Hacking Hardware With A $10 SD Card Reader

by Amir “Zenofex” Etemadieh, CJ “cj_000” Heres, Khoa “maximus64” Hoang
www.exploitee.rs

Introduction

Hardware hacking is increasingly important as more devices become connected. Systems are moving beyond standard hard drives and CPUs and security professionals find themselves at a loss without purpose built tools. This whitepaper aims to solve the biggest challenge facing a security professional that is looking at a hardware device: acquiring the target. To that aim, this paper contains information on reading and writing eMMC flash devices through the use of an SD Card reader/writer. Techniques for learning how to recognize eMMC flash, identifying the correct eMMC pinout, attaching to eMMC flash within an embedded device, and selecting the correct USB SD card adapter for use in reading/writing to flash will be addressed. Properly equipped, security professionals should gain the ability to read the code from targets and use that in their standard analyses. We urge readers to read this entire paper before attempting to utilize this technique as there is crucial information spread throughout and, as with all hardware, a misstep could break the target hardware.

Prior work

There have been many different papers and presentations on auditing hardware in the past, and several of them went into the topic of acquiring the code from the target. One such example comes from Micah Elizabeth Scott (@scanlime). On her blog, Micah discussed how she went about reverse engineering an eMMC flash pinout from a Nintendo DSi console. She stated that her intention was to build a passive sniffer to view the CPU’s attempts at reading and writing from flash[1].

Following her post, we began our research on eMMC flash and eventually demonstrated some of our work at the DEF CON 21[2] security conference in 2013. Over the following years we have developed a systematic approach of identifying, reading, and writing eMMC flash as well as a set of specialized low-cost tools we use to simplify the process. The main benefit from the current work being presented is that it takes into account this breadth of experience and distills that experience into standard procedures and advice.

Introduction to eMMC flash

Before eMMC, many different types of Flash storage were in use within electronic devices. Flash chips required programming parameters a priori that were specific to the vendor, model, and sometimes revision of the flash chip. Additionally, each device manufacturer was required to implement algorithms such as data whitening, bad block handling, and other low level nuances. The Embedded Multi-Media Card, more commonly known as eMMC, was designed to address these shortcomings and has become one of the most prevalent types of flash storage media in the world. eMMC flash exists in a wide range of devices including phones, tablets, televisions, automobiles, and refrigerators. Developed by the Joint Electron Device Engineering Council (JEDEC), eMMC in now in revision 5.1 of the specification which has increased speed and capabilities over previous revisions.

According to a 2014 NXP Presentation, there are an estimated 4.375 Billion eMMC chips on the market[3]. Although this seems a bit farfetched, the first five Samsung Galaxy Smart Phones (Galaxy S - 24M[4], Galaxy S2 - 28M[5], Galaxy S3 - 30M[6], Galaxy S4 - 20M[7], Galaxy S5 - 12M[8]) sold over 110 million units, and these numbers are on the low end of the public figures.

---

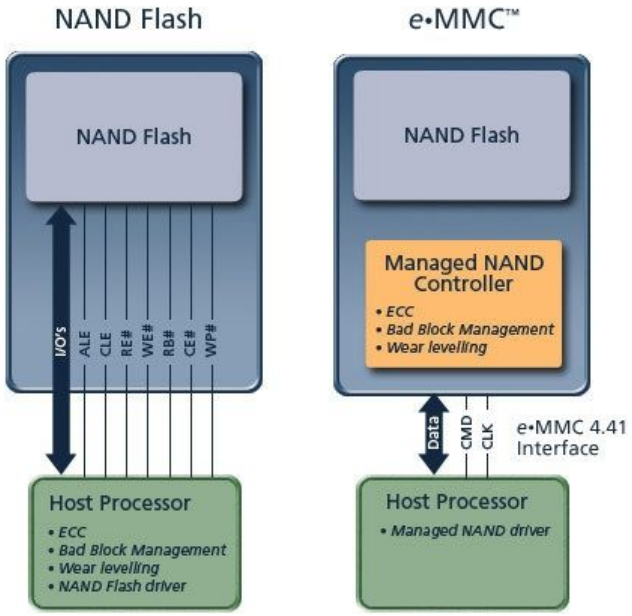
[1] http://scanlime.org/2009/07/dsi-hacking-bga-rework/
[2] http://download.exploitee.rs/file/generic/GTVHacker-DEFCON21.pdf
[3] http://cache.freescale.com/files/training/doc/ftf/2014/FTF-AUT-F0375.pdf (Slide 11, estimated at 16gb units)
[4] http://www.androidauthority.com/galaxy-s-galaxy-s2-note-sales-records-91660/
[5] http://www.androidauthority.com/galaxy-s-galaxy-s2-note-sales-records-91660/
[6] https://thenextweb.com/asia/2012/11/05/samsung-galaxy-s3-30-million
[7] https://www.theverge.com/2013/7/3/4489460/samsung-galaxy-s4-20-million-shipments
[8] https://www.theverge.com/2014/11/24/7273817/samsung-management-shakeup

eMMC flash has become prevalent in many devices due to its low cost, speed, multitude of storage sizes, smaller footprint, lower power usage[9], and integrated controller. This allows engineers to create smaller, lower power, less complex boards, with faster development cycles afforded by an easier to use interface. The eMMC interface is similar to that of a Multi-Media Card (MMC) and also that of a Secure Digital (SD) card. As many system-on-chip environments support a direct connection to the Secure Digital Input Output (SDIO) bus, it allows for an easy and direct interface to the eMMC chip.

Comparison of eMMC to Other Flash Types

eMMC is similar to other types of flash, such as NAND and NOR, but is unique in that it contains not only the flash memory but also an integrated flash controller. This allows eMMC to self manage typical flash issues, such as bad blocks, and conduct life lengthening techniques, such as wear leveling, typically done externally with NAND without a device manufacturer having to write a single line of code.

Figure 1: NAND vs eMMC[10]



As illustrated above, NAND flash requires traces routed to an external controller leading to a more complex design and higher part count while eMMC contains both within the same silicon die. This eMMC integration leads to a less complex design that requires fewer communication lines and allows for easier communication. Additionally, the integration allows for reading and writing to flash without having to account for modifying error correction data, which the controller handles.

Identifying eMMC flash

Identifying eMMC flash can be challenging but there are a few key indicators which can help identify the type of flash memory inside the device. The location of the flash memory on the board, package type, markings/silkscreening, chip manufacturer, and electrical lines can all help to identify eMMC flash memory.

[9] http://www.kingston.com/us/embedded/emmc
[10] http://www.micron.com/products/nand-flash/choosing-the-right-nand

Some devices feature a System on Chip (SoC), which is the main CPU coupled with I/O interfaces within the package. Nearby, devices will often feature some type of memory (RAM), input/output at a high level (USB, Ethernet, or video output), and some type of flash memory chip. On occasion, the flash may be on the opposite side of the circuit board requiring full documentation of all large components on the board. Locating the memory chip on the board is integral in determining if the device is using eMMC flash.

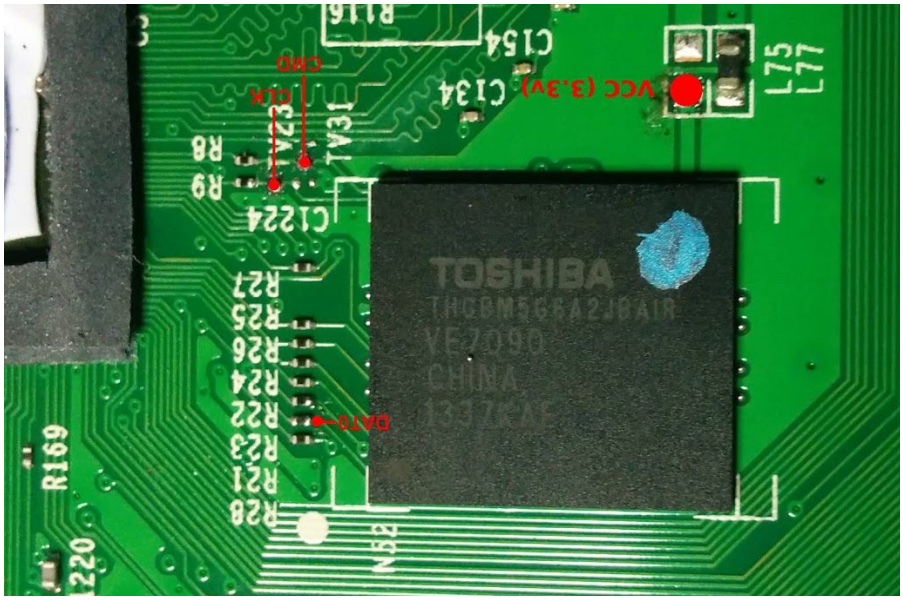
Flash can come in several packages however, we will focus on two: Ball Grid Array (BGA) and Thin Small Outline Package (TSOP). eMMC flash is typically found in the BGA form factor **(Figure 2)** which features a grid of circular pads beneath a thin layer of silicon. Other package types for flash memory may include TSOP which is very commonly used for NAND **(Figure 3)**.

Figure 2: eMMC Flash in Ball Grid Array Package

Figure 3: NAND Flash in Thin Small Outline Package

Another mechanism for identifying both the eMMC chip and eventually the data lines are the markings on the board (the silk-screen layer). In some cases, engineers will clearly identify an integrated circuit (IC) as an “eMMC flash” with all of the data lines labeled, while others may only have a designator such as “U2”. Expanding on this, some manufacturers can place stickers with a software revision on top of the flash chip which can obfuscate any other labeling but may provide other clues on identification. Although such revisioning information such as stickers or colored paint can get in the way, they provide quite an obvious clue that software is contained within the chip.

Figure 4: Toshiba eMMC Flash - THGBM5G6A2JBAIR



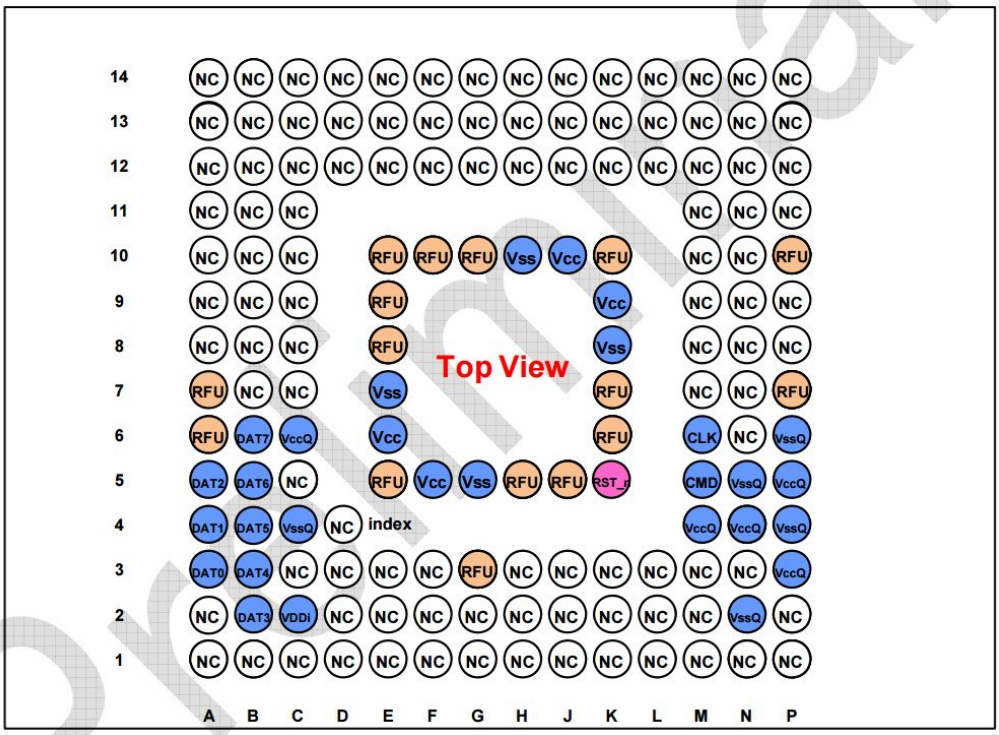
While looking at the suspected chip, the top line of the labeling will, in most cases, be the manufacturer's name with the line below being the part number. With a bit of research, this part number can be used to correctly identify critical information such as the flash type and pad layout. For example, in the above **(Figure 4)**, the BGA eMMC chip that is shown is made by Toshiba and has the part number “THGBM5G6A2JBAIR”. In most cases, the end of the part number is used to identify the size, package, heat, lead, and date of manufacture parameters. Usually, these letters and numbers will not appear within a data sheet; therefore, if no results are found for the entire line when searching for the chip, continue trying queries by removing alphanumerics at the end. In the above example, searching for the full part number on the internet found a result containing the datasheet. The datasheet indicated the flash type as well as details on the communication requirements and layout[11].

eMMC Devices

Now that we have found a datasheet for the eMMC chip, we can see how its pads are laid out. Often, the pinouts for the main data lines are the same across multiple manufacturers (**Figure 5**) to allow parts to be interchanged since many manufacturers require multiple suppliers to ensure supply chain resilience.

Figure 5: eMMC Flash Datasheet - Toshiba THGBM5G6A2JBAIR

[11] http://www.boardcon.com/download/THGBM5G6A2JBAIR-19nm-8GB-e-MMC_E_rev0.2_120607.pdf



The image above (**Figure 5)** is shown from a top down view (looking “through” the chip). The NC or “No Connect” circles in the image are simply present to help secure the chip to the board and provide no meaningful connectivity. The important circles are highlighted in blue and are required for communication. Specifically, these are *DAT0* to *DAT7* (on the left of **Figure 5**), as well as the *CMD*, *CLK*, *VCC*, *VccQ* and *VSS*.

Finding in-circuit pinouts of eMMC

Taking the most non-invasive approach while attempting to find the in-circuit pinouts of an attached eMMC chip can be difficult but the process can be simplified. In the datasheet excerpt above (**Figure 5**), the most salient pieces of information are that the data lines are located in the lower left side of the footprint, while the *CMD* and *CLK* lines are located in the right side of the footprint.

With a small amount of photo manipulation the background of the above image (**Figure 5)** can be removed (**Figure 6**) and the result is a very usable image that can be overlaid against the original chip (**Figure 7**).

Figure 6: Manipulating eMMC Pinout from Datasheet in GIMP

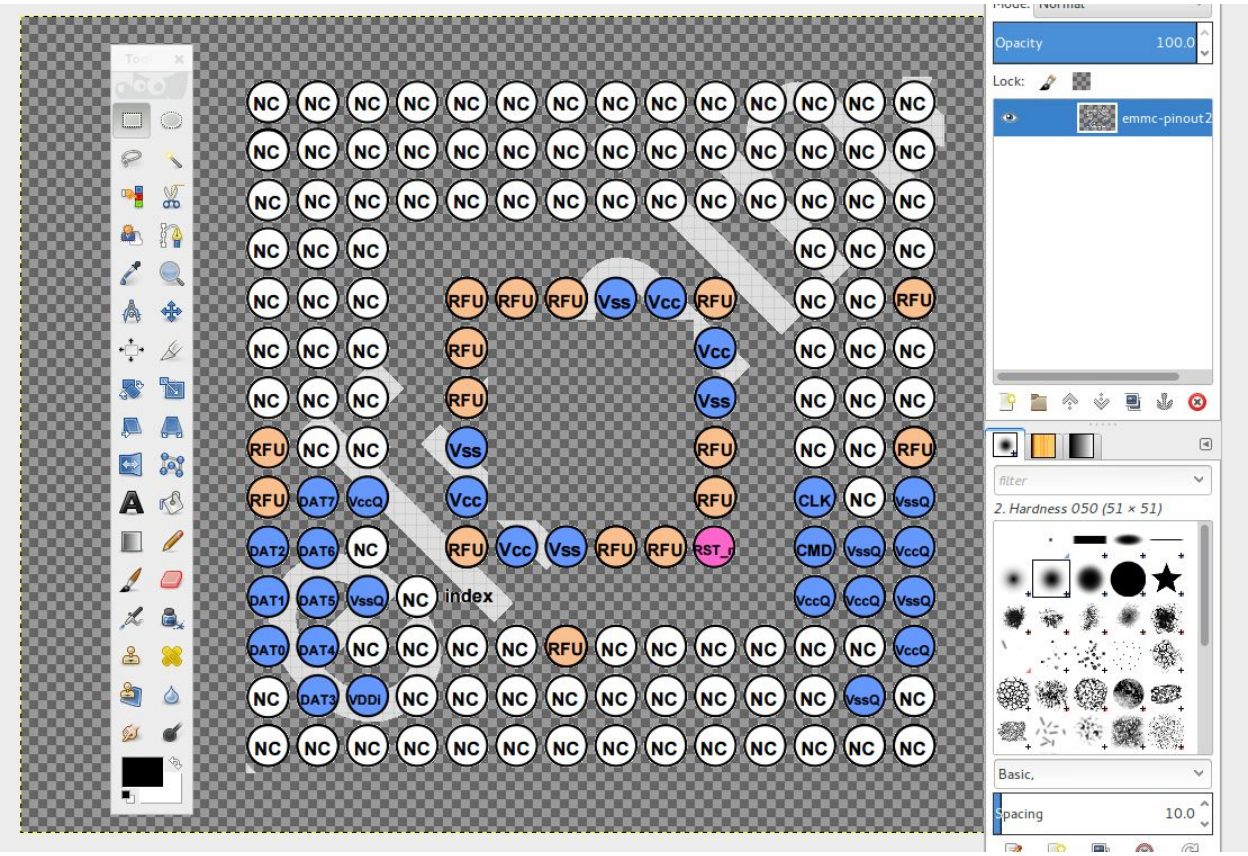
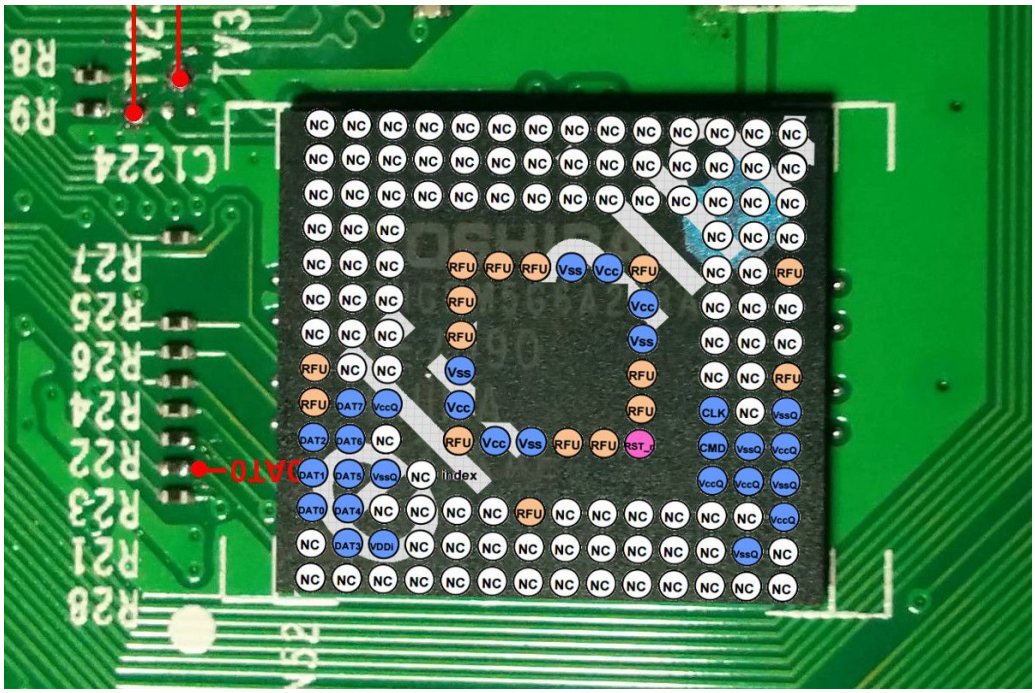


Figure 7: Datasheet aligned over the Toshiba eMMC Flash - THGBM5G6A2JBAIR



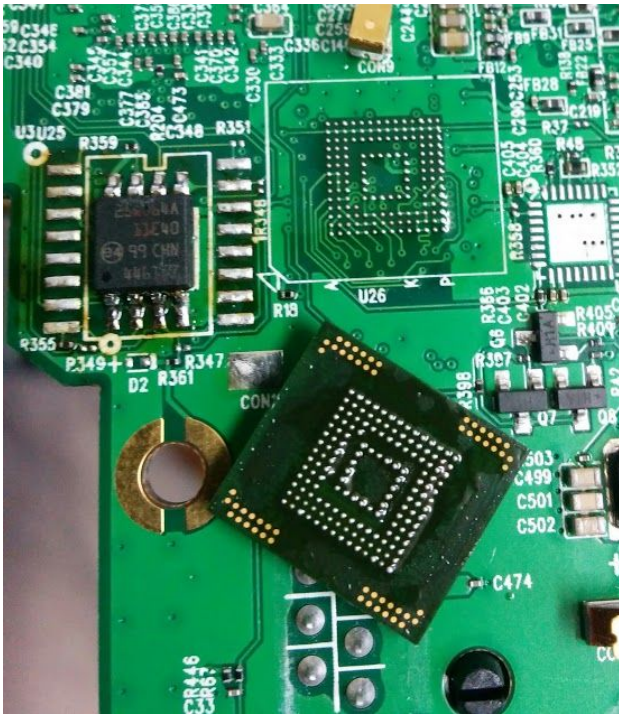
The above image (**Figure 7)** contains the approximate location of each pad after overlaying the layout diagram provided in the datasheet. This overlay allows us to view the approximate space the pads share with the resistor banks as well as how the traces are likely to lead to specific pads. The above image also indicates that based on the location, it is probable that the resistors are connected to the specific DATA lines on the left side of the eMMC chip. Finally, based off the silk screening (R21-R28), it can be concluded that R21 is most likely *DAT0* because it has the lowest designator ID number and may have been the first data line added during design. *CMD* and *CLK* can be more difficult to visually identify. In the above image (**Figure 7)**, there are no traces on the right side of the chip where the *CMD* and *CLK* pads are. However, knowing that the SoC is to the left of the chip (**Figure 4**), it can be hypothesized that they may be connected to R8 and R9 in the upper left of the image above (**Figure 7**).

Finding pinouts of eMMC by desoldering the chip

In some cases, the pinout of the chip cannot be determined utilizing the previously described method. A more invasive solution is to detach the eMMC chip from the board by desoldering it. This process requires a very steady hand, a heat gun or a hot air rework station, a good set of tweezers, and soldering flux. To begin, add flux generously to the area around the eMMC chip. Then begin to slightly warm the board up with hot air by applying heat at a distance in small circles avoiding focusing on any one area. Now, slowly tighten the circular motions and focus the heat on the eMMC chip. Once the solder of the nearby components begins to shine, the melting point of the solder has been reached and it is time to remove the chip. Using the tweezers, bump the eMMC chip very lightly on the side to see if it moves. If it doesn't move, carefully heat the chip up further and try again. If the chip does move easily when bumped, then the eMMC can be removed. Take the tweezers and lift the chip straight up off the board.

If completed successfully (practice with a low cost PCB first), the board will look like **Figure 8** with all of the solder balls still attached. If the process is unsuccessful, the chip will need to be reballed before it can be reattached. Best practices dictate that when you remove a device you must reball it; however, many times you can still get your device to work without reballing. We've found that the best method of reconnecting the chip is to use solder paste and a BGA stencil, both of which can be found on eBay or Amazon.

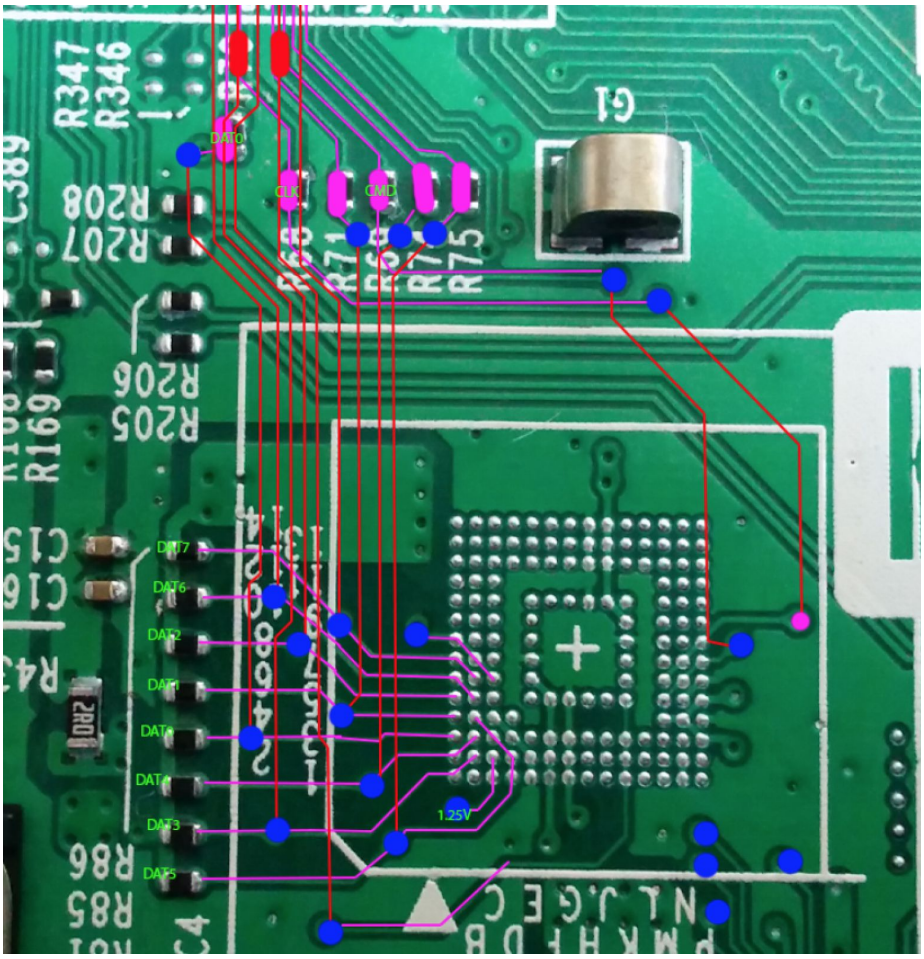
Figure 8: eMMC chip desoldered and removed from a target device



Now that the chip is detached, the pins can be matched to test points or components on the board to establish an in-circuit pinout, or the chip can be flipped over and soldered to directly also known as dead-bugging. If the chip is dead-bugged for reading, it will then need to be reballed before being reattached.

The simplest way to have continued access to the eMMC flash and still operate the device is to establish alternate communication points. This requires tracing out all the eMMC pins and finding alternate points on the board to attach to a reader. Establishing a pinout is a better option than dead-bugging because the chip will not need to reballed if any software-based mistakes are made. The chip can simply be reprogrammed once it is reattached to the board. For an in-circuit approach, the pins be can traced visually or using multimeter in continuity mode. In most cases, It is generally less time consuming to take a picture of both sides of the board and use a photo editor like GIMP or Photoshop to trace out all the pins as seen below (**Figure 9**).

Figure 9: Key eMMC Lines traced out to alternate points



After an alternate pinout has been established, the eMMC chip can be reattached. This process can be difficult and may require a few failed attempts before being perfected. This starts with cleaning the board and eMMC chip with isopropyl alcohol. Then begin to apply a thin layer of flux onto the board. After applying the flux, carefully align the chip onto the board making sure that the chip is oriented correctly and centered within the silkscreen outline. The board is now ready to be reflowed. Warm the board to prevent warpage and then focus heat onto the chip similar to the process used for removing the eMMC. Ensure that the chip is heated evenly by moving the hot air in a circular motion. After the appropriate temperature has been reached, the chip will lower onto the board and flux will begin to ooze out around the sides. The heating process is now complete and the board can cool down naturally. Once the board is room temperature, power the device up and check to see if reattaching was successful (**Figure 10)**.

Figure 10: eMMC soldered back in place

Finding pinouts via Signal Identification with a Scope

In some cases, a reverse engineer may be uneasy with the idea of removing a BGA chip from a board and will not be able to easily determine the alternate pinout from the positioning as detailed above. In these situations, one of two paths can be taken. One involves brute-forcing the pinout by first verifying proper voltage levels with a multimeter, then attaching to the pin for verification. The alternative path involves using an oscilloscope or logic analyzer to view the signal on the lines. Although bruteforcing does work, the more elegant approach is to use a scope.

Most eMMC flash chips support multiple modes of reading or writing data. These modes offer operations that can be 1 to 8 bits and that use 1 to 8 data lines. The amount of data lines used for reading or writing will increase or decrease the speed of data reads and writes.

Pin Name	Decription
VCC/VDD	eMMC Power (typically 3.3v)
VCCQ/VDDQ	Power for IO bus (typically 1.8v or 3.3v)
GND	Ground
CMD	Command
CLK	Clock line
DAT0-DAT7	Data lines (DAT0, DAT1, DAT2, etc.)

The pins required for successful communication can be seen in the table above. Although there are multiple data pins listed above, only one is required for reading. Although Vcc, VccQ and Gnd are easy to find with a multimeter, using an oscilloscope to determine pin type is sometimes needed. To determine what an eMMC pin is with an oscilloscope, attach probes to possible points while the device is accessing the eMMC. Probe pads that may be attached to the eMMC bus based on initial research. The image below (**Figure 11**), shows an example of what *CMD*, *CLK*, and *DAT0* look like. The *CLK* signal should look very periodic, as can be seen better in **Figure 12**. The *CMD* line will only burst a short bit of data at the beginning, and will appear less busy than data lines as seen in **Figure 13**. Locating the *DAT0* line is where the process becomes a little more difficult. With up to 8 different data lines (*DAT0-7*), finding the first one can be challenging. That is why it's important to utilize other clues to determine the correct data line. Either utilize the location of the pads, or silkscreening to help determine which pad is *DAT0*.

Figure 11 - Scope capture of DAT0, CLK, and CMD

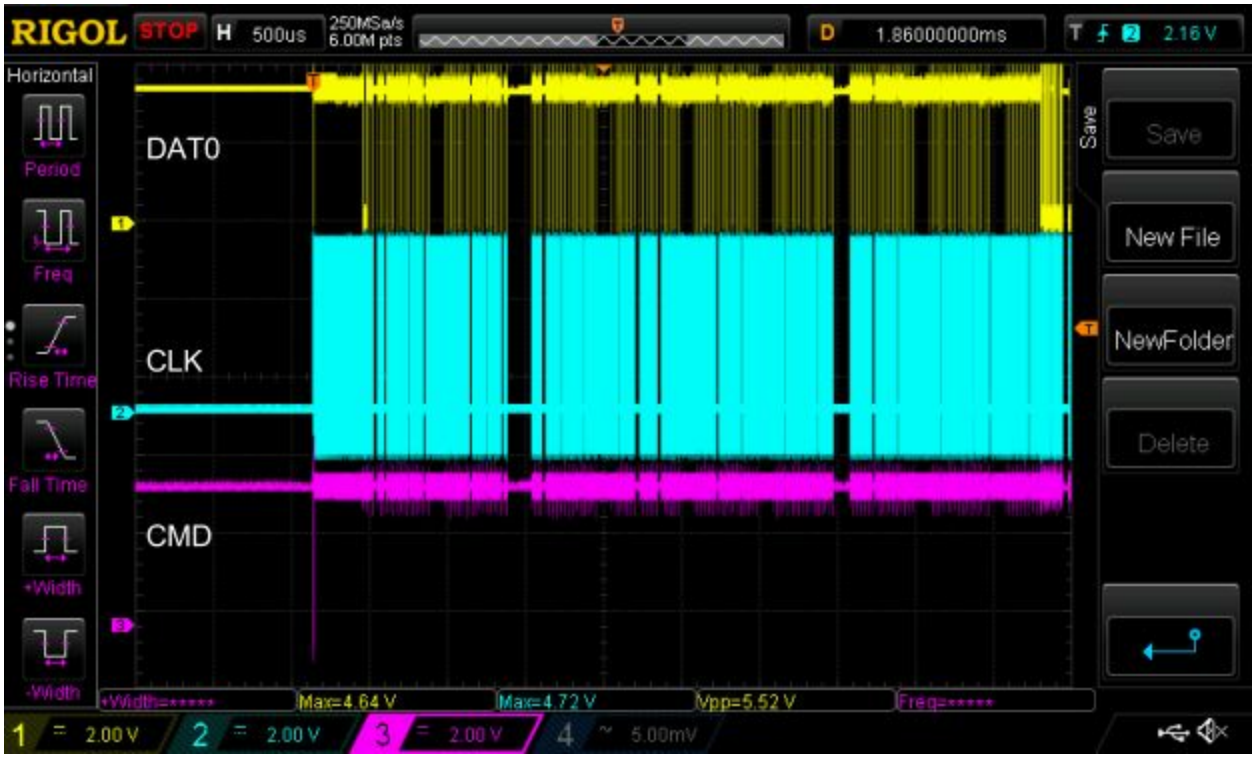


Figure 12 - Scope capture of eMMC clock signal

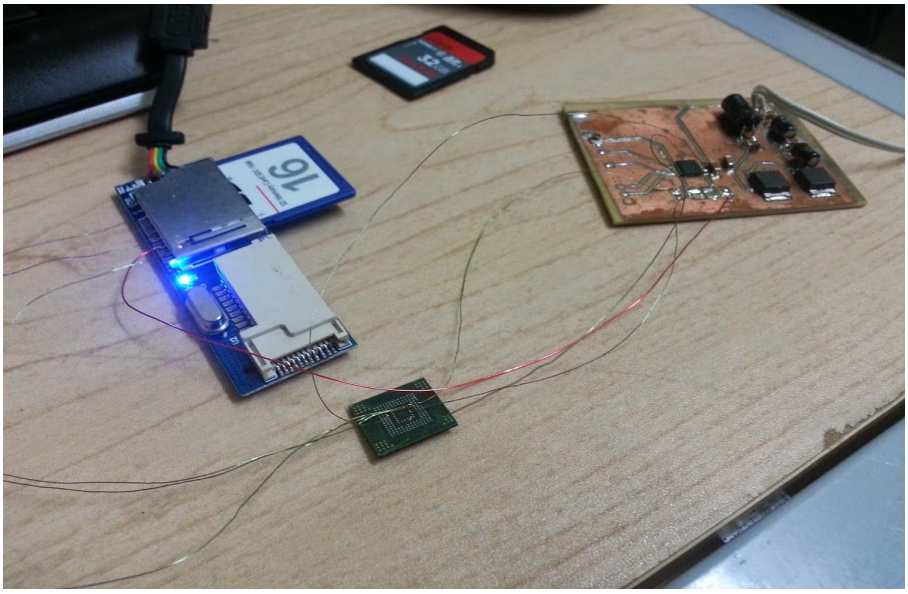
Figure 13 - Scope capture of CMD signal

At this point, all of the minimum required eMMC lines have been identified. The next few sections will discuss the ways to connect to the physical chip as well as how to hook up to an SD card reader.

Connecting to eMMC flash

There are three different approaches when connecting to a reader, each requiring various levels of skill and approach. The first method involves powering the system normally with the eMMC chip attached to the board to read and write from flash. This method requires fewer pins (4 pins) for communication but can have issues which will be discussed later. The second method involves applying power externally to the device with the eMMC attached and attempting to read and write from the flash. This requires more pins than the previous method but, suffers from similar problems. Finally, the third method involves attaching to the eMMC chip directly while it is detached from the board to read and write from the flash. This method is known as “dead-bugging” the chip because of how the IC is placed on its back and soldered to. Dead-bugging provides direct access to the eMMC and therefore removes any complications that can be introduced from the surrounding circuits. However, dead-bugging can require the most skill as it requires the complete removal of the eMMC flash as well soldering to the underside of the IC and reballing to repair the BGA pads (**Figure 14**)**.** Although this can be accomplished with a hot air gun and soldering iron, any attempt to put the eMMC back without BGA reflow work (AKA reballing) will generally result in failure. Each method introduces its own complications but with enough practice, the number of damaged devices can be significantly reduced.

Figure 14: eMMC flash chip removed, and hooked up like a dead bug



Leveraging SD to read/write eMMC

Due to the similarities between SD and MMC, an SD reader can be used to read eMMC flash. The SD card protocol was designed to be a superset of the MMC protocol[12] with the caveat being that SD cards only utilize 4 data pins. Therefore, the maximum that an SD Card reader can utilize is 4-bit mode. For our usage, SD readers that support 1-bit mode are prefered, allowing for the minimum possible number of pins attached. While this can result in slower reading and writing, the benefit of simplicity outweighs the slow speed of operation. Unfortunately, some readers do not support 1-bit operation. To verify if a reader supports 1-bit operation, a piece of tape can be placed over the DAT1 to DAT3 pads on an SD card. The card can then be inserted into the reader. This will force the reader to communicate with the SD card over 1-bit mode, testing if the reader supports 1-bit operation. In our experience, the reader that has performed the best is the Transcend RDF5 USB 3.0 SD; this device is cheap and supports 1-bit operation.

After selecting an SD Card reader, the next step is to connect the eMMC to the reader. There are multiple ways of connecting including soldering directly to the pads on the reader or using an older SD card skeleton as a makeshift adapter. However, the more versatile option involves using a breakout board. A breakout board allows for the breaking out of the pins of an SD card to pin headers which can then be soldered to the device (**Figure 15**). After connecting to a breakout board and inserting the breakout into an SD card reader, the eMMC will show as a block device. If the operating system supports the underlying filesystem, the partitions may be mounted resulting in an experience similar to that of inserting an SD card.

Figure 15: eMMC flash connected to SD card breakout, and to SD card reader

[12] http://www.ti.com/lit/an/spraao7/spraao7.pdf

Problems with reading/dumping

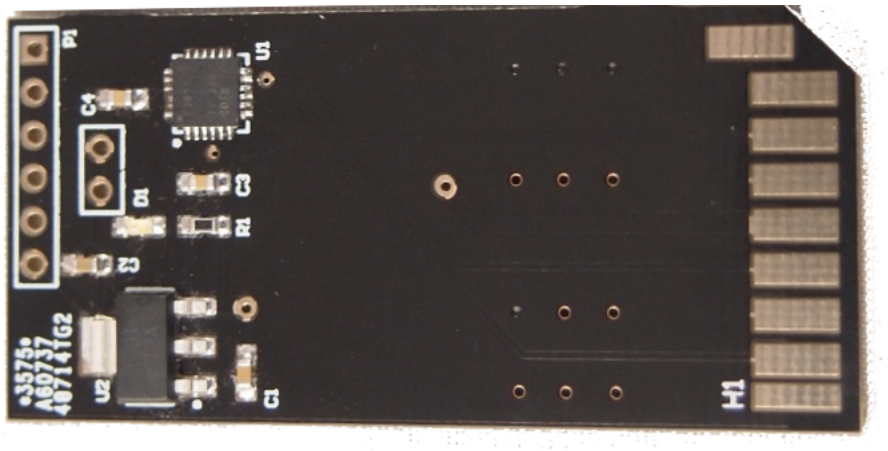
There are a number of problems that can arise while attempting to communicate with an eMMC device. These can vary depending on the method of attachment used and can consist of complications with the chosen communication voltage levels, issues providing proper power to the eMMC, and issues related to the board design (when doing in-circuit programming).

There are a number of issues that need to be considered when trying to program an eMMC still attached to a board. For example, when feeding an eMMC chip power, other devices in the same power rail will also receive power. If the CPU is on the same power rail, it may boot up and attempt to access the eMMC flash while the reader is also accessing the flash. This will lead to corrupted data or an undetected chip. This issue can be solved by finding the CPU reset signal and pulling it high, by disabling the CPU clock source (e.g. remove the CPU crystal oscillator), freeing the CPU from the power rail (e.g. cutting traces), or by isolating the eMMC signal from the CPU. Depending on the design of the device, many of the bus signals may be routed to a resistor prior to going into the CPU. If this is the case, the resistor can be removed or the trace to the resistor cut, which will isolate the CPU from the eMMC allowing for reading or writing.

Another issue that can occur is due to an incompatibility with an SD card reader and an eMMC flash chip's voltage level. Many lower power devices utilize a 1.8v logic level; however, most SD card readers only work with the 3.3v logic level. The logic level of the eMMC chip is set by the *VCCIO* pin, or by *VCCQ*/*VDDQ*. The logic level can be determined by checking the eMMC's datasheet. If the eMMC is being accessed in-circuit, the logic level cannot be adjusted without the risk of damaging or destroying the other chips that share the same power rail. For example, if *VCCIO* is provided with 3.3v on a device where the *VCCIO* line of the eMMC is on the same rail as the CPU and DRAM is expecting 1.8v, the maximum rating of the components may be exceeded causing permanent damage. To solve this problem, a voltage translator that can translate the 1.8v logic level to 3.3v at a fast enough rate is required. By utilizing a voltage translator, a standard 3.3v SD card reader can interface with a low voltage eMMC. Fortunately, Texas Instrument makes an SDIO

voltage translator[13] that can be used for this application. The board we created[14] to perform the necessary voltage translation utilizing the Texas Instruments chip can be found below (**Figure 16**).

Figure 16 - exploitee.rs Low Voltage eMMC Adapter



eMMC boot partitions

A little known fact is that many eMMC flash devices also feature boot partitions that can be read from after initializing the device with a proper series of commands[15]. With a normal SD- card reader, this mode is inaccessible as these commands cannot be sent to the eMMC chip. However, on some laptops there is an SDIO controller that does allow to directly interface with the eMMC chip. This allows for commands to be sent directly to the flash controller, including the ones needed to access the eMMC boot partitions. Support for these boot partitions can already be found in the mainline Linux kernel. Therefore, if using a supported SDIO controller under Linux, the devices will be found within the "/dev" folder with a filename similar to "/dev/mmcblkXboot0" and "/dev/mmcblkXboot1".

If the machine being used does not contain a built-in SD card reader, an SDIO PCIe bridge can be used. One example of a tested SDIO PCIe bridge is the RICOH R5U230 which is supported by the Linux kernel but can cost $150 USD. However, a more cost effective solution can be accomplished by using a BeagleBone Black Development board. The BeagleBone Black is a small microcomputer similar to the Raspberry Pi but, features two SDIO busses, one for the internal eMMC, and another for the SD card (**Figure 17)**. By wiring the eMMC chip to the SD card slot on the BeagleBone Black (**Figure 18**), both boot partitions are able to be read (**Figure 19**). Without modification, the Linux kernel drivers used on the BeagleBone Black do not properly communicate in 1-bit mode (for optimal use). Luckily, we have developed a patch for the kernel drivers which will be available on our website[16] to fix the issue.

Figure 17 - BeagleBone Black connected in-circuit to eMMC flash

[13] http://www.ti.com/lit/ds/symlink/txs02612.pdf
[14] https://www.exploitee.rs/index.php/Exploitee.rs_Low_Voltage_e-MMC_Adapter
[15] https://www.micron.com/~/media/documents/products/technical-note/nand-flash/tn2918.pdf
[16] http://www.exploitee.rs/

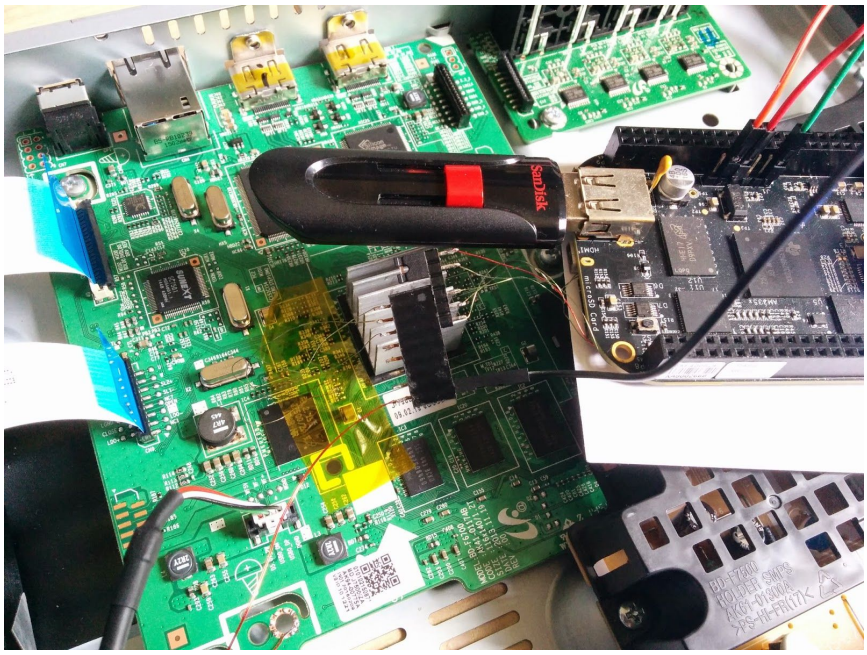


Figure 18: BeagleBone Black SD card breakout

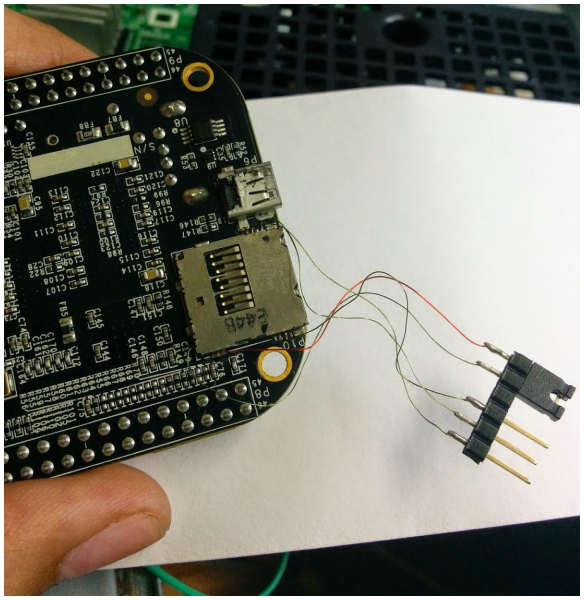
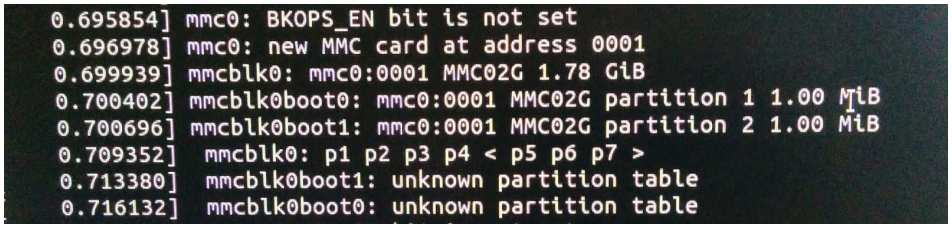


Figure 19 - BeagleBone Black dmesg



The Future of eMMC

The use of eMMC flash is starting to decline as speed becomes a concern, with more manufacturers using new storage technologies such as Universal Flash Storage (UFS). However if the history of NAND flash is any indication to the future of eMMC, eMMC chips will continue to be used and be in designs for years to come.

Conclusion

eMMC is a popular form of embedded storage which allows for easier use and manipulation than that of its NAND counterparts. eMMC flash can be manipulated using cheap commonly found hardware with the number of connections required as few as 4. This makes it an excellent target for reverse engineers and creates a low barrier of entry for researchers looking to utilize it in reverse engineering.