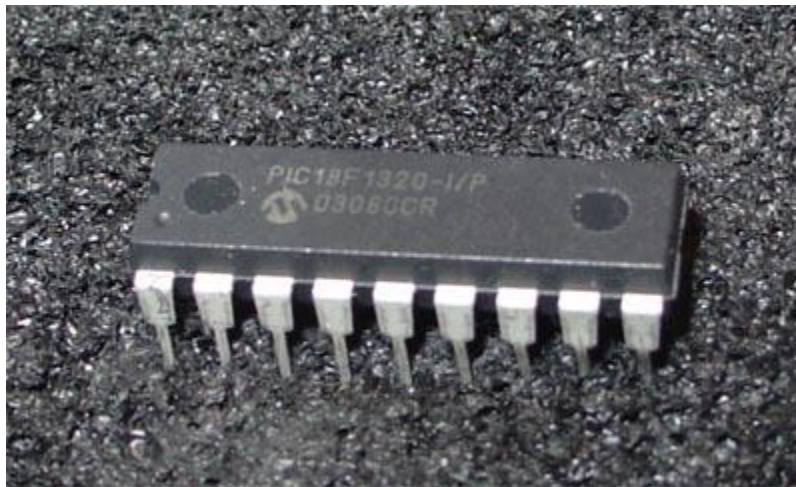


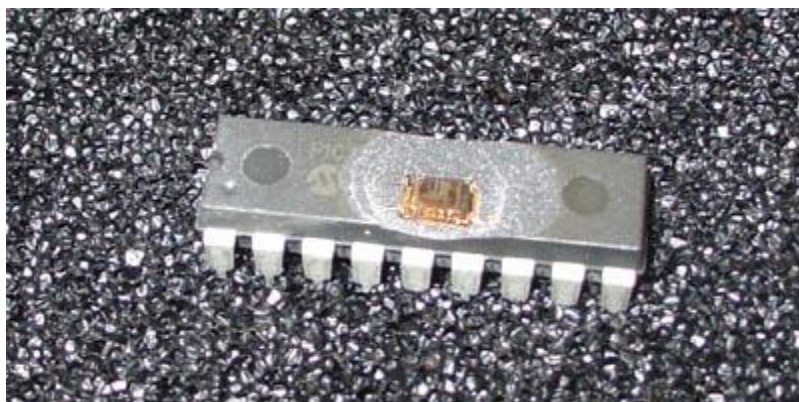
Hacking the PIC 18F1320

I thought it would be fun to try out some of the hacking techniques I had heard about on the PIC series of microcontrollers. PIC microcontrollers typically come with a set of “configuration fuses” that typically include settings to prevent the modification or readback of certain regions of memory. Quite often, a legitimate need arises to read out the contents of a secured, programmed PIC. A typical example is a company that has lost the documentation or the personnel that originally created the codes for a secured PIC. This often happens when a company needs to revise or upgrade a legacy line of products.

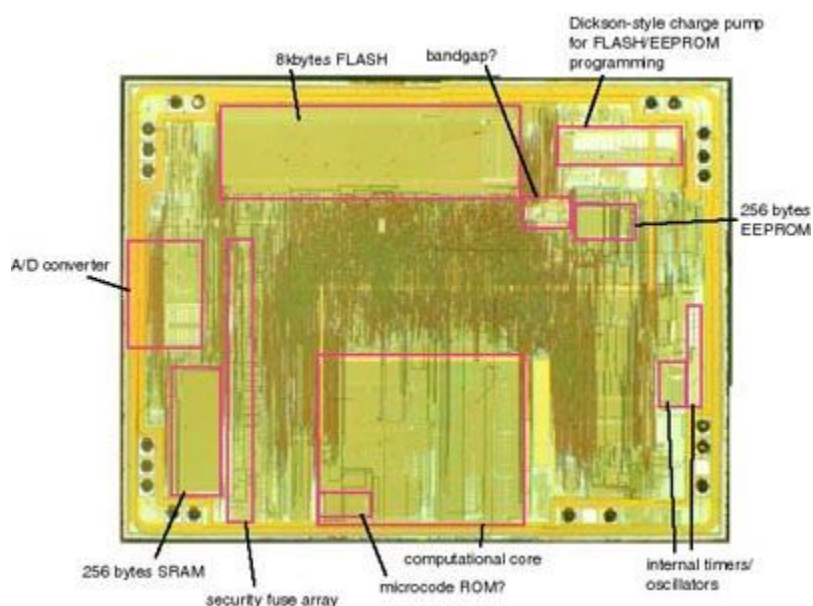
I scored four PIC18F1320's from [Joe's](#) stash (it's nice having lots of fellow hackers in San Diego) and started stripping them down. This is what a PIC18F1320 looks like in its native state:



The first thing to do is to take the top off so you can see the silicon within. While there are many homebrew techniques for doing this, they typically involve the application of fuming Nitric or Sulfuric acid. Neither of these are compounds that you would want to have around your home, nor are they easy to obtain since Nitric acid in particular is an important compound for explosives fabrication. I've found that the easiest and most reliable way to do this is to just send the part to a failure analysis lab, such as [MEFAS](#), and for about \$50 and a two-day wait, you can have a decapped part in your hands. For this project, I decapped three total parts; two were functionally decapped (silicon revealed with device still in lead frame, fully functional), and the last one was fully decapsulated so that it was just a bare silicon die completely absent a package. The last die was fully decapsulated because my inspection microscope has a very short working distance at the highest magnifications.

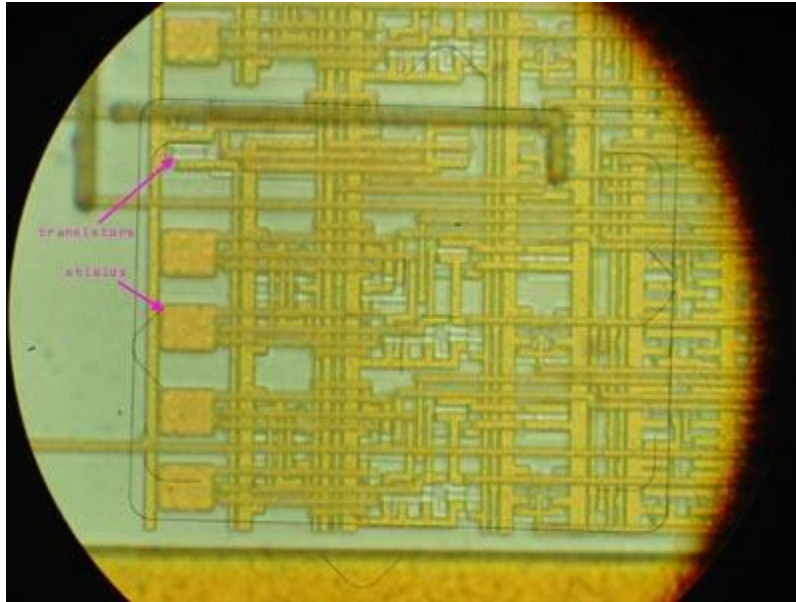


A little sweeping around the die revealed several prominent features, as shown below:



The above annotations are my best guesses at what various structures do; I could be wrong, and if you happen to have anything to share, please do post a note!

One set of structures grabbed my attention immediately: a set of metal shields over transistors, following a regular pattern that had about the right number of devices to account for all the security bits. Full metal shields covering a device is very rare in silicon, and like a big X marking the spot, it draws attention to itself as holding something very important.

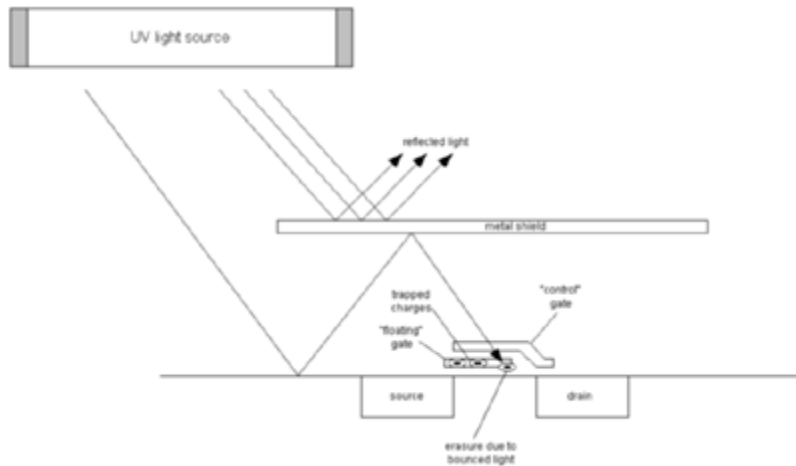


Let's think a little bit more about this metal shield. What is the significance? First, let's review some interesting facts about FLASH technology (the type of memory technology used in this PIC device to store the security fuse information). FLASH technology uses a floating-gate transistor structure very similar to that found in the old UV-eraseable EPROM technologies (remember the days of the ceramic packaged 2716's with quartz windows?). Data is stored in both FLASH and UV-EPROM devices by causing electrons to tunnel into the floating gate, where the electrons will remain for decades. The extra electrons residing in the floating gate creates a measurable offset in the characteristics of the storage transistor. The difference is that FLASH memory can withdraw the stored electrons (erase the device) using only electrical pulses, whereas a UV-EPROM requires energetic photons to knock the electrons out of the floating gate. The UV light required to accomplish this is typically on a wavelength of around 250 nm. This wavelength of UV is a bit difficult to manipulate, since it requires expensive quartz optics to manipulate without excessive loss.

Here's the important observation that comes out of these facts: FLASH devices can usually *also* be erased using UV light since they have a similar transistor structure to UV-EPROM devices. The encapsulation around a FLASH device normally prevents any UV light from effectively reaching the die, but since the PIC devices had the plastic around them removed, I can now attempt to apply UV light to see what happens.

I performed a simple experiment where I programmed the PIC device with a ramping pattern (0x00->0xFF over and over again) and then tossed it in my UV-EPROM eraser for the length of oh, about a good long shower and some email checking. When I took the device out of the eraser, I found that indeed the FLASH memory was blanked to it's normal all 1's state, and that the security fuses were unaffected. Significantly, if I did not bake the PIC device for long enough, I would get odd readings out of the array, such as all 0's, a phenomenon that I do not understand. I'm supposing it could be due to some effect involving incomplete erasure and the reference bitlines used to drive the reference leg of the sense amps on the FLASH array. Also note that the UV light works just as well on the EEPROM array.

Clearly, the metal shields over the security fuses were provisioned to thwart attempts to selectively erase the security fuses while leaving the FLASH memory array unaffected.



The picture above illustrates the problem I have (and its solution) (click on the image for a larger, clearer version). In order for the FLASH memory transistor to be erased, high-intensity UV light must strike the floating gate. The metal shield effectively reflects all of the incident light.

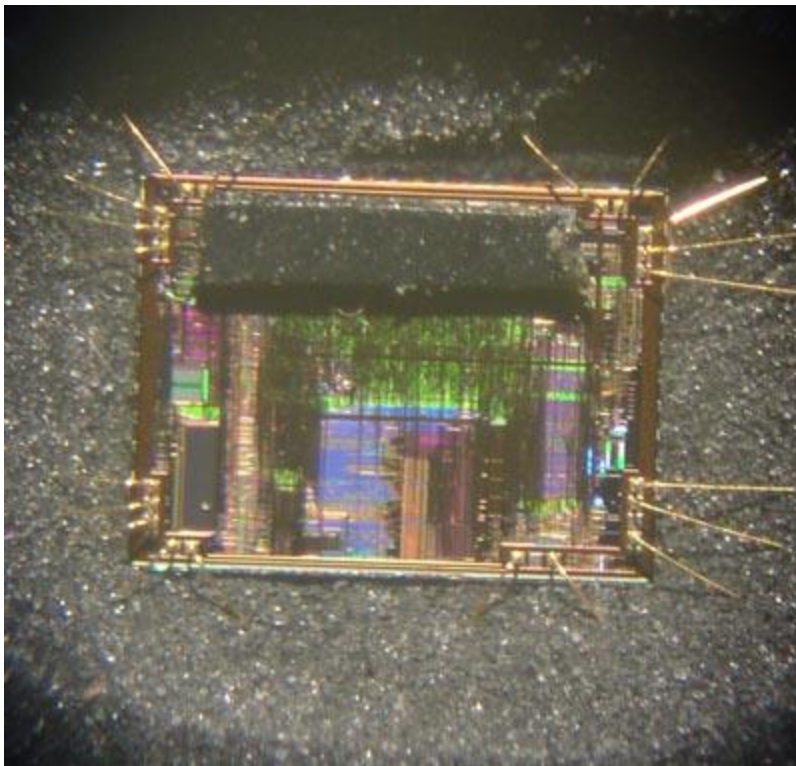
However, due to the optical index mismatch between the oxide and the silicon interfaces, light at certain angles will reflect off of the silicon surface. In order to witness an example of this reflective effect, jump in a swimming pool and submerge your head and look up at the water-air interface. You will note that the water looks highly reflective at an oblique angle. This is due to the index mismatch between water and air causing total internal reflection of light.

This reflection can be used to cause the UV light to bounce up and the metal shield, and bounce back onto the floating gate. Thus, by angling the PIC inside the ROM eraser, I can get enough light to bounce into the FLASH memory transistor region and cause erasure. After a couple of attempts, I developed a technique that seems to work relatively well.



Picture of the chip inside the UV eraser (note blue halo around chip due to active UV lamp). The chip is stuck into the antistatic foam at an angle.

This still doesn't prevent me from erasing the desired data in the program FLASH space. In order to prevent erasure of this data, a hard-mask is formed using a very carefully cut piece of electrical tape that was stuck onto the surface of the die using a steady hand, two tweezers, and a microscope. The electrical tape effectively blocks the UV light from directly hitting the FLASH code memory regions, and it also somewhat absorbs light bounced back from the silicon substrate.



Here's a picture of the die in package with electrical tape over the FLASH rom array.

The screenshot shows a Windows XP desktop with the 'Task Scheduler' window open. The 'Task Scheduler' window is displaying a list of tasks. The 'Task Name' column is highlighted, and the task 'Task Scheduler' is selected. The 'Task Scheduler' window is also open, showing the 'Task Scheduler' window. The 'Task Scheduler' window is also open, showing the 'Task Scheduler' window.

And thus one can selective erase portions of a PIC's contents. Fun!

rmashmoul