xuất bản trên 18 Tháng sáu, 2020 trong Sáng tạo mẹo & thủ thuật công nghệ

Các cuộc tấn công chi phí thấp vào bảo vệ đầu đọc STM8

Là một phần trong dự án hack HC-12 của mình , tôi cần lấy phần sụn của bộ vi điều khiển STM8 đã bật tính năng bảo vệ đầu ra.

Tôi đã bị hấp dẫn từ lâu bởi các cuộc tấn công tiêm lỗi , gần đây nhất được kích hoạt bởi Cuộc nói chuyện 35C3 này trên PS2 Vita Hacking sử dụng sự cố điện áp để vượt qua các biện pháp bảo vệ.

Từ hướng dẫn tham khảo STM8 :

> **4.5.1: Bảo vệ đầu ra**
>
> Bảo vệ đầu đọc được chọn bằng cách lập trình byte tùy chọn ROP thành 0xAA. Khi bật tính năng bảo vệ đầu đọc, việc đọc hoặc sửa đổi vùng nhớ chương trình Flash và DỮ LIỆU [sử dụng giao diện gỡ lỗi SWIM] bị cấm.
>
> Ngay cả khi không có biện pháp bảo vệ nào có thể được coi là hoàn toàn không thể phá vỡ, tính năng đọc ra cung cấp mức bảo vệ rất cao cho bộ vi điều khiển có mục đích chung.

Điều này trông giống như một mục tiêu tuyệt vời cho sự cố điện áp. Do đó: thách thức được chấp nhận! :-)

trục trặc phần cứng

Trong khi những người khác báo cáo đã thực hiện các dự án tương tự với Arduino, tôi nghĩ rằng tôi cần độ chính xác về thời gian của một FPGA (và dù sao cũng muốn thử viết phần cứng). Không muốn chi quá nhiều cho việc này, tôi đã sử dụng EPM240, một bảng phát triển CPLD thực sự rẻ tiền (~ 5 €) có khả năng chuyển đổi các Ghim IO ở tốc độ lên đến 50 MHz, điều mà tôi hy vọng là đủ.

Khó chịu là nó không dễ dàng sử dụng được từ Linux, nhưng cuối cùng tôi đã quản lý được .

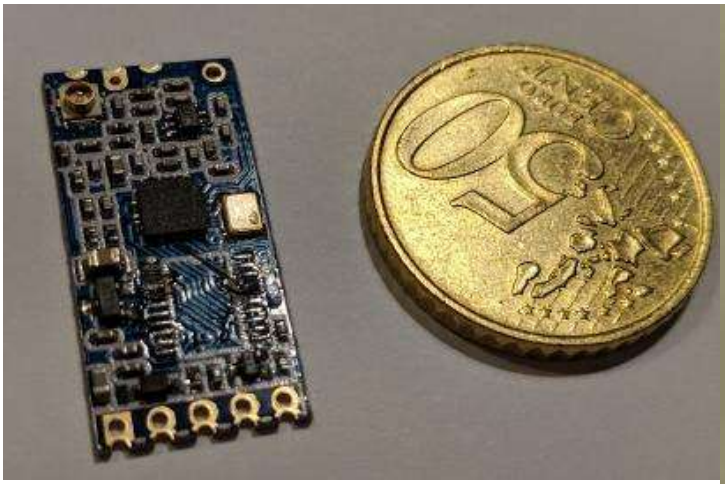
Việc viết VHDL tương đối đơn giản, nhưng ai có thể nghĩ rằng phần cứng thật kỳ lạ và việc gỡ lỗi phần cứng là một điều khó khăn (gỡ lỗi printf chẳng là gì so với chuyển đổi GPIO).

Vì phần biên dịch và phản xạ lại gây khó chịu và chậm, nên tôi đã viết phần mềm để nhận cấu hình thông qua triển khai UART ghi thông số trung tâm vào thanh ghi:

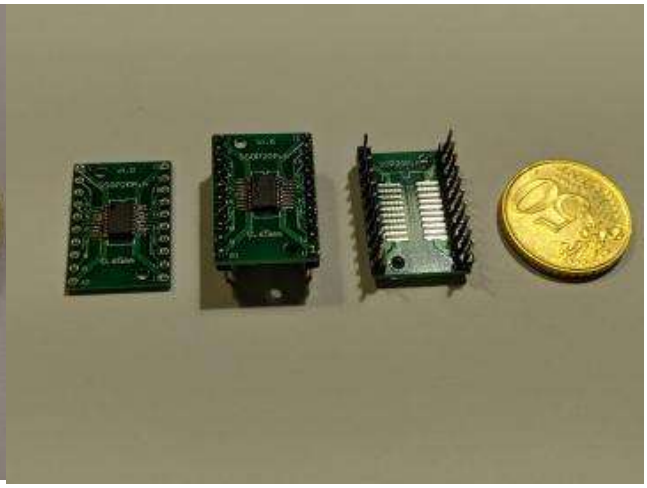
- kích hoạt phân cực
- số lần kích hoạt
- trục trặc phân cực
- độ trễ kích hoạt đến trục trặc
- thời gian trục trặc

Thiết lập trục trặc

Bo mạch HC-12 có một loạt các tụ điện sẽ đánh bại các mục tiêu trục trặc của tôi, vì vậy, trước tiên tôi đã cố gắng hàn HC-12 STM8 vào một bo mạch biệt lập, nhưng có thể đã làm hỏng nó trong quá trình này (vì vậy điều đó không được khuyến nghị).

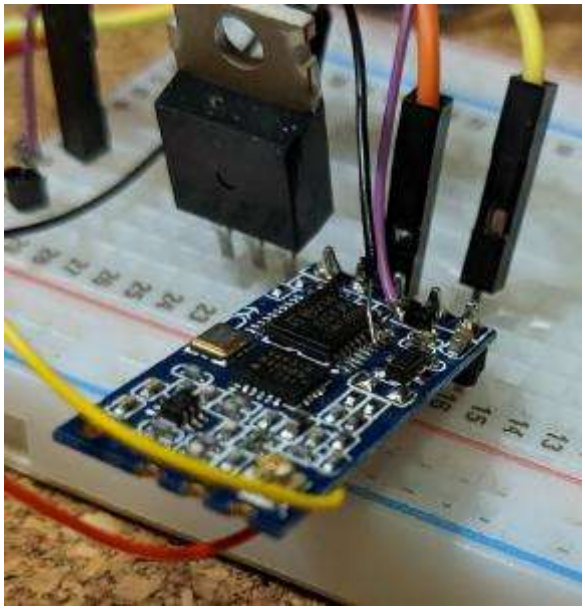


Tháo dỡ MCU

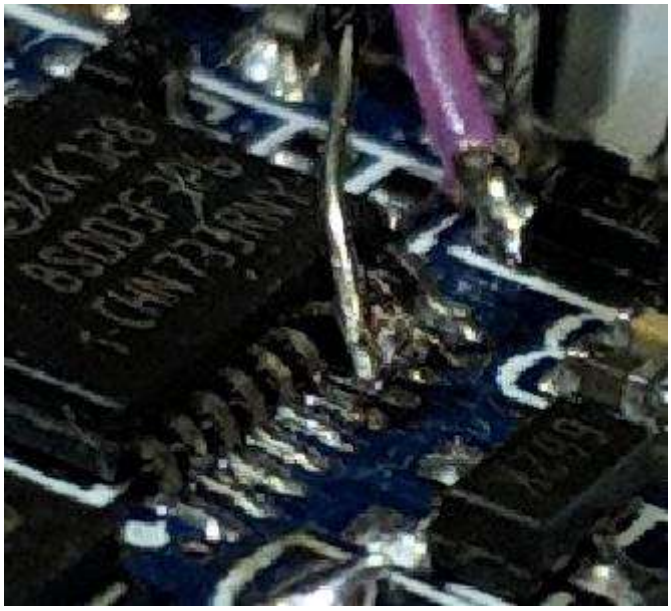


Đặt nó vào một bộ chuyển đổi nguyên mẫu

Nỗ lực tiếp theo của tôi là tháo chân GND ra khỏi ổ cắm và gắn một dây mới.

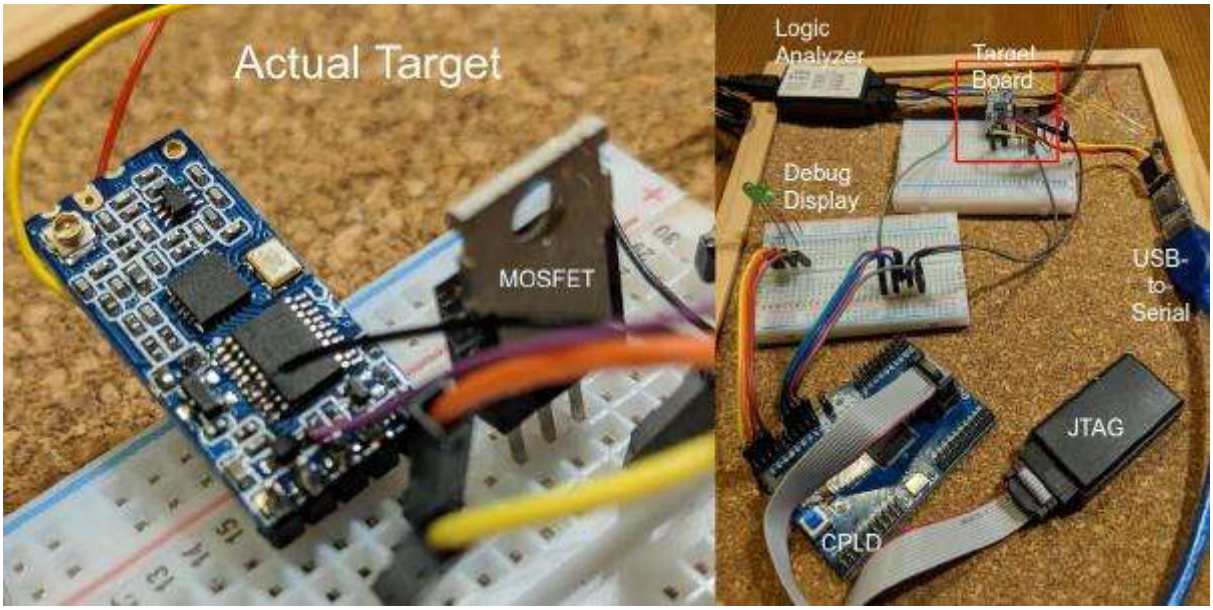


Pin GND tách rời (tổng quan)



Chân GND đã tách rời (đã phóng to)

Đây là toàn bộ băng ghế dự bị trông như thế nào:



Thiết lập trục trặc

Tổng số thiết bị được sử dụng là <20€.

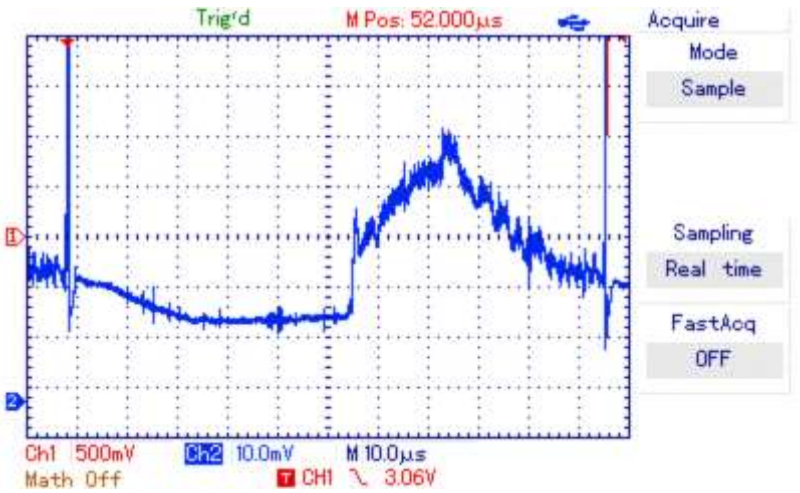
Thiết lập của tôi dường như hoạt động và tôi có thể gây ra hành vi dường như không xác định, nhưng tôi không thể trục trặc việc tải byte tùy chọn mặc dù bù thời gian và thời gian trục trặc trong nhiều đêm.

Cuối cùng, tôi nhận ra rằng nhiều khả năng tôi đã gặp trục trặc trong phần truyền UART hoặc bộ chuyển đổi nối tiếp USB của mình.

phân tích sức mạnh khác biệt

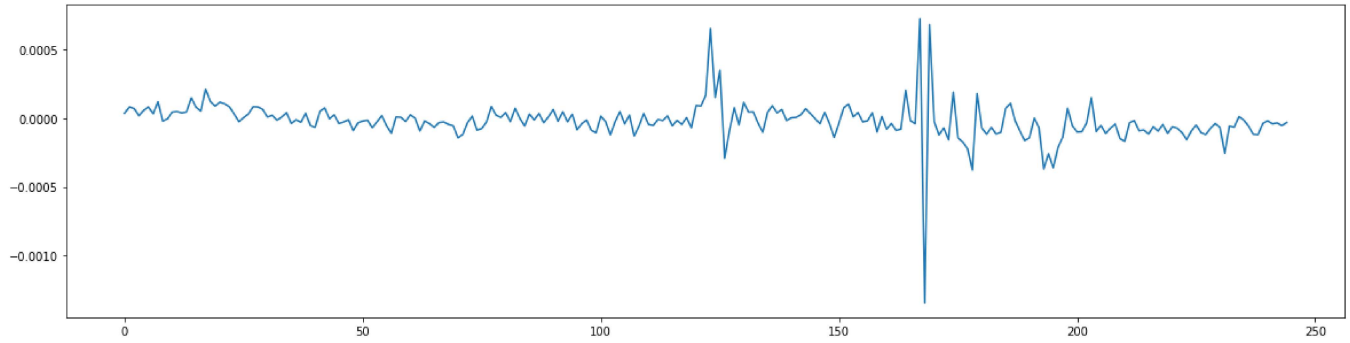
Một trong những vấn đề mà tôi gặp phải là tôi không thực sự biết rõ khi nào thì trục trặc và thử tất cả các tùy chọn. Vì vậy, tôi đã nghĩ ra một thiết lập phân tích năng lượng dựa trên giả định rằng bộ vi điều khiển tiêu thụ nhiều hơn (hoặc ít hơn) năng lượng khi tải các byte tùy chọn với số lượng bit khác nhau được đặt thành một.

Tôi đã mượn một máy hiện sóng UT2052 và phải viết một số trình điều khiển linux để nó có thể tự động truy xuất dấu vết. Sau đó, tôi đã chuẩn bị trước một STM8 với byte bảo vệ đầu đọc được đặt thành 0x00, 0xFF và 0xAA. Sử dụng máy hiện sóng, tôi đã ghi lại hàng trăm dấu vết năng lượng xung quanh giai đoạn thiết lập lại.

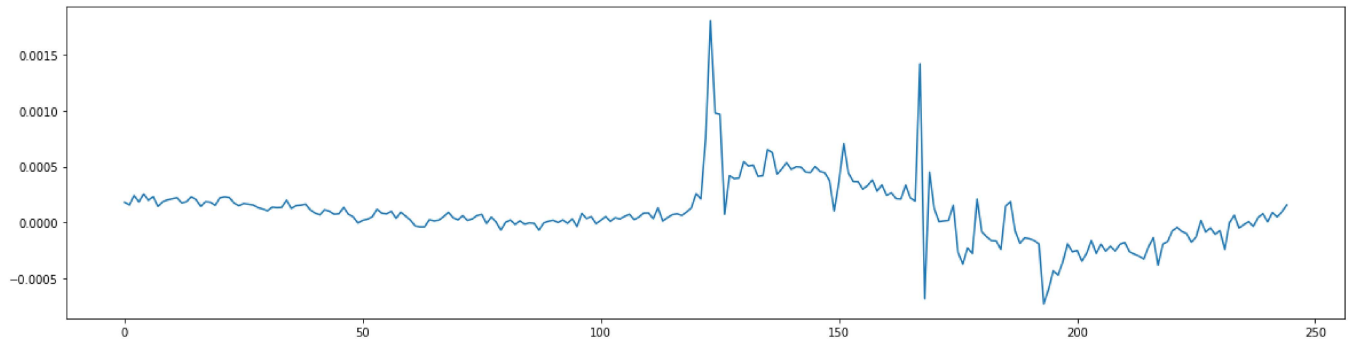


chế độ xem dao động của các dấu vết nguồn xung quanh giai đoạn thiết lập lại

Tôi đã căn chỉnh các dấu vết, tính trung bình, kiểm tra tính nhất quán và loại bỏ những dấu vết quá nhiễu. Biểu đồ nhiễu minh họa lượng nhiễu trong một tập theo dõi. Bất kỳ sự khác biệt đáng kể nào xảy ra tại những điểm này có nhiều khả năng là do nhiễu hơn là do các dòng điện khác nhau trong quá trình tính toán các giá trị khác nhau.



Với suy nghĩ đó, tôi đã xem xét sự khác biệt giữa giá trị trung bình của các tập hợp dấu vết khác nhau. Máy hiện sóng có độ phân giải thực sự kém (bản ghi <250 byte @8bits) và do đó SNR khá tệ. Theo đó, có những hiện vật mạnh mẽ, nhưng cũng có sự khác biệt rõ ràng. Điều hứa hẹn nhất là xung quanh vị trí 150: tăng đột biến tương quan âm theo sau là tăng đột biến tương quan dương mà không có sự kiện quan trọng nào trong biểu đồ nhiễu.



Có những ứng cử viên như vậy rõ ràng đã giúp tinh chỉnh không gian tìm kiếm, ngay cả khi chỉ nói rằng bỏ qua 50µs đầu tiên là an toàn.

Glitch với thành công

Thử lại với các mosfet khác nhau và không gian tìm kiếm được tinh chỉnh, tôi không chỉ thử trục trặc các chân VCC và GND mà còn cả các mã PIN của bộ điều chỉnh điện áp bị hở, vì biểu dữ liệu đã đề cập:

> **10.3.1 Tụ điện bên ngoài VCAP**
>
> Sự ổn định cho bộ điều chỉnh chính đạt được bằng cách kết nối một tụ điện bên ngoài CEXT với chân VCAP.

Trong nhận thức muộn màng, đây là một mục tiêu thực sự rõ ràng, nhưng tôi đã rất ngạc nhiên khi những nỗ lực của mình có hiệu quả ngay lần thử đầu tiên trong vòng vài giây sau khi thực hiện vũ phu. Sử dụng giao diện gỡ lỗi SWIM hiện được kích hoạt, tôi có thể chỉ cần kết xuất phần sụn.

Điều này có nghĩa là: **Bảo vệ đầu đọc STM8S bị hỏng** (mặc dù không thực sự ngạc nhiên ở đó).

Và nếu bạn cần đọc bộ nhớ của chip STM8 được bảo vệ bằng đầu đọc, hãy cho tôi biết!

---

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

rumpeltux commented on Jul 6, 2020 Owner

> 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

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

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

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?

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!)

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