Leveraging UART, SPI and JTAG for firmware extraction

2019-05-10

Why ?

- Everything that is powered is/may become a computer
- Computer made from CPU, RAM, storage
- Many devices (IOT) offer only strictly limited access, no way to modify, reinstall or even observe the software running on them.
- Our duty to discover and report security and privacy problems requires full access thus the need to learn methods to extract software running on IOT devices.

Easier methods to obtain firmware images, try them first

Known or find-able firmware images on manufacturer web sites, use search-fu

Manufacturer may provides trial version of their appliance on virtual machines

Some appliances will have firmware stored on removable devices

- hard drives
- SD cards

Leveraging UART, SPI and JTAG for firmware extraction

When all of these fails…

Visual inspection

Upon disassembly …

Search for ground and VCC traces for voltage measurements

Search for uart, SPI, I2C and JTAG headers, look for vias, pins, markings on the board

Search for CPU and flash chips, take note of chip labeling and search for online documentation

Caution: risk of frying flash chip and/or entire board

VCC is usually 3.3V but check with multi-meter or scope before connecting anything.

VCC may be 1.8v on newer chips and boards. Of course, applying 3.3v on a chip designed to receive 1.8v may destroy it and even brick the device.

Finding ground and VCC.

- 0V mapped to logical 0
- VCC mapped to logical 1

Ground traces are usually the largest, look for larges traces on the bottom side of the board

Faraday cages should be grounded

Read labeling on chips, search online documentation for ground and VCC pins

On headers, some pins will be connected to the ground, some other will be pulled high to VCC

Use continuity tester while board powered down to test ground connection between power outlets, battery terminals, faraday cages, chip and header pins.

Finding UART headers

Usually 3 to 6 pins straight

Uses 3 pins, may even be marked

- GND
- RX
- TX

Most of the time, headers will need to be soldered in

Leveraging UART, SPI and JTAG for firmware extraction

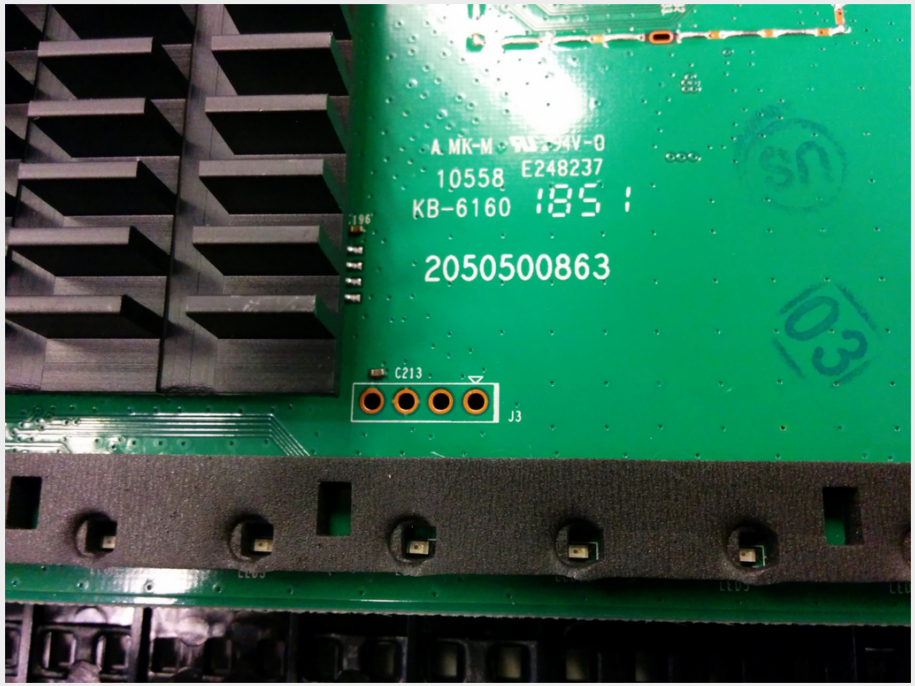

Figure 1: UART header J3

Finding SPI NOR flash chips

More on NOR vs NAND flash later

SPI flash chips may contain the bootloader, configuration data and, on some devices, the entire operating system.

Look at markings on such chips

```
- Those made by Winbond starts with W25
- Those made by Macronix starts with MX
```

Leveraging UART, SPI and JTAG for firmware extraction

Other common manufacturers: SST, SPANSION, Intel, Hyundai

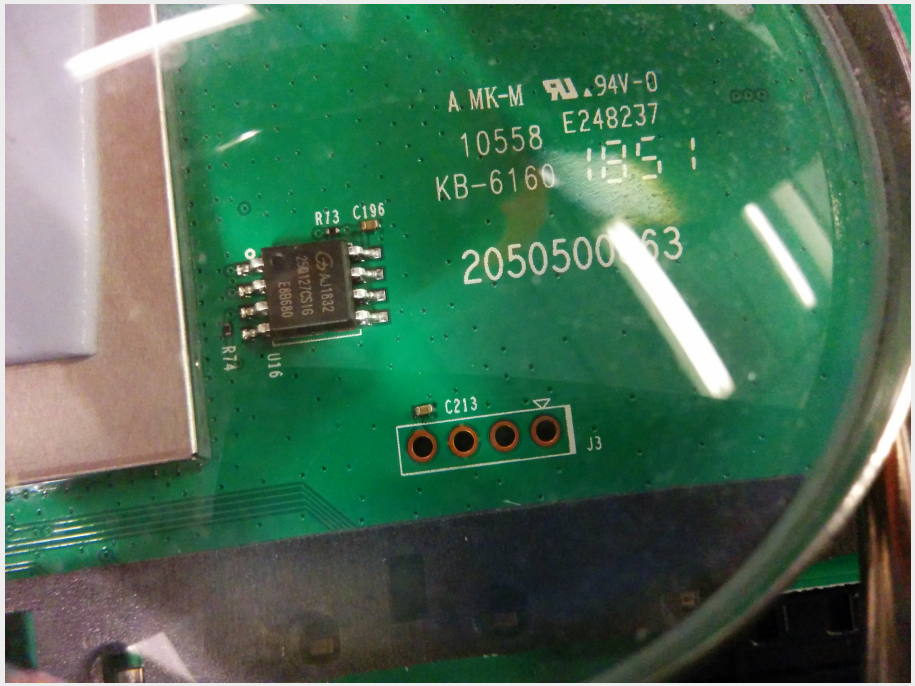

Figure 2: Popular packages for SPI flash are SOIC 8 pins or SOIC 16 pins

Finding JTAG headers

JTAG provides a hardware debugging facility. Can set breakpoints, access registers and memory, read flash banks.

Usually 10, 14 or 20 pins on 2 rows.

Could be a single row, pads or vias.

Look for JTAG labels: TCK, TMS, TDI, TDO, TRST

Leveraging UART, SPI and JTAG for firmware extraction

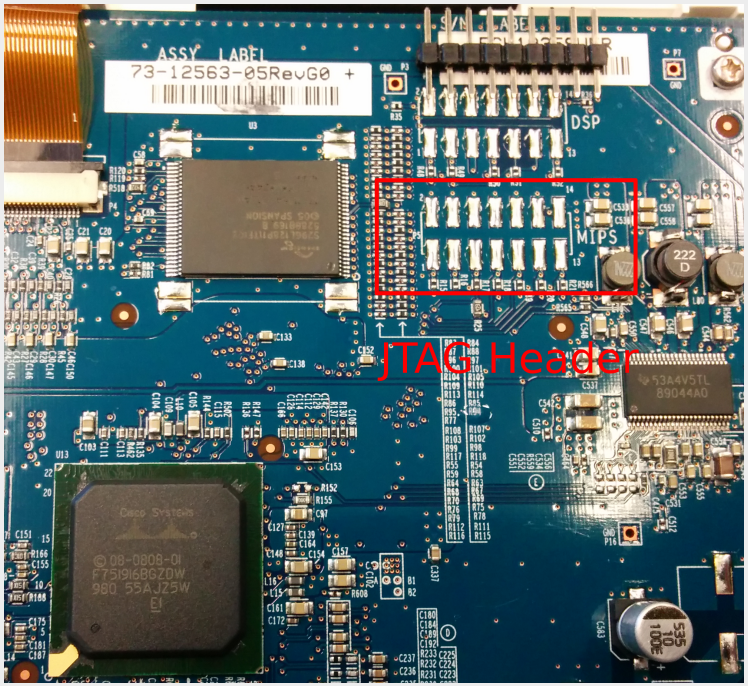

Figure 3: MIPS JTAG header on a Cisco IP phone

Other components of interest

CPU: lookup model and manufacturer. From documentation look for architecture, memory map and reset vector. Such documentation may be held by NDA, search harder.

Nand flash chips. They are harder to read than SPI NOR flash but possible. Use 360 clip, E3 NOR flasher, allsocket adapters or similar tools

I2C memory chips. Smaller capacity than NOR flash, can still hold useful information.

Tool set

- wires (male to male, female to male, female to female)
- soic 8 test clip (pomona 5250)
- uart to usb adapter (pl2303 or bus pirate)
- spi programmer (C232hm, bus pirate, beaglebone, rasberry pi)
- jtag programmer (c232hm, bus blaster, arm-usb-tiny, black sphere, etc)
- jtagulator http://www.grandideastudio.com/jtagulator/

Tool set contd

- logic analyzers
  - saleae
  - bitscope
  - saleae clone https://www.amazon.com/dp/B07K6HXDH1
- digital oscilloscope

open-source software

- compile from sources
  - flashrom
  - openocd
  - sigrok (with pulseview)
- install from package manager
  - screen
  - minicom

Proprietary software (specific machine recommended)

- saleae logic
- bitscope logic
- might need proprietary JTAG hardware & software for some boards

UART aka serial port

UART = Universal Asynchronous Receiver Transmitter

Used to link modems to computers a while ago

Still commonly used on switches, routers

Found on many circuit boards on consumer grade and industrial devices

Why go for UART headers first

- may give root shell right away, limited shell or some configuration software with limited functionality
- bootloader may be interrupted, could allow to load custom firmware
- bootloader and kernel messages can be captured. They contains valuable information about the platform such as:
  - memory map (what is being loaded at which address in memory)
  - SPI flash partitions
  - CPU manufacturer, model and architecture

Easy to use

- 3 pins, RX, TX, GND
- hardware needed:
- usb to UART adapter or bus pirate

software:

- can be accessed with screen or minicom on Linux
- putty on windows
- similar software on other operating systems

UART electrical signals

Asynchronous means no clock signal used to synchronize transmitter and receiver One start bit at the beginning and at least one stop bit at the end of each frames Most common setting is 8 data bits, no parity and 1 stop bits, 115200 bps

At 115200 bps, 1 bit will last about 9us; 1s/115200 bps = 8.68e-06s

Other commonly use speed settings: 1500000 bps (newer devices) 57600 bps 9600 bps (on older devices)

Transmit pin usually pulled high when silent (polarity setting)

Leveraging UART, SPI and JTAG for firmware extraction

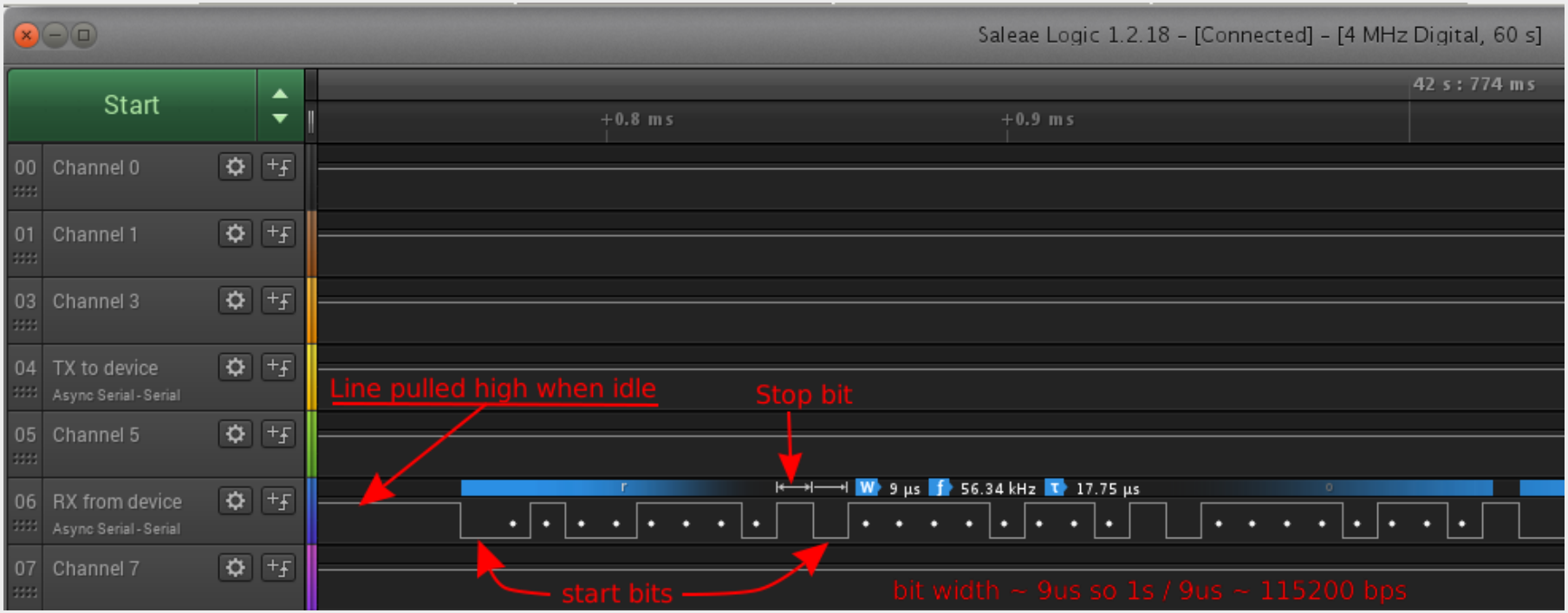

Figure 4: UART frame decoded using a logic analyzer

How to determine serial port parameters

- most common settings: 115200 bps, 8 bits, no parity, 1 stop bit (115200-8N1)
- settings may be different, some methods :
- trial and error aka bruteforce
- determined using a logic analyzer e.g: http://bitscope.com/product/BS10/
  - Insert screenshots of logic analyzer
- presence may be determined with a multimeter, voltage will fluctuate rapidly between 0 and VCC on the TX pin

- http://www.devttys0.com/2012/11/reverse-engineering-serial-ports/

- Oscilloscope with serial decode feature can also be used to find uart settings:
  - https://www.amazon.com/Siglent-Technologies-SDS1202X-Oscilloscope-Channels/dp/B06XZML6RD#detail-bullets

bus pirate

- Versatile tool made by dangerous prototypes.
- Easy to work with, less error prone than beagle bone or rasberry pi.
- Expose a serial port through USB for control and data transfers.
- The bus pirate support:
  - UART
  - SPI
  - I2C
  - JTAG (required latest firmware)

Leveraging UART, SPI and JTAG for firmware extraction

In this workshop, we will use the bus pirate to connect to the UART and access the SPI flash of a TPLink AC1200 router.

Accessing a serial port with the bus pirate.

Connect

- GND to GND
- MISO to TX
- MOSI to RX
- configure bus pirate as uart bridge (115200 8N1)

Using the bus pirate

Upon connection, a FTDI serial device converter should show up. Look for new devices added to your operating system.

Listing 1: Linux kernel logs when the bus pirate is connected

```
1 dmesg | tail -5
2 usbcore: registered new interface driver ftdi_sio
3 usbserial: USB Serial support registered for FTDI
      USB Serial Device
4 ftdi_sio 3-2:1.0: FTDI USB Serial Device
     converter detected
5 usb 3-2: FTDI USB Serial Device converter now
     attached to ttyUSB0
```

First, you may want to use script to record your terminal session

Listing 2: recording terminal session

```
1 $ script buspirate.log
2 Script started, file is buspirate.log
```

Use screen to connect to the bus pirate

Listing 3: connecting to the bus pirate

```
1  screen /dev/ttyUSB0 115200
```

(or use your favorite terminal emulator)

Listing 4: Press enter to get the HiZ> prompt

```
1 HiZ>
2
3 Press ? to show the bus pirate's online help
```

Listing 5: set bus pirate mode, Press 3 to select UART

```
1 Press m to select mode
2 HiZ>m
3 1. HiZ
4 2. 1-WIRE
5 3. UART
6 ...
7 (1)>3
```

Listing 6: configure UART speed

```
1  Set serial port speed: (bps)
2  ...
3   8. 57600
4   9. 115200
5  10. BRG raw value
6  Press 9 to select 115200 bps
7  (1)>9
```

Listing 7: configure UART bits and parity

```
Data bits and parity:
 1. 8, NONE (default)
 2. 8, EVEN
 3. 8, ODD
 4. 9, NONE
(1)>1
```

Listing 8: configure UART stop bits

```
1 Press 1,1 and 2 to set stop bits, polarity and
      output type
2
3 stop bits:
4  1. 1 (default)
5  2. 2
6 (1)>1
```

Listing 9: configure bus pirate UART polarity

```
Receive polarity:
 1. Idle 1 (default)
 2. Idle 0
(1)>1
```

Listing 10: configure bus pirate UART output

```
1 Select output type:
2  1. Open drain (H=Hi-Z, L=GND)
3  2. Normal (H=3.3V, L=GND)
4 (1)>2
5 Then you should get the UART> Prompt
6 UART>
```

Listing 11: list bus pirate macros relevant to UART mode

```
1 UART>(0)
2  0.Macro menu
3  1.Transparent bridge
4  2.Live monitor
5  3.Bridge with flow control
6  4.Auto Baud Detection
```

Listing 12: To communicate directly with the device, set the transparent bridge mode with (1)

```
1 UART>(1)
2 UART bridge
3 Reset to exit
4 Are you sure? y
```

How to extract firmware using UART

- Several ways to extract files or even entire filesystems from the serial port

Encode in base64 and send to terminal (slow but almost always available)

- Binaries may also be introduced into the platform using this method

Leveraging UART, SPI and JTAG for firmware extraction

Leverage other communication facilities of the device, is this case:

- USB port
- Wired and wireless connection

Additionnal data transfer utilities may be available in other devices:

- SATA ports
- SD card readers

USB port method

Find any thumb drive, format in fat32, connect to the port at the back of the router.

The thumb drive should be mounted under /tmp/ftp/sda1 after a few seconds. The LED at the right of the LED row will be lit solid.

Try ps w to enumerate server software, look for interesting processes and related files.

Copy files over /tmp/ftp/sda1

cp /usr/bin/cloud-brd /tmp/ftp/sda1/

Or copy entire filesystems …

A word about MTD partitions

MTD stands for Memory Technology Devices

Type of device driver to handle flash memory. Can be partitionned but information about partitions are not stored at a defined location on the flash memory.

A word about MTD partitions contd

Instead, information about MTD partitions will reside inside the boot-loader and the Linux kernel. Usually, the MTD partition table will be visible to whoever has access to the serial console port.

Relevant stackoverflow post:

https://stackoverflow.com/questions/8585864/nand-partitioning-in-u-boot

Listing 13: listing MTD partitions from kernel logs

```
root@Akronite:/etc/config# dmesg | grep -A 6 MTD
Creating 6 MTD partitions on "bcmsflash":
0x000000000000-0x000000080000 : "boot"
0x000000080000-0x000000ff0000 : "linux"
0x000000280000-0x000000ff0000 : "rootfs"
0x000000fb0000-0x000000fc0000 : "usb-config"
0x000000fc0000-0x000000fe0000 : "log"
0x000000ff0000-0x000001000000 : "nvram"
```

boot will be mapped to /dev/mtd0, linux (kernel) to /dev/mtd1 and so on.

Use dd to copy them

```
1 root@Akronite:/etc/config# dd if=/dev/mtdblock2
      of=/tmp/ftp/sda1/rootfs.bin bs=1
2 M
3 13+1 records in
4 13+1 records out
```

Umount /tmp/ftp/sda1 and plug thumb drive back into pc

Network method

Use one of the Ethernet ports or configure the wireless interface

The default password for the wireless access point is written under the router or try “uci show | grep psk” from the serial console

The default router IP address is 192.168.0.1

The installed busybox provides netcat for file transfers

Setup the netcat listener

Listing 14: setting up a listener on your PC

```
1  nc -l -p 8888 > file.txt
```

Extracting files using netcat on the router

Listing 15: extracting files from the AC1200

```
root@Akronite:/www/webpages# cat data/location.
    json | nc 192.168.0.147 8888
```

Listing 16: dumping filesystems from AC1200

```
1 dd if=/dev/mtdblock2 bs=1M | nc 192.168.0.147
      8888
2 13+1 records in
```

UART Exercises

Apply power to the AC1200 and capture boot and kernel messages

Get a root shell on the TPlink AC1200

Locate and extract a binary which make or receive network connections

- You may want to load /usr/bin/cloud-brd into your favourite debugger

Extract the linux kernel

Extract the root file system and unpack it

Flash memory

NOR flash chips usually hold from 32KB up to 16 MB. NOR flash memory can by accessed byte by byte in random order which allow the CPU to execute code directly from it . NAND flash pages must be copied to RAM first for code to be executed.

- NOR flash suitable for bootloaders, NVRAM.
- NOR flash can host small filesystem.
- NAND flash used to store larger filesystems on more powerful devices.

SPI NOR flash

- SPI stands for Serial Peripheral Interface
- NOR flash usually connected using a SPI bus.
- common formats: soic8, soic16, wson8
- Documentation on SPI flash memory generally available: e.g: Winbond w25q128fv
- Can be read/written with a soic test clip and SPI programmer (bus pirate, beagle bone, C232HM)

SPI Protocol

- Synchronous serial bus, use clock signal given by master
- Can be chained, selected with CS and HOLD pins

Inputs: DI, CLK, CS, WP, HOLD

- DI (data in) to be connected to programmer's MOSI (master out slave in) output
- CLK (clock) provides timing for serial input and output operations
- CS (chip select) enable a device when brought low. When CS is high, all IO set to high impedance.
- WP (write protect) used to prevent status registers from being written
- HOLD allow to pause device while actively selected, used when chip are chained on the bus

Outputs: DO

- At least one output: DO
- DO (data out) to be connected to programmer's MISO (master in slave out) input
- DUAL and QUAD SPI use bidirectional IO pins
- Some chips do have a special QE (Quad enable) register that will turn WP and HOLD into bidirectional pins for QUAD SPI operation

The SPI programmer will query the NOR flash

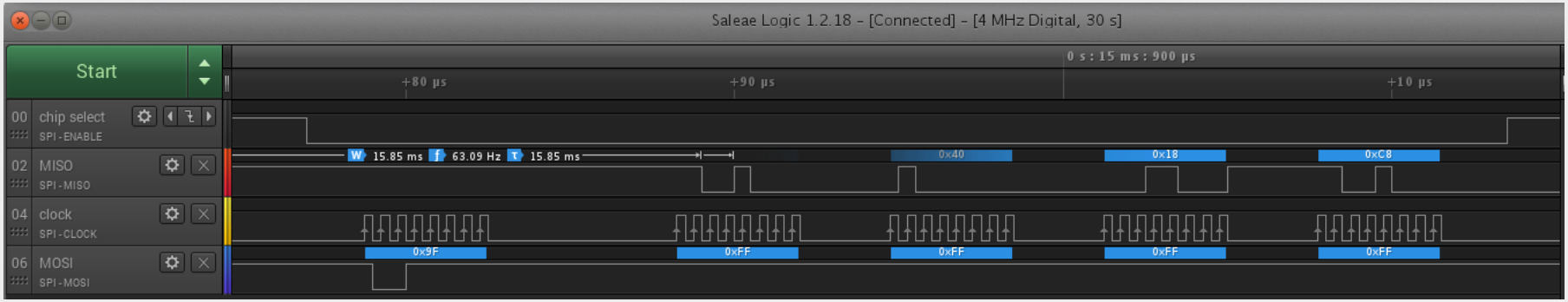

Figure 5: Querying NOR flash chip using SPI protocol

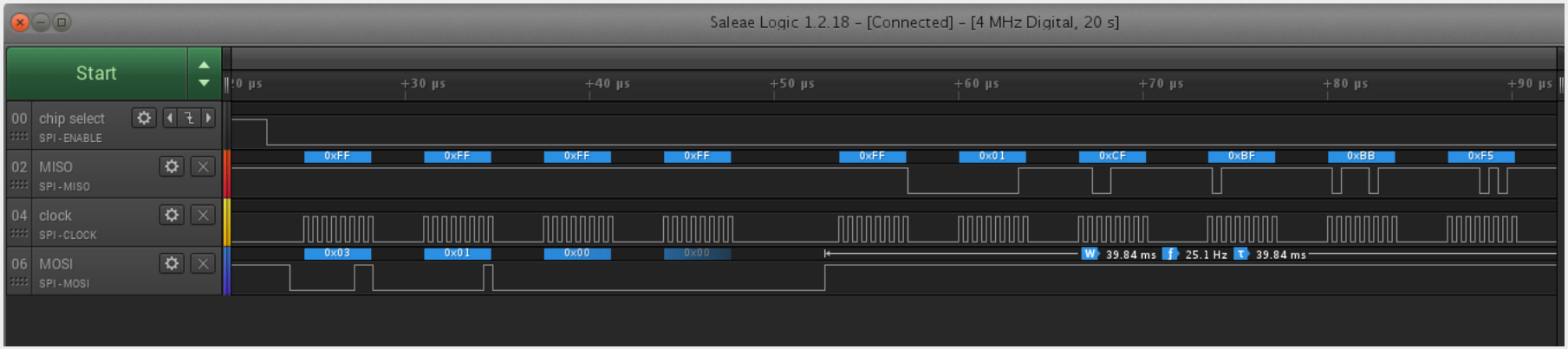

Figure 6: Reading a NOR flash chip using SPI protocol

In-situ programming

Is is possible to read and write SPI NOR flash while still soldered to the circuit board using a test clip.

- The board must be powered down as the programmer will supply power to the chip.
- Programmer may not be providing enough current as others part on the mainboard may be connected to the same VCC line

Leveraging UART, SPI and JTAG for firmware extraction

- In my own experience, a programmer will supply enough current for embedded/IOT devices but external power supply is required when flashing laptops
- Use 2 programmers, one to read/write and the other to supply power
- Use the 3.3V line of an ATX power supply

Dealing with chained SPI chips

Seen when flashing coreboot on a Levovo T530 laptops. This laptop have 2 SPI NOR flash chip, 1 of 8MB and 1 of 4 MB.

- Only one chip must be enabled at a time
- While reading/writing a chip, force the other one's HOLD pin to low

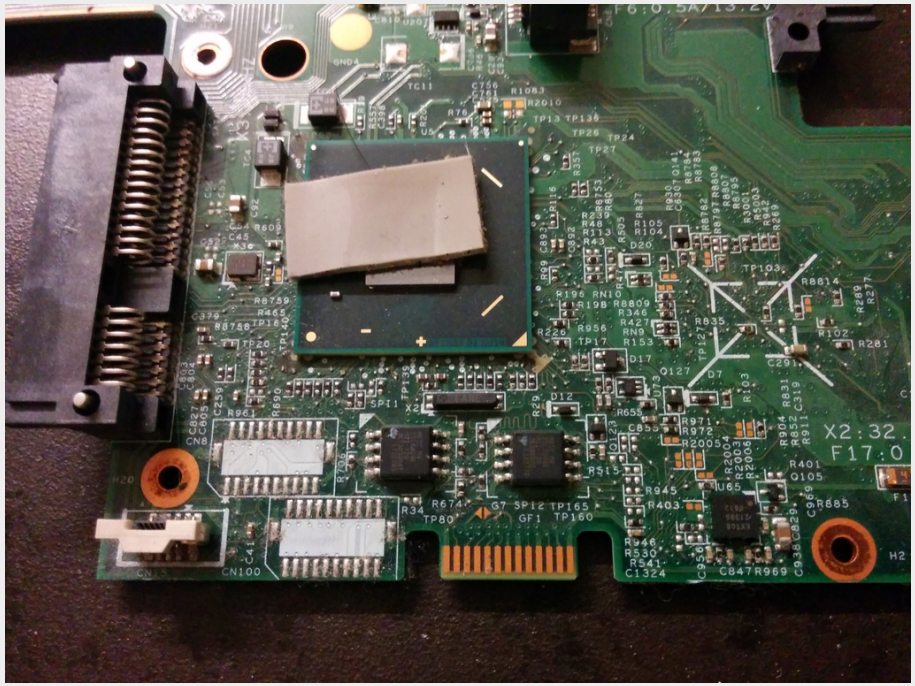

Figure 7: Chained SPI flash chips on Lenovo T530

Other issues with in-situ programming

- hold input can be floating which will prevent the SPI flash chip from being accessed. Pull hold and wp to VCC with resistor.
- If in-situ programming is being attempted, another chip on the board may attempt to access the SPI chip at the same time.
- Hold competing chip (usually CPU) to reset
- Unsolder spi flash chip and use external socket to read/program it

More on SPI protocol

Good to know while troubleshooting

- Commands
  - 0x9f => return vendor id
  - 0x03 => read memory
  - read protection may be enabled, possible to eavesdrop on bus to get unlock sequence ?

Build flashrom tool

- Clone from [https://github.com/flashrom/flashrom]
- Install required packages

```
apt install make build-essential libpci-dev
    libusb-dev libusb-1.0-0-dev autoconf automake
    libtool pkg-config texinfo libftdi1-dev
```

- Build using the make command

SOIC 8 SPI flash pinout

Always lookup voltage and pin configuration in documentation but many SOIC 8 pins flash chip do have the same pinout as Winbond's W25Q64FV.

- GND is at pin 4 (bottom left), VCC at pin 8 (top right)
- CS is pin 1, CLK at pin 6
- DO is at pin 2, goes to programmer's MISO (Master IN, Slave OUT)
- DI is at pin 5, goes to programmer's MOSI (Master OUT, Slave IN)

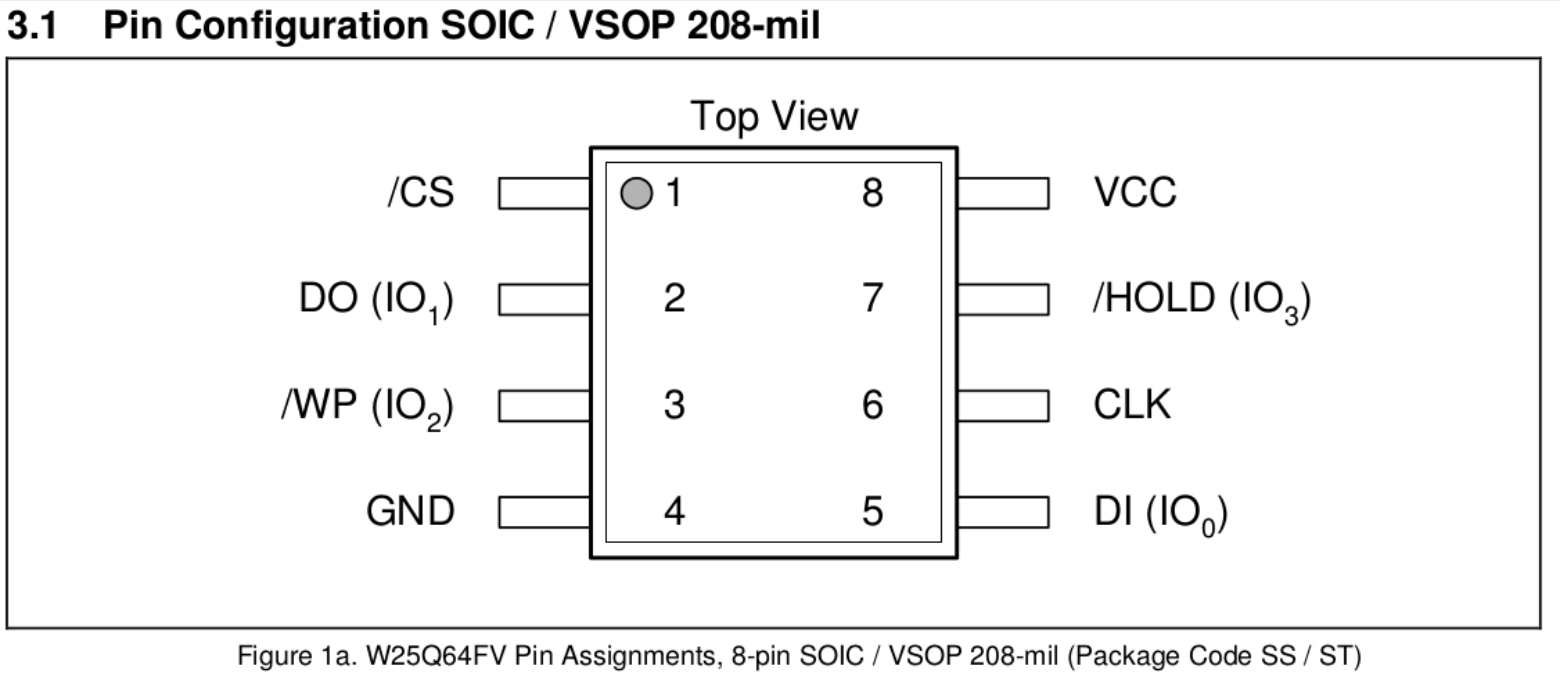

Figure 8: Winbond W25Q64FV pinout

Connecting the SOIC test clip

In this case, the blue SOIC test clip is already wired to the bus pirate

- The brown wire, connected to GND, goes to the bottom left of the chip
- The red wire, connected to VCC, goes to the top right of the chip

Disconnect the power adapter from the AC1200 router and make sure the clip is connected as in the picture in the next slide before you apply power to the bus pirate.

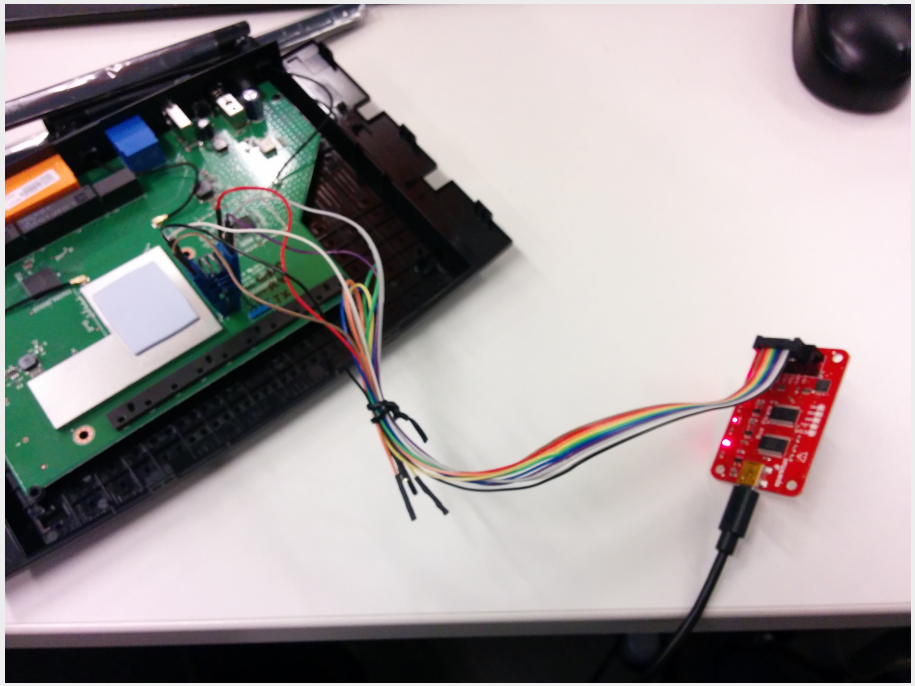

Figure 9: Connecting the SOIC test clip

Using flashrom to access the SPI flash

Connect the bus pirate, pay attention to the port it is assigned by the operating system, usually /dev/ttyUSB0.

Query the SPI flash:

- ./flashrom -p buspirate_spi:dev=/dev/ttyUSB0,spispeed=2M

Read the SPI flash into file ac1900.bin:

- ./flashrom -p buspirate_spi:dev=/dev/ttyUSB0,spispeed=2M -c “GD25Q128C” -r ac1200.bin

SPI Exercises

- Read the whole firmware from the SPI flash into file ac1200.bim
- Extract the linux kernel from the ac1200.bin file
- Extract the root filesystem from the ac1200.bin file

JTAG

- Synchronous serial bus
- TDI, TDO, TCK, TMS and GND, only one data pin for swd
  - Offer CPU debug facility
- Can be used to read flash memory banks

NSEC badge JTAG

In the NSEC 2018 badge, the debugger is implemented on the circuit board and exposed through a serial port over the USB port. There is also physical connectors on the badge's circuit board but the needed cable from Tagconnect is expensive.

Configuring GDB to access the badge's hardware debugger

- install gdb-multiarch
- Clone NSEC badge 2018 repository: https://github.com/nsec/nsec-badge
- cd into nsec-badge/nrf52
- cp flash_config.gdb.template flash_config.gdb
- set target in flash_config.gdb, on linux target should be "target extended-remote /dev/ttyACM0"

Finding information about memory map

Search online about the target CPU for documentation. In this case, search for Nordic nRF52 flash

- https://devzone.nordicsemi.com/tutorials/b/getting-started/posts/adjustr of-ram-and-flash-memory

Search for CPU and reset vector, memory map

Finding information about memory map contd

Also, look for information in the software or SDK related to the circuit board.

In this case, information about flash and ram can be found in nsec_badge.ld

- FLASH (rx) : ORIGIN = 0x23000, LENGTH = 0x5d000

Running gdb-multiarch

gdb-multiarch -x flash.gdb gdb should be able to stop execution on the badge. The program counter (pc register) should be in the flash section, between 0x23000 and 0x80000

```
1  0x0002e5b2 in ?? ()
```

Running gdb-multiarch contd

info reg should show registers

```
1  ...
2  r11            0x0   0
3  r12            0x2000fcf0    536935664
4  sp             0x2000fcc8    0x2000fcc8
5  lr             0x0   0x0
6  pc             0x2e5b2   0x2e5b2
```

Dumping flash memory with gdb attached to the hardware debugger

Use the gdb dump command:

gdb dump file start address stop address

use the help command in gdb for details

NSEC badge Exercises

- Extract flash memory from the badge
- Find flags in the flash memory