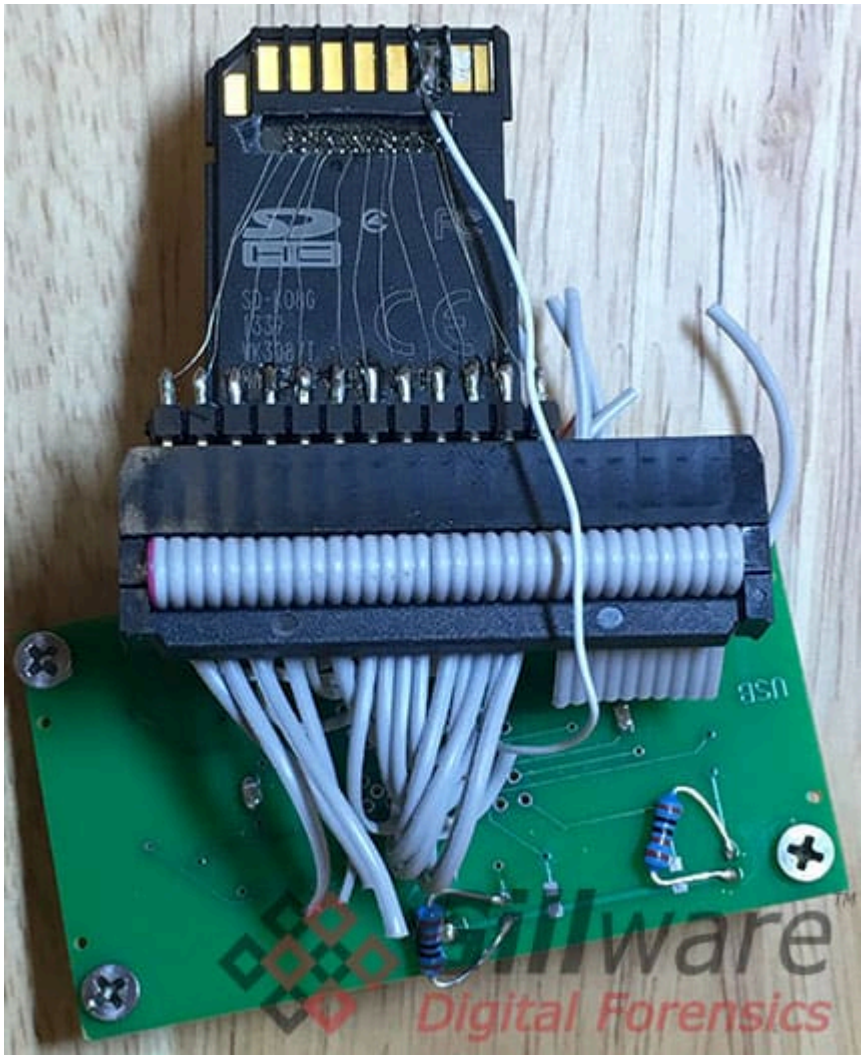
Battling Flash Memory Amnesia:

Greg went to battle with Murphy's Law on behalf of the photographer. He successfully fought flash memory amnesia by resurrecting the data through a direct NAND Memory read. Direct read of the NAND wasn't the only challenge he faced. The SD Card was a monolith style card rather than a traditional SD Card with easily accessible components.

In the picture below, you can see the comparison. On the right is a traditional SD Card with the PCB components exposed. On the left is a monolith style SD card:

Monolith style SD Card (Left) Compared with traditional SD Card (Right)

Accessing the NAND chip on a monolith card is no easy matter. But the NAND chip is in there somewhere under all that plastic. The question is, how do you get to it?



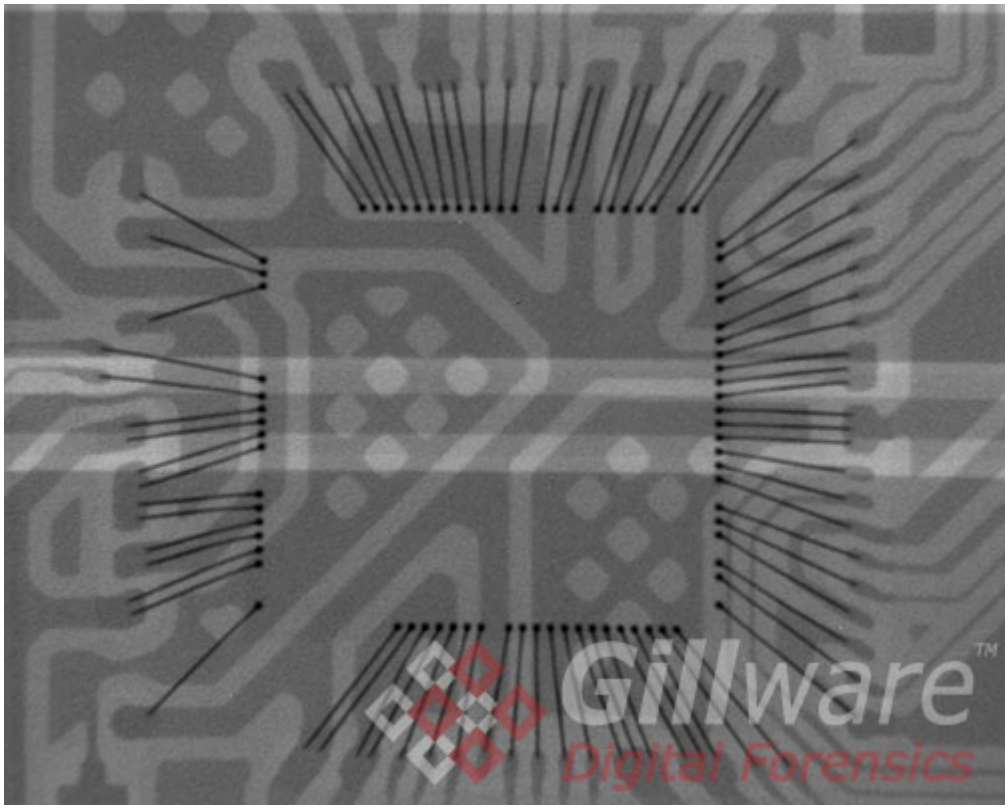
Direct access to NAND Memory

If you look closely at the underside of the monolith, you'll notice a pattern of tiny test points just below the SD contact fingers. Test points are used to write an initial firmware image to the flash memory during the manufacturing process. If test points can be used to write to raw flash memory, then and it stands to reason that they can also be used to read from it. Trouble is, there's no telling which test point corresponds to which signal. Deciphering the pin mapping requires a bit of good old reverse engineering. A logic analyzer is then used to observe activity at each test point during normal operation. If the test points truly connect to the NAND flash memory bus, the signal assignments will slowly reveal themselves as the controller interacts with flash memory.

For more detailed information of how this works, see Greg's video.

With the pin mapping in-hand, it's simply a matter of making the necessary connections to dump the raw NAND flash.

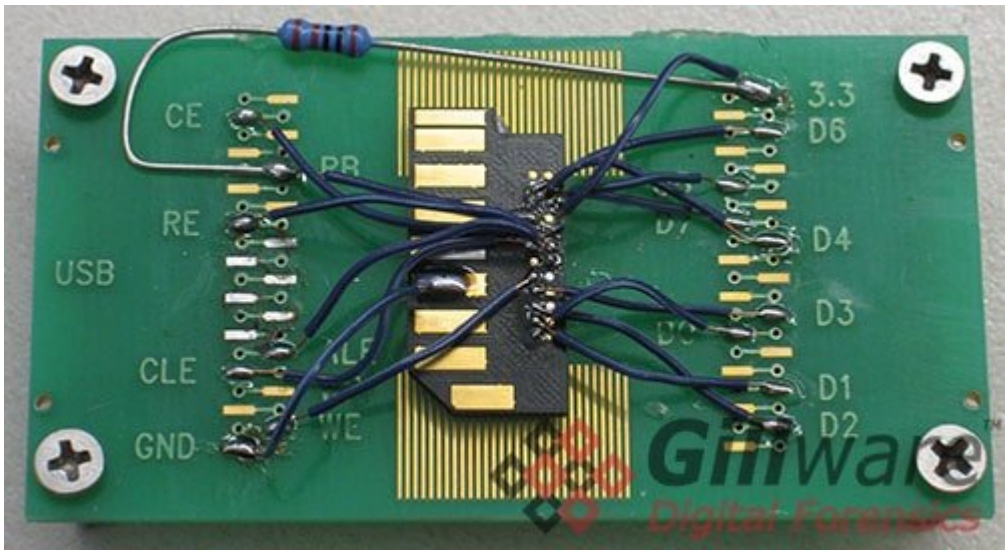
And thus, with a lot of patience and a good deal of testing, Greg got the data out of the NAND on the SD Card. He then reconstructed a disk image from a hand-assembled FTL, carefully restoring TRIMmed logical-to-physical mappings, applying the correct proprietary error-correction routine in a process worthy of its own future blog post.



X-ray image of a monolith style SD Card showing bond wires connecting to the controller die.

A Surprising and Happy Ending:

And Greg defeated Murphy's Law. He recovered over 300 pictures from direct read of the NAND chip from the SD Card.



SD Card wired for data extraction via direct read of flash

The application of this data recovery concept to digital forensics is clear. That SD Card could just as easily belong to a suspect in a child exploitation case, or any number of other types of cases for that matter. The card could be reformatted on purpose.

A seemingly “empty” SD Card could actually contain a plethora of user created data. As forensicators, we need to be aware of that fact, and we need to find ways to access that potential data if the case requires us to dig that deep.

Just remember: Nothing is impossible, unless you let Murphy’s Law stop you from trying again and again when things go wrong.

So here we arrive at Andrzejewski’s corollary to Murphy’s Law: When things can go wrong and do go wrong, keep working on solving the problems in front of you one at a time until the seemingly impossible becomes possible.