λ

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-discovery-and-spi-flash-extraction.md

using-buildroot-for-reverse-engineering.md

tricore-basics:-using-the-hightec-tricore-toolchain-in-linux.md

30 Jan 2020

gging for Reverse Engineers Part 1: SWD, CD and Xbox One Controllers

otential targets for my next post and was pleasantly surprised to find the packaging:

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

I thought it might be interesting to tear down this controller and see what kind of he target?

mented in such a way that allows us to learn more about it’s internal operations?

ıged, either through software exploitation or hardware modifications?

ɜe questions will be a hardware teardown.

ıg PCB:

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

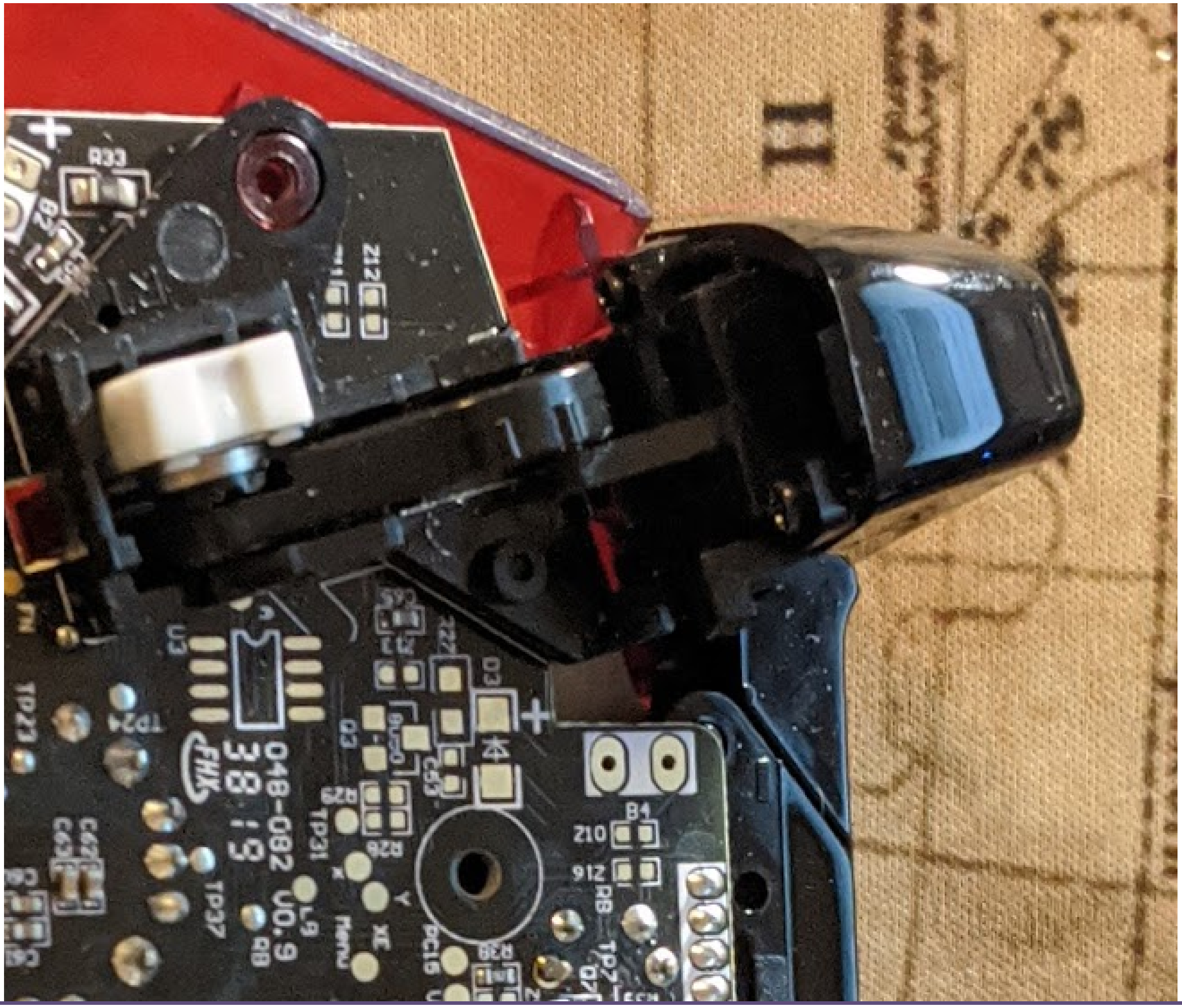
writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-



Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

1 /stm-xbox-jtag/ ⌘ blogspace λ * Menu

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

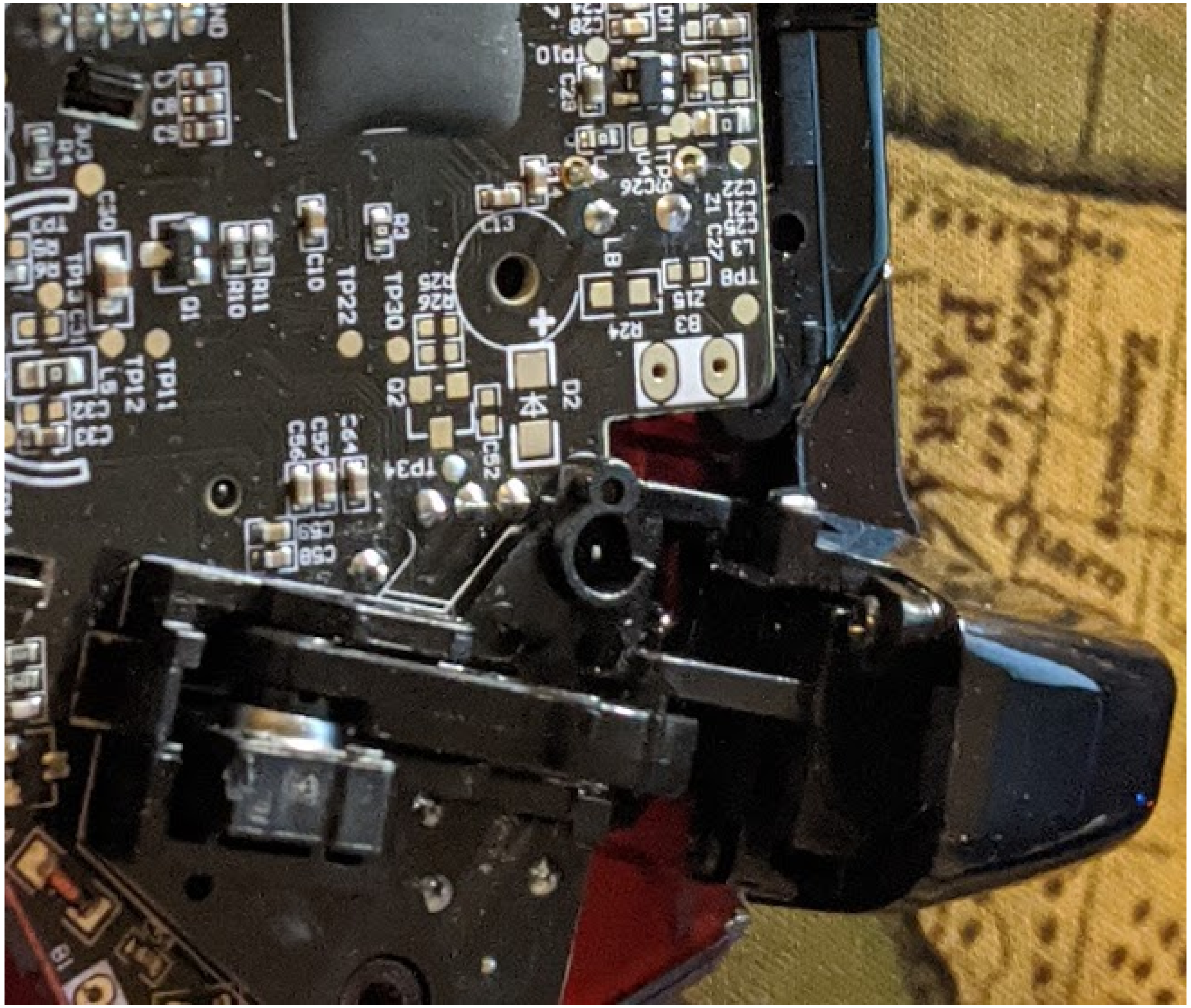
writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-



he bottom of the board, but this is an audio codec chip. The datasheet can be

bit stereo CODEC with a microphone, headphone and speaker amplifiers.

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

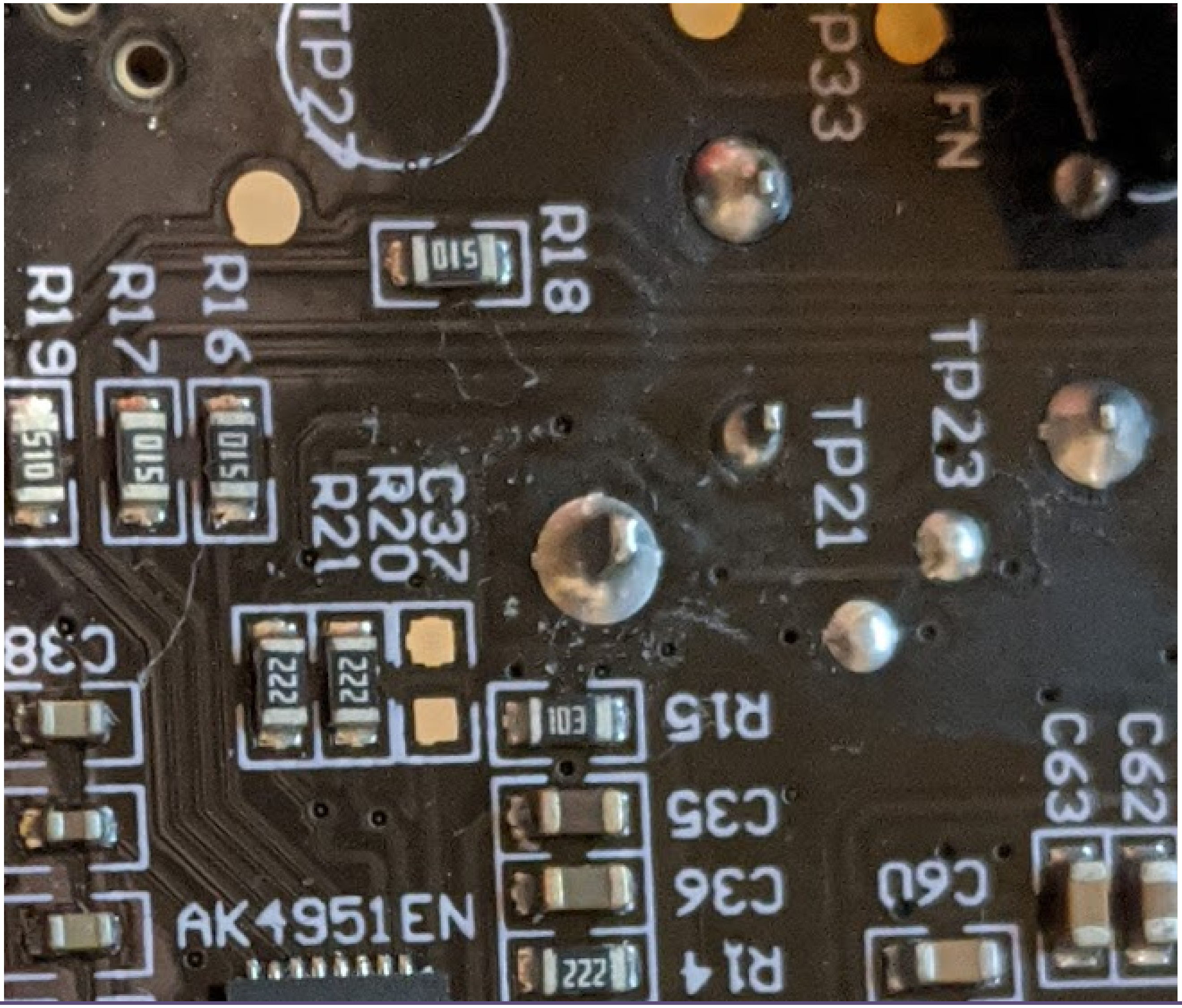
writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

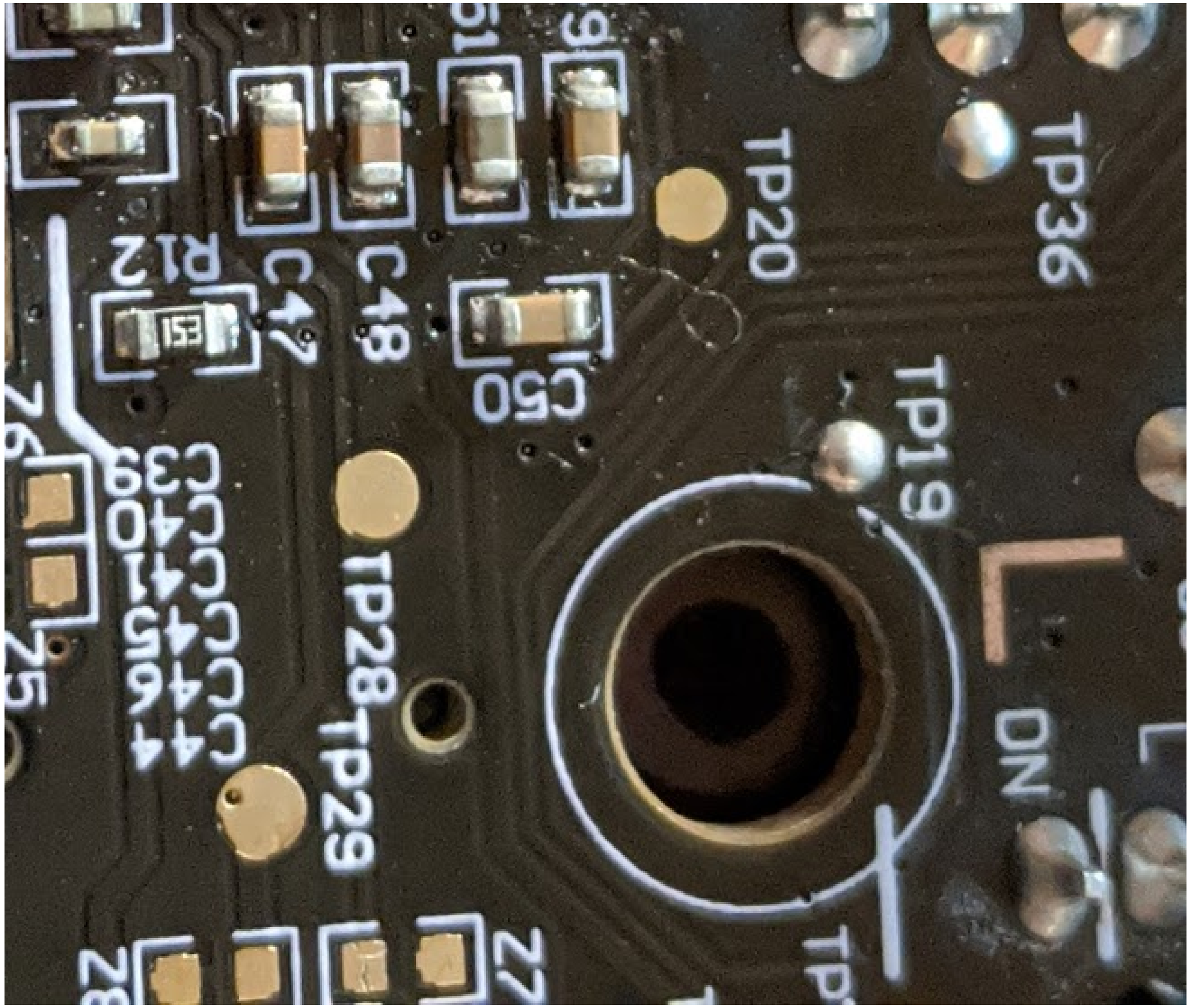
basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-



1 /stm-xbox-jtag/ ⌘ blogspace λ * Menu

1 → home
2 → my posts
3 → series
4 → tags
5 → about me



Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

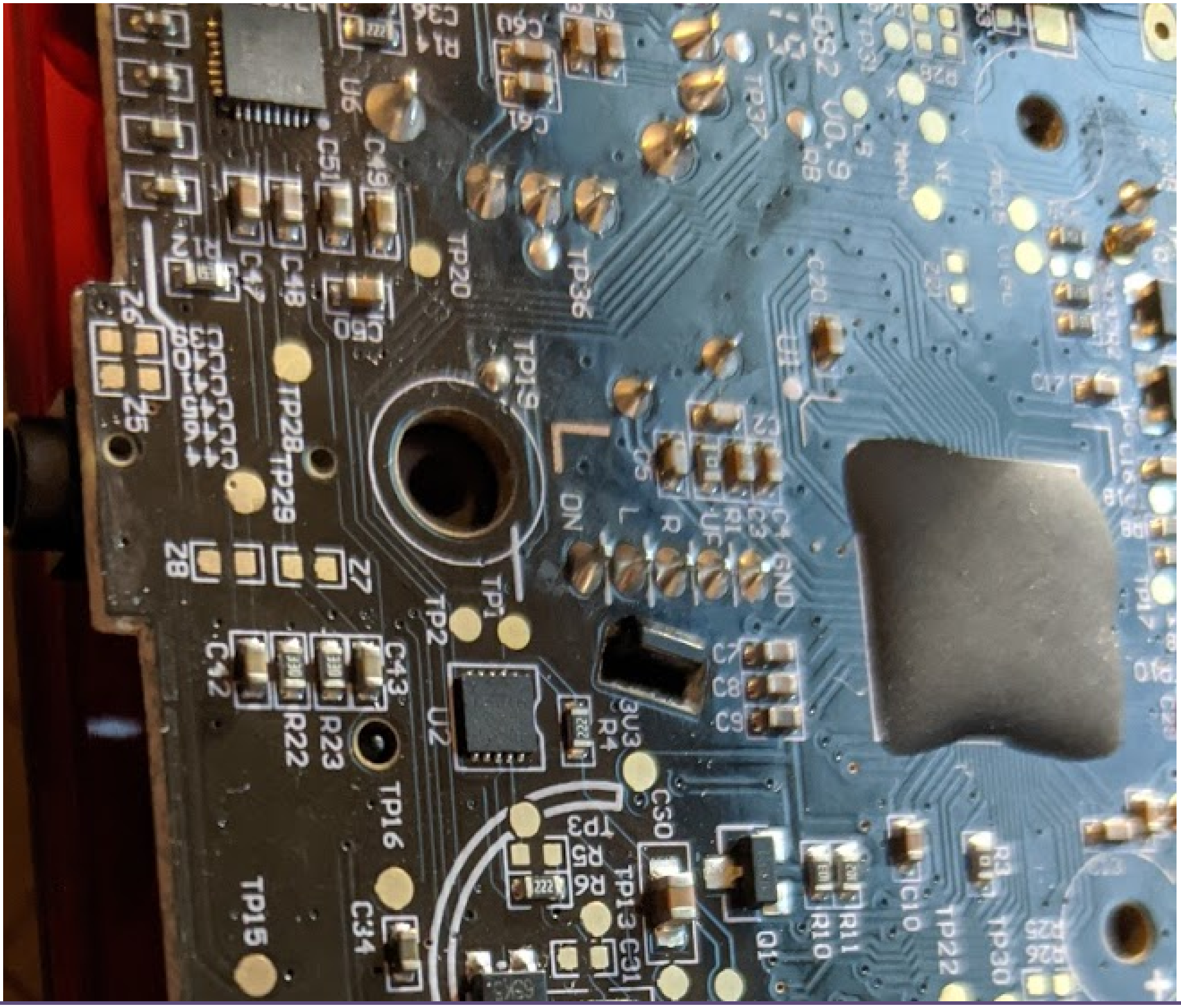
writing-a-ghidra-loader:-stm32-edition..md

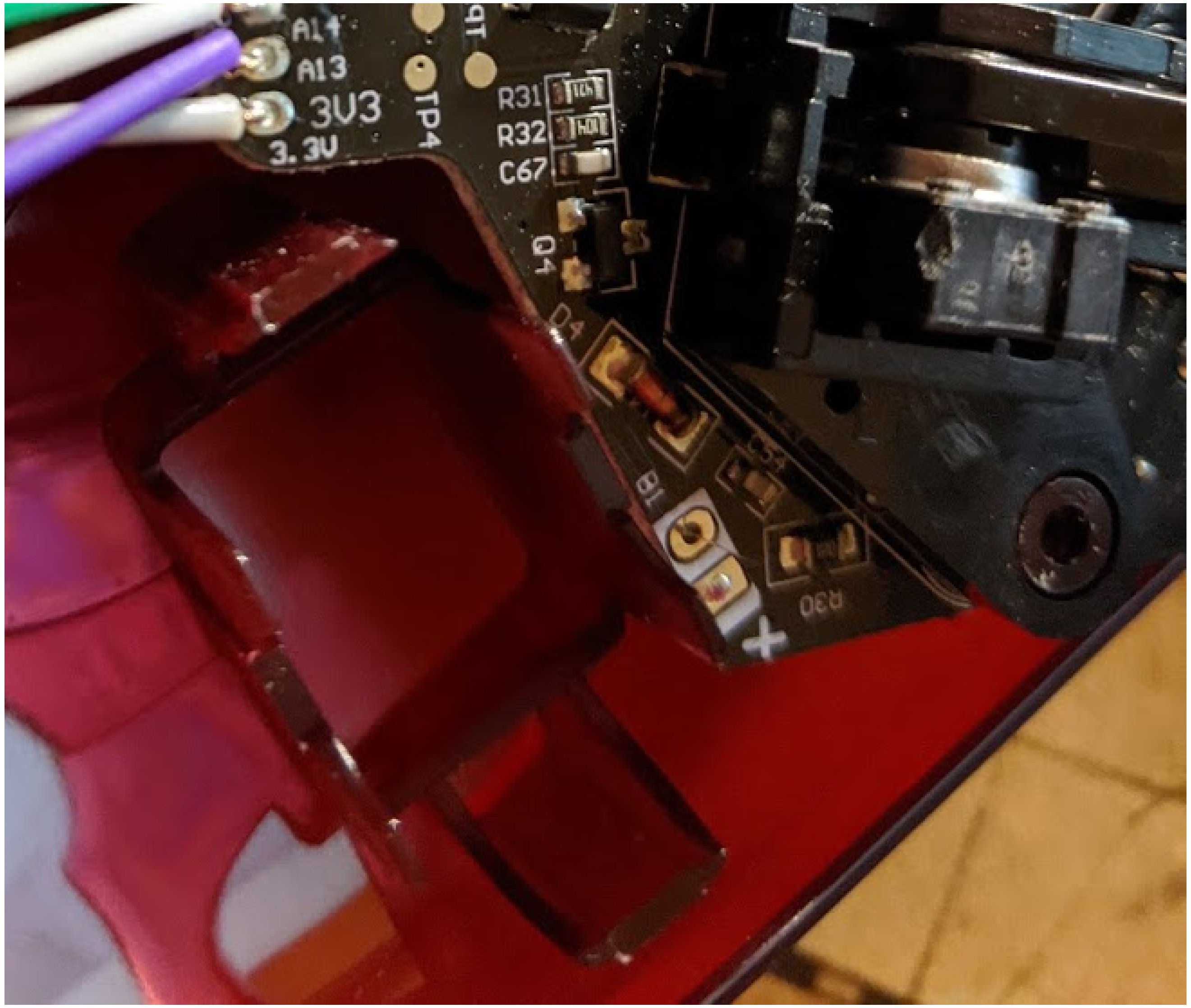
hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-





Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Pin	Value
0/NA	0 (GND)
RES	3.3V
A14	0.1V
A13	3.3V
3V3	3.3V

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

on RES, A14 or A13, so these must be for something else, but what? Given that

ıds for system reset) there is a good chance that there are JTAG or SWD headers.

s the target by pulling it low with a 10k resistor (remember we're reversing things

t something!). If you are not familiar with these types of headers or how a system

active low meaning that they idle at a high value and have to be pulled low to be

f `dmesg -w` and toggle this line low with a 10k resistor, what do we see?

```
device strings: Mfr=1, Product=2, SerialNumber=3
 PDP Wired Controller for Xbox One - Crimson Red
urer: Performance Designed Products
mber: 0000AE38D7650465
x pad as /devices/pci0000:00/0000:00:14.0/usb1/1-6/1-6.4/1-6.4:1.0/input/input26
```

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

roller to reset, that's one pin down, 2 more to go.

, a common assumption is that it's for JTAG or some other form of hardware level
iires that there be at least 4 pins, `TDO`,`TDI`,`TMS` and `TCK`. We only have two on our
is is a Single Wire Debug (SWD) port.

hat is used for ARM Cortex targets. As the name implies, SWD only requires one
ı we determine which one is which? Before we go down that route, we should
) works and what tools can be used to interface with it.

e below pulled from this document provides a visual representation of how the

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

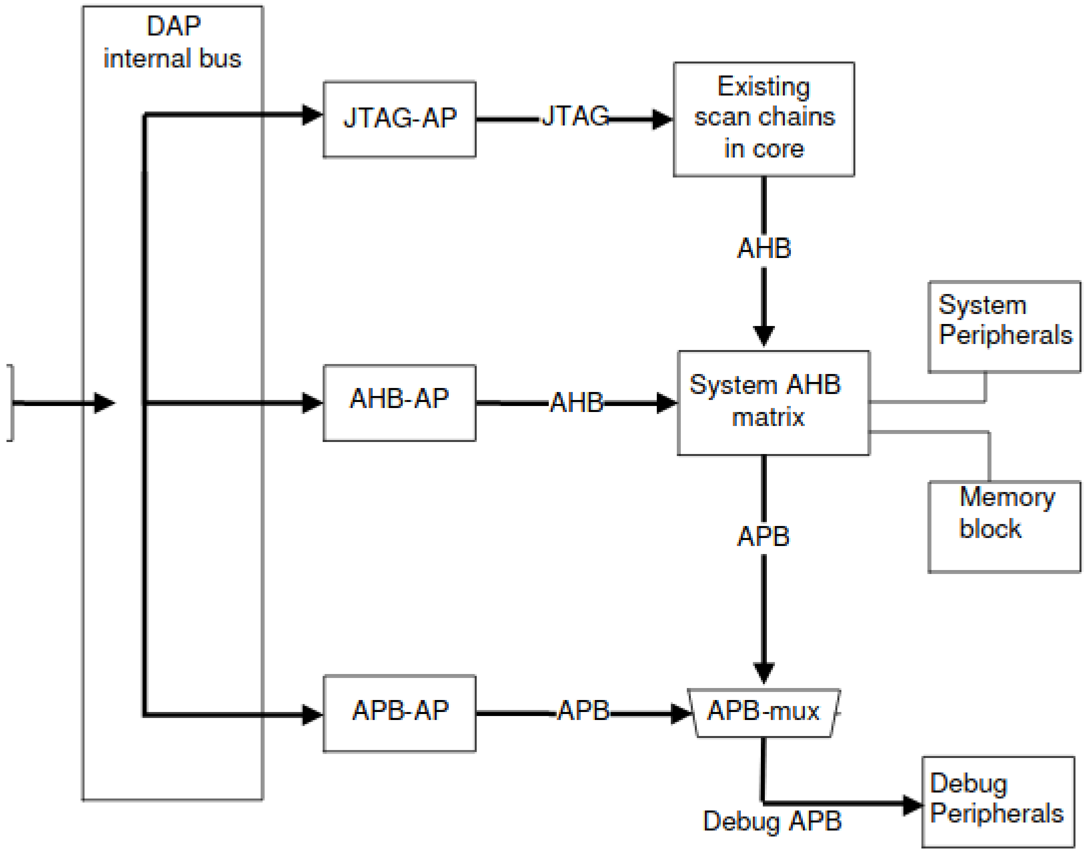
writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-



1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

egisters, with one register that is used to identify the type of AP. The function and

registers are accessed and utilized. You can find all of the information regarding

dard APs here. The ARM interface specification defines two APs by default and they

MEM-AP also includes a discovery mechanism for components that are attached to

eloped as a pseudo-replacement for JTAG. With SWD the pin count was reduced

ame functionality of JTAG. One downside to SWD however is that devices can not

llowed for. The two pins that are used in SWD are below:

Purpose
gnal to CPU, determining when data is sampled and sent on `SWDIO`
onal data pin used to transfer data to and from the target CPU

onse

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

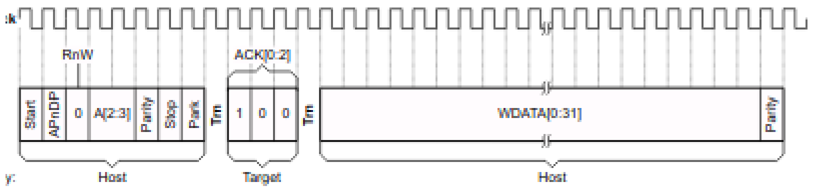
hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

image below, I've broken out the various fields in the table as well.



Usage
r the Debug Port access register or the Access Port access register is to be used.
or DP address
quests
ring the turnaround period

1 /stm-xbox-jtag/ ⌘ blogspace λ * Menu

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

arget) there is a turnaround period, which basically means that the target will now

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

ɔort uses these packets to interface with the DAP, which in turn allows access to ebugging as well as memory read / write capabilities. For the purposes of this post rform these transactions. We will review how to build and use OpenOCD next.

```
al libusb-1.0-0-dev automake libtool gdb-multiarch
```

ld!

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

debug this controller over SWD. In order to do this we need to tell OpenOCD at

ch debug adapter are we using)

2232H which we used in a previous post to dump a SPI flash. With this interface we

about the target via SWD, which is important because at this stage in the reversing

rget CPU is!

ns on the FT2232H need to be connected to a SWD target:

FT2232H Pin	SWD Target

SWD adapter, you have to put a 470 OHM resistor between `AD1`/`AD2` on the

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

hooked up to the target we can use the following script to query the `DPIDR`

```
fy this here

2H

the default

s is used to properly set and confiture the state of the lines we are using

 are using, and the port

 pin, in our case we're using
010
osed to another transport layer such as JTAG

is will vary based on what your hardware supports

terms) with name chip and role CPU, -enable let's OpenOCD to know to add it to th

st be explicitly created according to the OpenOCD docs
```

1 **/stm-xbox-jtag/** ⌘ blogspace λ *** Menu**

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

shown, with the following output (note that the first time it was run, there was no

ıes the following output was printed out). See the table below for the connections

T2232

FT2232H Pin	Controller
`AD1`	SWD (`A13`)
`AD0`	SCLK (`A14`)
`AD4`	SRST (`RES`)

```
x$ sudo openocd -f openocd.cfg
1040-ge7e681ac (2020-01-27-18:55)


ygen/bugs.html

tcl connections
telnet connections


re targets defined
```

the exact processor that is being used - if this is one that has a configuration file

sh banks and get other auxiliary information from the target processor. With this

OCD to create a target, using the chip with the Cortex M definition, this will

ge of the DAP and get access to some of the more generic features while we try to

ting:

```
U, cortex_m is the CPU type,
ap chip.dap
ary information from the DAP, kicks off the debugging session, etc

e DAP, including the ROM table
```

ile we see the following results:

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

1 **/stm-xbox-jtag/** ⌘ **blogspace λ** *** Menu**

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

```
gdb connections
```

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

```
ress 0xe00ff000
000a0411
 STMicroelectronics
ecognized
 0x1, ROM table
ory present on bus

ress 0xe000e000
000bb00c
 ARM Ltd.
x-M4 SCS (System Control Space)
 0xe, Generic IP component

ress 0xe0001000
003bb002
 ARM Ltd.
x-M3 DWT (Data Watchpoint and Trace)
 0xe, Generic IP component

ress 0xe0002000
002bb003
 ARM Ltd.
x-M3 FPB (Flash Patch and Breakpoint)
 0xe, Generic IP component

ress 0xe0000000
003bb001
 ARM Ltd.
x-M3 ITM (Instrumentation Trace Module)
 0xe, Generic IP component
 ARM Ltd.
tex-M4 ETM (Embedded Trace)
 0x9, CoreSight component
e Source, Processor
```

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

```
tcl connections
telnet connections
```

y interact with the DAP and MEM-AP, but we can also debug the target via GDB.

PU is an STM32F2X series because of the 0x411 part number in the MEM-AP entry:

```
ress 0xe00ff000
000a0411
 STMicroelectronics
ecognized
 0x1, ROM table
```

information we can modify the OpenOCD script to read these regions and look for

has the necessary offsets for the ID information:

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

STM Series	Offset
Generic Device ID Reg	0xE0042000
STM32F0/STM32F3	0x1FFFF7AC
STM32F1	0x1FFFF7E8
STM32F2/STM32F4	0x1FFF7A10
STM32F7	0x1FF0F420
STM32L0	0x1FF80050
STM32L0/ L1 Cat.1,Cat.2	0x1FF80050
L1 Cat.3,Cat.4,Cat.5,Cat.6	0x1FF800D0

1 /stm-xbox-jtag/ ⌘ blogspace λ * Menu

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

```
ffff

ffff

3639


4


4


4
```

ing command, using the flash address from the datasheet for this chip, or the

1 **/stm-xbox-jtag/** ⌘ **blogspace λ** *** Menu**

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

:an remove the target `swd`, `dap` and `target` lines from our config file, and replace

ıare/openocd/scripts/target/stm32f2x.cfg from the command line. This

We also know now that this STM32F2 series chip has 0x100 1kb pages of flash

```
x$ sudo openocd -f openocd.cfg -f /usr/local/share/openocd/scripts/target/stm32f2:

1040-ge7e681ac (2020-01-27-18:55)


ygen/bugs.html



tcl connections
telnet connections


6 breakpoints, 4 watchpoints
gdb connections
```

1 /stm-xbox-jtag/ ⌘ blogspace λ *** Menu**

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

```
idth 0 chip_width 0} {name stm32f2x base 536836096 size 0 bus_width 0 chip_width
```

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

```
in from flash bank 0 at offset 0x00000000 in 3.690861s (69.361 KiB/s)
```

```
from flash bank 1 at offset 0x00000000 in 0.007852s (63.678 KiB/s)
```

db using the commands below:

```
x$ gdb-multiarch
1.0.20180409-git
oundation, Inc.
or later <http://gnu.org/licenses/gpl.html>
e to change and redistribute it.
t permitted by law.  Type "show copying"

-linux-gnu".
iguration details.
ease see:
ugs/>.
```

```
31385114        0x30373639        0xc000fcc0
67ff47d2        0x05dcf000        0x04a803b3
ffffffff
```

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

ed, we can debug and single step through the firmware, but … can we reflash the

gs in the firmware image and patch them, we can use that as a visible method to

Let's load up the firmware in GHIDRA and see if we can find them, the firmware

)000. We know that the firmware is loaded at `0x8000000` based on the datasheet,

et this could be determined from OpenOCD by issuing the `reset halt` command

truction. Luckily, this firmware image is rather small and Ghidra makes quick work

`mesg` output can be seen in the screenshot below:

1 /stm-xbox-jtag/ ⌘ blogspace λ * Menu

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

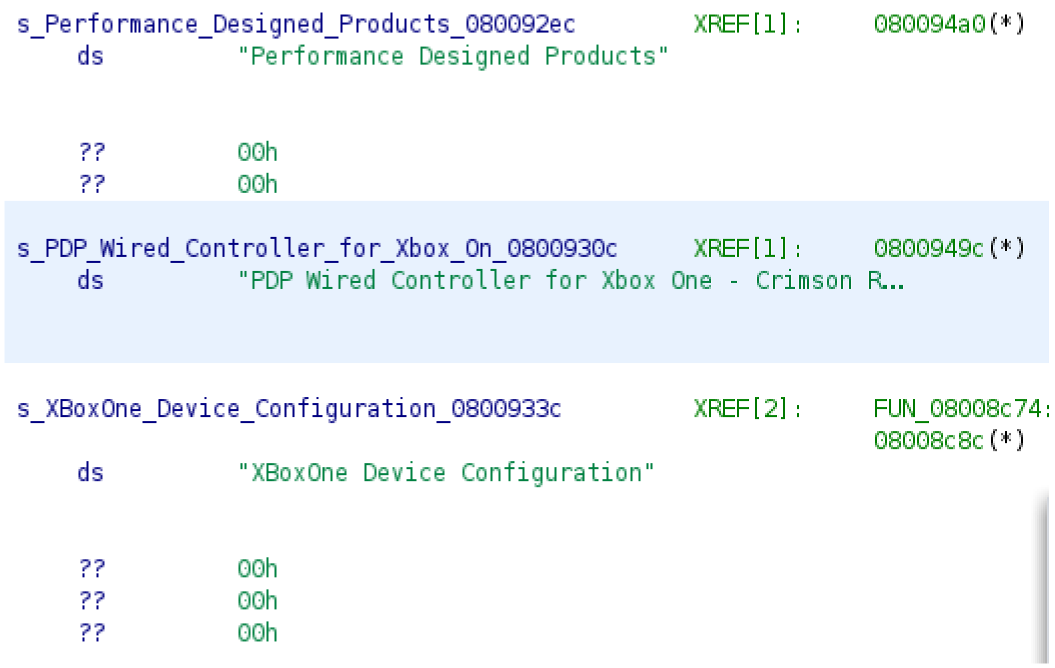
writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-



```
s_Performance_Designed_Products_080092ec        XREF[1]:     080094a0(*)
    ds          "Performance Designed Products"


    ??          00h
    ??          00h

s_PDP_Wired_Controller_for_Xbox_On_0800930c     XREF[1]:     0800949c(*)
    ds          "PDP Wired Controller for Xbox One - Crimson R…


s_XBoxOne_Device_Configuration_0800933c         XREF[2]:     FUN_08008c74:
                                                             08008c8c(*)
    ds          "XBoxOne Device Configuration"


    ??          00h
    ??          00h
    ??          00h
```

: string, changing it to “Testing Firmware Patches”. The flash can be overwritten with

D telnet console:

```
x$ telnet localhost 4444
```

```
, current mode: Handler External Interrupt(67)
p: 0x2000ff48

quired for the new settings to take effect.

, current mode: Thread
p: 0x20010000


in
-patch.bin to flash bank 0 at offset 0x00000000 in 3.744948s (68.359 KiB/s)
```

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

ɔt make sense, so I wanted to explain them:

ɔf *any* flash image before you attempt to reflash.

bit which keeps prevents unwanted writes. This is set in the “Option bytes” of the

lock the flash, which can sometimes not be an option!

1 /stm-xbox-jtag/ ⌘ blogspace λ * Menu

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

, and restart the CPU, prompting the following to show up in `dmesg`

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

```
-speed USB device number 14 using xhci_hcd
device found, idVendor=0e6f, idProduct=02a2, bcdDevice= 1.0f
device strings: Mfr=1, Product=2, SerialNumber=3
 Testing Firmware Patches
urer: Performance Designed Products
mber: 0000AE38D7650465
x pad as /devices/pci0000:00/0000:00:14.0/usb1/1-6/1-6.4/1-6.4:1.0/input/input28
```

fully extracted and loaded into ghidra, as well as the ability to modify it as we see

But it's getting late and I almost missed the monthly post deadline for January so I

embedded system, you typically want to enumerate and explore all possible

1 **/stm-xbox-jtag/** ⌘ blogspace λ *** Menu**

1 → home
2 → my posts
3 → series
4 → tags
5 → about me

Posts

introduction-to-reverse-engineering-with-ghidra:-a-four-session-course.md

hardware-debugging-for-reverse-engineers-part-2:-jtag,-ssds-and-firmware-extraction.md

writing-a-ghidra-loader:-stm32-edition..md

hardware-debugging-for-reverse-engineers-part-1:-swd,-openocd-and-xbox-one-controllers.md

basicfun-series-part-4:-i2c-sniffing,-eeprom-extraction-and-parallel-flash-extraction.md

basicfun-series-part-3:-dumping-parallel-flash-via-i2c-i/o-expanders.md

router-analysis-part-1:-uart-

ging tools. OpenOCD was also used with a FT2232H based interface to extract irmware onto the target. Thanks for reading and if you have any questions or just f please feel free to ping me on twitter

Share

Made with 💚