published on June 18, 2020 in Tech Tips & Tricks Creations

Low-cost attacks on STM8 readout protection

As part of my HC-12 hacking project I needed to acquire the firmware of an STM8 microcontroller that had readout protection enabled.

I was long-time intrigued by fault-injection attacks, most recently triggered by this 35C3 Talk on PS2 Vita Hacking which used voltage glitching to overcome protection measures.

From the STM8 reference manual:

> **4.5.1: Readout protection**
>
> Readout protection is selected by programming the ROP option byte to 0xAA. When readout protection is enabled, reading or modifying the Flash program memory and DATA area [using the SWIM debug interface] is forbidden.
>
> Even if no protection can be considered as totally unbreakable, the readout feature provides a very high level of protection for a general purpose microcontroller.

This looks like a great target for voltage glitching. Thus: challenge accepted! :-)

Glitch Hardware

While others reported doing similar projects with an Arduino, I thought I need the timing precision of an FPGA (and wanted to try writing hardware anyways). Not wanting to spend too much on this, I went with an EPM240, which is a really inexpensive CPLD devboard (~5€) capable of toggling IO Pins at up to 50MHz, which I hoped would be sufficient.

Annoyingly it wasn't easily usable from Linux, but I eventually managed.

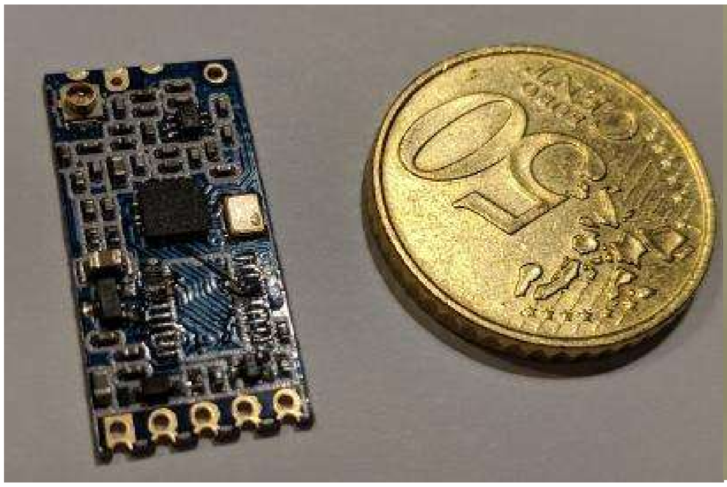
Writing the VHDL was relatively straightforward, but who would have thought that hardware is weird and debugging hardware is a pain (printf debugging is nothing compared to GPIO toggling).

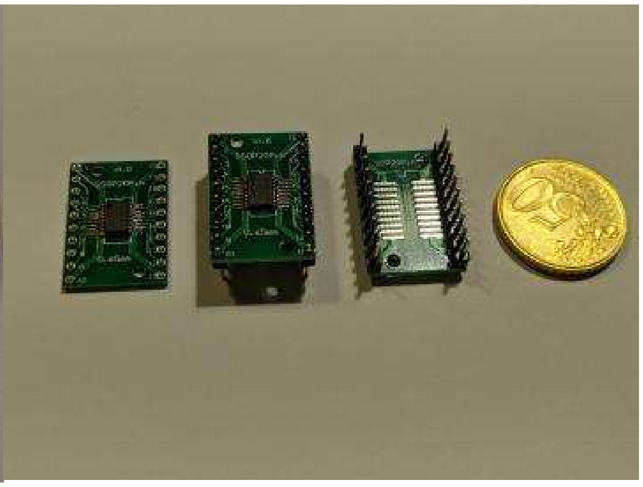
Since the compiling and reflashing part is annoying and slow, I wrote the software to receive a configuration via an UART implementation writing central params to registers:

- trigger polarity
- trigger count
- glitch polarity
- trigger-to-glitch delay
- glitch duration

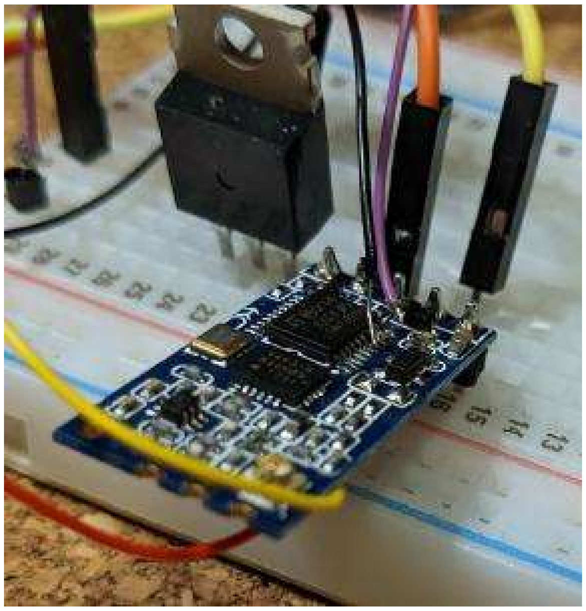
Glitching Setup

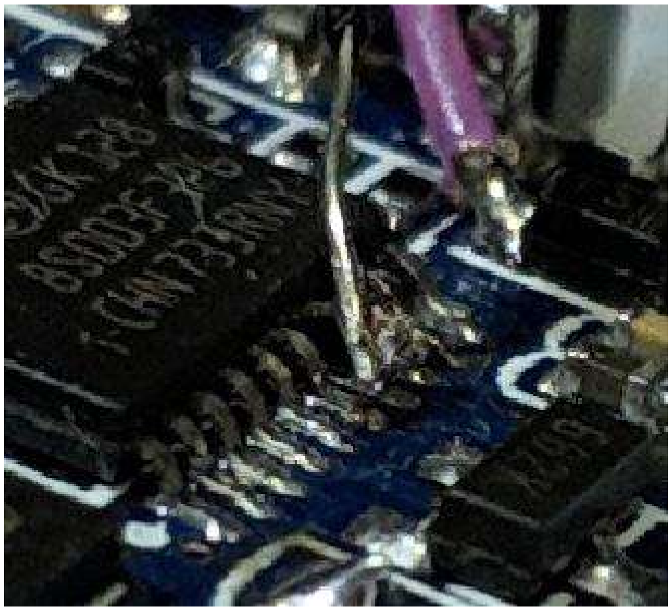
The HC-12 board has a bunch of capacitors which would defeat my glitching goals, so I first tried to unsolder the HC-12 STM8 to an isolated board, but may have broken it in the process (so that's not recommended).


Unsoldering the MCU

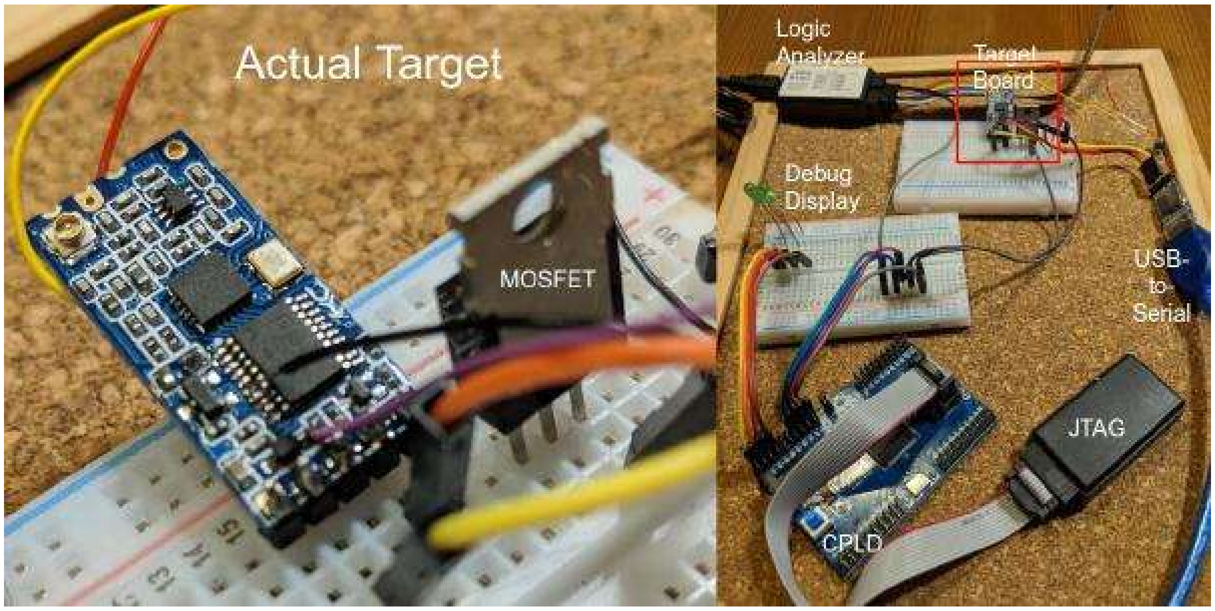

Putting it onto a prototype adapter

My next attempt was to detach the GND pin from the socket and attach a new wire.


GND Pin detached (overview)


GND Pin detached (zoomed)

Here’s what the whole bench looked like:



Glitching Setup

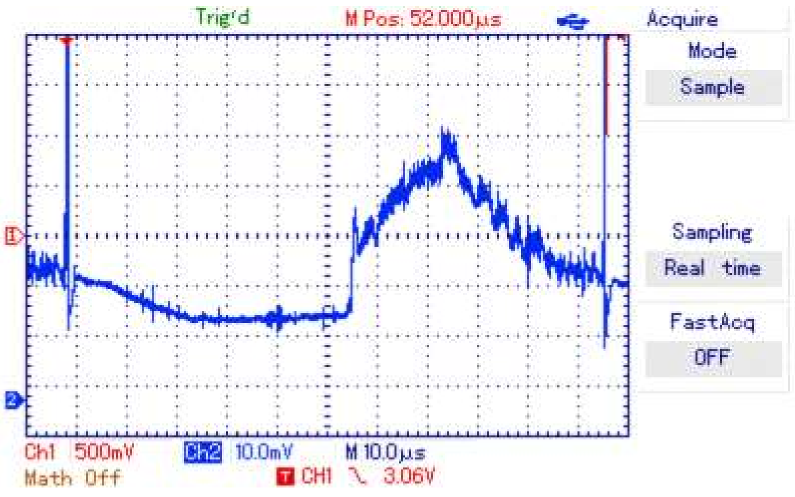
The total of the equipment used is <20€.

My setup seemed to work and I was able to cause seemingly undefined behavior, but I wasn’t able to glitch the option byte loading despite bruteforcing time offset and glitch duration for many nights. I eventually realized that I'm more likely glitching the UART transmission part or my USB-to-serial converter.

Differential power analysis

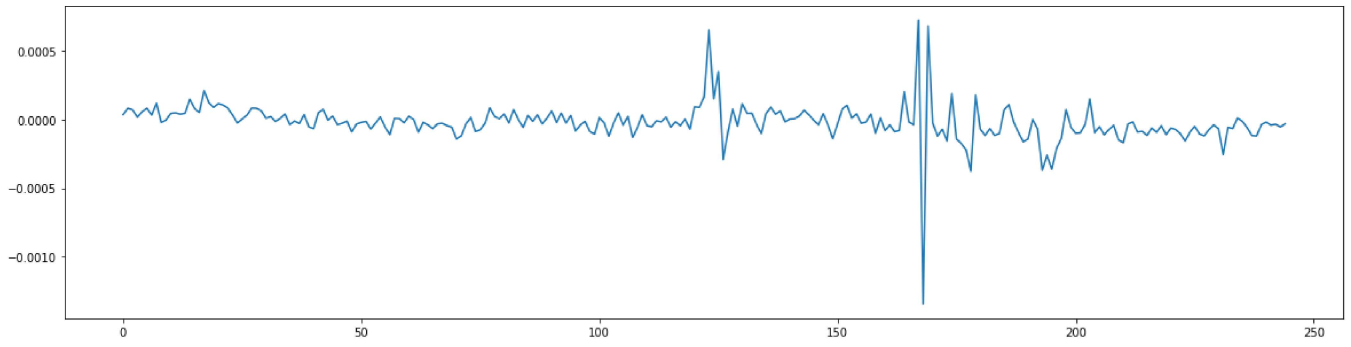
One of the problems I was facing was that I didn't really know well when to glitch and trying out all the options. So I devised a power analysis setup on the assumption that the microcontroller consumes more (or less) power when loading option bytes with different amounts of bits set to one.

I borrowed an UT2052 oscilloscope and had to write some linux drivers for it to be able to automate trace retrieval. Then I prepped an STM8 with the readout protection byte set to 0x00, 0xFF and 0xAA. Using the oscilloscope I recorded hundreds of power traces around the reset phase.

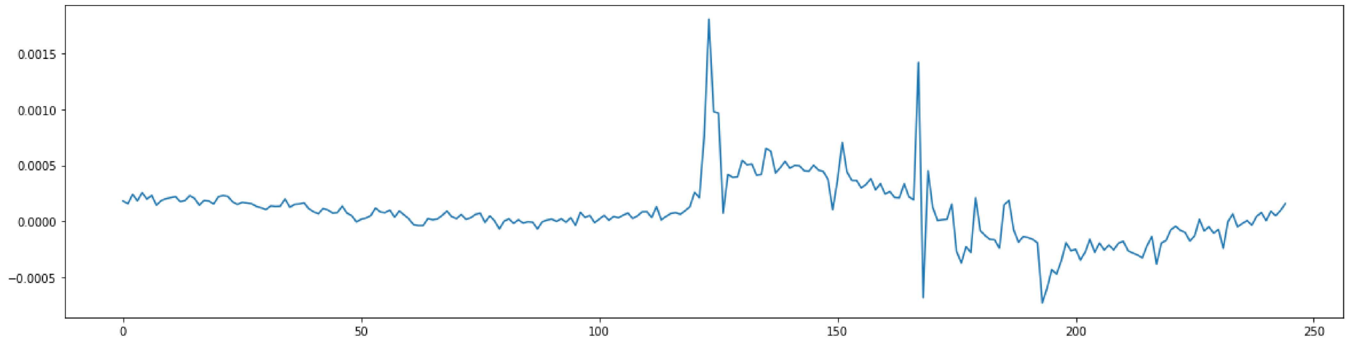


oscilloscope view of power traces around the reset phase

I aligned the traces, averaged them, did consistency checks and removed too noisy ones. The noise graph illustrates the amount of noise within a trace set. Any significant diffs occuring at these points are likely more due to noise than due to different currents during computation of different values.



With that in mind I looked at the difference between the averages of different trace sets. The oscilloscope has a really bad resolution (records <250 bytes @8bits) and thus the SNR is quite bad. Accordingly there are strong artifacts, but there’s also a visible difference. The most promising is around position 150: negative correlation spike followed by a positive correlation spike without a significant event in the noise graph.



Having such candidates clearly helped to refine the search space, even if just saying it’s safe to ignore the first 50µs.

Glitching with success

Trying again with different mosfets and a refined search space I not only tried glitching the VCC and GND pins, but also the exposed voltage regulator PINs, since the datasheet already mentions:

> **10.3.1 VCAP external capacitor**
>
> The stabilization for the main regulator is achieved by connecting an external capacitor CEXT to the VCAP pin.

In hindsight this is a really obvious target, but I was very surprised when my attempts worked on first try within seconds of bruteforcing. Using the now-enabled SWIM debug interface I could simply dump the firmware.

This means: **STM8S readout protection is broken** (though not really a surprise there).

And if you need to read out the memory of a readout-protected STM8 chip, let me know!

53 Comments - *powered by utteranc.es*

Prehistoricman commented on Jul 5, 2020

No way! I just saw this on Hackaday and I was doing a very similar project on a power supply's front panel. I got very similar results to you: glitching on Vcc (even with capacitors removed) produced glitches but not strongly enough to skip the bootloader check. I did spot the VCAP pin and played around with the capacitor value, which did change the behaviour of glitches, but still no success.

Did your STM8 have SWIM completely disabled? Mine is an STM8S, which can't have SWIM disabled. Curiously, combined with read-out protection, that means that only the bootloader and RAM could be read out. I used SWIM to get timings down (as I could read out the program counter) and similarly found that 45-70µs after reset is when execution starts.

SWIM was actually really nice for this because it would freeze the processor immediately after the last serial bit was sent; you could delay sending that last bit as long as you wished.

You didn't elaborate on your VCAP glitching setup much. For example: timing, success rate, exact method. Was this on purpose?

Arduinos can be timing precise as long as you disable interrupts. They're just like any other microcontroller in this regard. For all my glitching attacks (I did clock glitching on a Mitsubishi MCU), I use an Arduino Due clone that can toggle pins at ~16MHz. Unfortunately this bad boy does have inherent timing issues due to the instruction cache.

rumpeltux commented on Jul 6, 2020 Owner

> I did spot the VCAP pin and played around with the capacitor value, which did change the behaviour of glitches, but still no success.

I directly went for this pin, pulling it to GND IIRC.

> Did your STM8 have SWIM completely disabled? Mine is an STM8S, which can't have SWIM disabled. Curiously, combined with read-out protection, that means that only the bootloader and RAM could be read out. I used SWIM to get timings down (as I could read out the program counter) and similarly found that 45-70µs after reset is when execution starts.

This is also an STM8S, so yes, I could use SWIM (and I did in fact to check if the glitching was successful).
I don't think I was able to dump bootloader or RAM though with that.

> You didn't elaborate on your VCAP glitching setup much. For example: timing, success rate, exact method. Was this on purpose?

This was because I did all of this back in 2018 and just recently found the time (and will) to write it down, so some details have faded my mind. I did have a relatively large search-space of params (delay, glitch duration), randomized it and would very soon have a successful attempt. A good target was for example ~60µs delay with a glitch-time of 1µs. Again: a variety of params produces the right results, so an Arduino (or another STM8) may well be sufficient for this.

Prehistoricman commented on Jul 6, 2020

Interesting. I tried for a few hours yesterday without much success. I found that bringing VCAP to 1V and under produces glitches. However, too severe of a glitch (in duration or voltage drop) causes it to try to reset itself.

~~What is weird though is that the logic state of reset appears to be tied to the GPIO. So if I tie this pin high, the STM8 can't reset itself. I don't know why the designers would do this but it does work in our favour. Did you do anything special with reset?~~

When you got into the bootloader, did you use the UART interface? That's what I've been trying. I can rarely get out of the default program, but then the STM8's UART TXD behaves strangely (almost as if it's opposite polarity) or does nothing at all.

Prehistoricman commented on Jul 12, 2020

Finally got it to work!
Turns out that glitches are more likely to cause a reset when VCC is lower. I was running this board at 3.3V and couldn't get any successful ROP check skips into the bootloader.
I turned up VCC to 3.6V and was able to get in with a glitch at 84μs after reset and a glitch duration of 0.9μs and a VCAP capacitor of 100nF.

Now the even harder task: an IDA processor module for STM8 :(

rumpeltux commented on Jul 12, 2020 Owner

> Finally got it to work!
> Turns out that glitches are more likely to cause a reset when VCC is lower. I was running this board at 3.3V and couldn't get any successful ROP check skips into the bootloader.
> I turned up VCC to 3.6V and was able to get in with a glitch at 84μs after reset and a glitch duration of 0.9μs and a VCAP capacitor of 100nF.

Congratulations! :-)

> Now the even harder task: an IDA processor module for STM8 :(

Fear not, I got you covered (based on glorious work of others): https://github.com/rumpeltux/Stm8Ida

Prehistoricman commented on Jul 13, 2020

Any idea how to compile that for Windows? I've had no luck at making IDA accept any binary created by VS C++ 2019 since it isn't creating the proper file header. Perhaps I'll find the 2015 version and give that a try.
An additional struggle is that the IDA versions are pretty incompatible with each other and these github repos never include compatibility details like that. Actually, I'm astounded that your version

works since it uses both the deprecated inf_set_be() and the 7.3 new shiny jumptable.hpp

rumpeltux commented on Jul 13, 2020 Owner

No idea about Windows. But Linux works for me with ida 7.5.

Prehistoricman commented on Jul 25, 2020

I only used SWIM to figure out the approximate location of the bootloader entry check. I should go in and see if it's accurate. SWIM activation will halt the processor if it's ROP-enabled. You won't be able to resume execution until a power cycle.

In my setup, I used a MAX4619 with 1 switch controlling Vcc and the other two controlling VCAP. That way, the Arduino has control over glitching and being able to issue power cycles.

I'll try to capture a successful glitch on my scope and stuff it in a repo. I found it's quite sensitive to the severity of the glitch (such as minimum voltage and how long it stays at that minimum). I didn't bother checking how wide of parameters are needed for successful glitches.

rumpeltux commented on Jul 28, 2020 Owner

> I'm trying to follow your steps but did you use SWIM? How you managed reset? I tried to force low level on reset pin for 100ms and then wait for 45 - 70us until the voltage glitch of 1us but STM8s halts.

IIRC I did use the reset line (using https://github.com/rumpeltux/esp-stlink which directly allows me to control the reset gpio). A reset without entering SWIM should not leave the CPU stalled.

Prehistoricman commented on Sep 4, 2020

Your trace image is bad. Why so zoomed out and no vertical scale?
Did you short VCC or VCAP? You should short VCAP to ground.
You will have to play with the parameters that you can change: vcc voltage, delay between reset and glitch injection, short resistance, etc. Get the right conditions and you will glitch easily.

Ronerto commented on Nov 7, 2020

I have st-link 2 and
http://www.denontek.com.pk/STM8S003F3-Development-Board-in-pakistan.
it is possible to copy a stm8s003f3p6.

please help a beginner to realize a dream

Ronerto commented on Nov 9, 2020

I give a monetary bonus to those who teach me step by step to read a stm8s003f3 with R.O.P.

Alex037 commented on Mar 12, 2021

Hello,
Alex here. I have stumbled upon this article a week ago. I have been reading similar articles I could find ever since, trying to figure out what the heck is going on here. I do have an issue with an STM8 (stm8s103k3t6c, to be exact). This STM8 is the core of a step-up regulator that can be seen https://www.aliexpress.com/item/32782512138.html . Thankfully, I have a few of them still working, and the idea is to dump the firmware from the working device and transfer it to the blank unit. I cannot locate firmware anywhere officially (tried in dozens of places and manufacturers). I'd appreciate if you could give a bit more details about the procedure (wiring for example). I know that I am still far from realization, I do have some skill with arduino and STMs, but the problem troubles me a lot.
Thanks in advance.

rumpeltux commented on Mar 12, 2021 Owner

Hi Alex, this is too generic of a question. If you're looking for specific details, I'll try to help as time permits.

Alex037 commented on Mar 13, 2021

Many thanks for your reply. I know it's a little generic, trying to wrap my mind about it still. Wiring diagram would be a great start. It should at least show me where to move on to.

timmeh87 commented on Mar 18, 2021

Im in on this effort too, Im trying to get into a ROP protected ST8 chip, and I am practicing on an ST8 dev board with known firmware that I can reflash at will, for fear of breaking the real thing. When I get good at it then I will really attack the real target.

So right now I have a mosfet (AO3418) pulling the VCAP pin down. I chose one with a very low threshold voltage, which I am used to doing, so it drives the line down hard... I think I am just fully resetting the CPU. Im wondering what MOSFET you used was? especially the threshold voltage and on-resistance It looks "big" and Im thinking maybe you used something with like a 5 or 10v threshold voltage so it provides a much softer pull down?

I am glitching with a 180mhz ARM cpu and just polling the reset line for a transition and then injecting a glitch a set amount of time after

I programmed some code on my computer to repeatedly try to read a chip, many hundreds of times per second. The ST-LINK software is open source, It is basically just the stock command line programmer set into a loop. Im not sure if this is the right approach or not. Is a power cycle necessary? It only toggles the reset line.

rumpeltux commented on Mar 22, 2021 Owner

> I think I am just fully resetting the CPU. Im wondering what MOSFET you used was?

Have you tried doing shorter pulses then? Note that with most CPUs you don't get per-cycle control over pin toggles, so make sure to measure that the expected output / duration is matching reality ;)
IIRC I used the IRFZ44 mosfet...

> The ST-LINK software is open source, It is basically just the stock command line programmer set into a loop. Im not sure if this is the right approach or not.

So you do the reset yourself and aren't using the ST-LINK's reset functionality?
I really recommend using esp-stlink, which gives you finegranualar control over almost every aspect.
To detect for success vs failure: If ROP is active reading memory or data will return a fixed value.

> Is a power cycle necessary?

IIRC a clear reset was always sufficient and I didn't need a power cycle.

timmeh87 commented on Mar 26, 2021

Thank you for your response, I think I know how I will update my setup now.

Your mosfet does have a threshold voltage near 4volts, so it was probably turning on very softly. It looks like it would hit a Vds of 1.2volts at very low currents (less than 10 amps) and operate in the linear region. I assume you were using a 5v to drive the gate. My first attempt to turn my own mosfet on more softly was to use the weak pullup resistor to turn it on, rather than driving the gate high with a digital output. This did drive it "softer" in a way but really it was just turning on very slowly and approaching full conduction, so now My VCAP glitch voltage looks kind of like ---\ with a significant delay and then a ramp down. So not really what I would consider Ideal.

So in my next hardware I will use the same mosftet that I was using (I want this to be able to work with any generic mosfet in the saturation region) but I will experiment with inserting various diodes into the glitch path so that the voltage will be clamped to 0.6 or more volts (lets see what weird diodes I can find in my parts box!)

I am also going to get the ESP based ST8 programmer since I have a literal pile of ESP boards kicking around anyways. I will report back

mrx23dot commented on Aug 18, 2021

Is it possible to glitch at 3.3v or it has to be 3.6v?

timmeh87 commented on Aug 19, 2021

This project is sitting in a big pile in the corner and I *swear* ill get to it one day in the winter maybe... Conceptually you can glitch many different cpus and you just need to pull the core voltage down "the right amount" for "the right duration". Someone in this thread had success turning the voltage up which could have influenced the glitch but its not a requirement. If you are copying someone exactly, and specifically using the exact same mosfet, then it might be a good idea to use the same voltage. This hack is highly dependent on the mosfet and how it is driven, since that gives you your final glitch waveform

mrx23dot commented on Aug 19, 2021

I totally missed the 100nF comment, the fitted Vcap is usually 1uF, so I will give it a go.
Maybe that's why I couldn't see the ROP byte difference on Vdd.
Also I'm keeping the 3v3 on, and using the reset pin to restart, that should be more precise.

Is the bootloader readout faster than st-link? st-link takes like 2seconds to try.

A very convenient setup would be 2x 8bit DACs + 50mA OPAs with maxim switch, no need for mosfet.

timmeh87 commented on Aug 19, 2021

I was experimenting with the esp-stlink software made by rumpeltux, you can
get many resets per second and there is better control than using a 'real'
programmer. Never got it going though
...

mrx23dot commented on Aug 20, 2021

Damn, I just realised st-link generates it's own reset signal, overriding my timing.
So the reset signal needs to trigger my delay and glitch.
Is it the rising reset signal you trigger on?

timmeh87 commented on Aug 20, 2021

Yes I believe the standard way to do this hack with any target is to reset
it "normally", but trigger your glitch-hack off of the reset signal that is
then generated, and then try to read it "normally". that way you glitch
module can just be slapped onto any programmer and it just does one job:
time a delay pulse based on another pulse. very simple and robust

…

Prehistoricman commented on Aug 21, 2021

You don't need an ST-link! It's a simple UART bootloader and the commands are public. That's what I did: Arduino Due's UART connected to the device and would query it to detect if the glitch was successful or not. That query is just a few bytes, so it's fast. For flashloading, you will need to write the flashloader into RAM via the bootloader, otherwise it will crash. Cheapskates couldn't integrate the flashloader into the bootloader :)

I was not able to glitch into the bootloader when it was running from 3.3V. YMMV. 3.6V produced instant results.

25 hidden items
Load more...

oguzhifihome commented on Jan 25, 2023

Hi, I'm in trouble with stm32 too. there is code protection i need to read the program:(

timmeh87 commented on Jan 26, 2023

this thread does not cover stm32, the theory that is applied here is for
hardware that the stm32 does not have, In the stm8 the firmware decides
once at startup whether to enable ROP, if you can find something similar
for the stm32f then it might work, with different timing, but I dont think
it uses the same system. It sounds like it enables permanent write
protection so you might be dealing with a burned out fuse or something

https://www.st.com/resource/en/application_note/an3429-stm32-proprietary-code-protection-overview-stmicroelectronics.pdf

…

👍 1

Nagra3 commented on Feb 6, 2023

Good afternoon.
Could any friend write a code for epm240 with glith in 78us?

I don't know how to program.

Write Preview

Sign in to comment