

Introduction to Reverse Engineering

Mike Anderson
Chief Scientist
The PTR Group, LLC.
mailto: mike@theptrgroup.com
<http://www.ThePTRGroup.com>

What We Will Talk About...

- What is reverse engineering?
- Why do it?
- HW & SW Tools
- Impediments
- The process
- Knowing when you are done
- Where to from here?



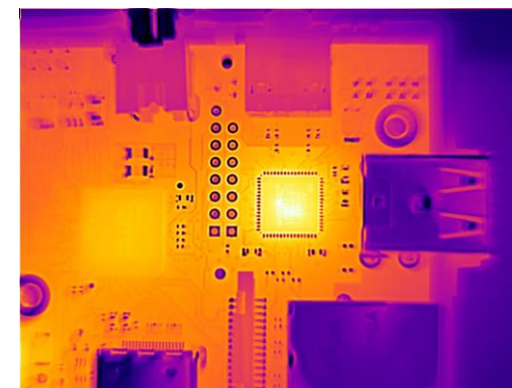
Source: N0where.net

What is Reverse Engineering?

- Given a device or piece of hardware or software, deconstruct it to determine how it was built
- This may entail removing the case, repopulating connectors, disassembling the software
- May require the use of hardware debuggers and other test equipment to determine the nature of the interfaces
- Beware! In some jurisdictions, just going this far can be illegal!
 - In the U.S., we have the DMCA that might preclude any work with the firmware

Why Do RE?

- Goals may be to repair/repurpose/upgrade or maybe just curiosity 😊
 - The Repair Movement is gaining some ground
- Often, projects get lost in the shuffle (device archeology)
 - Developers move on and the documentation may be scant at best
 - Needed modifications might be difficult or impossible
- You may be presented with a suspicious device
 - It may be counterfeit, or have additional circuits embedded
- There maybe some special software on the device that you're concerned about
 - Malware, spyware, govware, etc.



Source: eevblog.com

Tools You Should Have...

- In order to get access to the device, you'll likely need a few | special tools
 - Torx driver set, screw drivers of various sizes, precision utility knife, "spudgers", guitar picks, suction cups, small hex drivers, etc.
 - Thanks to the "Right to Repair" movement, these are available as a kit for those with a little \$\$\$
- In the event they use adhesives to seal the case, heat will often do the trick to soften the glues
 - Heat gun/blow drier, or microwaveable gel
 - Then, spudge away!
- Inspection microscope
 - Useful to examine small part numbers



Source: ifixit.com



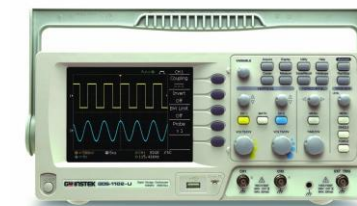
Source: amazon.com

Electrical Test Equipment

- A VOM is a must
 - Spend a little \$\$\$ and get a good one
- A DSO is also handy for examining high-speed signals
 - Dual channel, 50-100 MHz is good enough for most applications
 - Make sure you get high-voltage probes as well
- A 8/16-channel logic analyzer
 - More channels are nice, but often not needed
 - I love my Salaea Logic Pro 8 with the USB 3 interface
- A SiGrok-compatible signal identification interface
 - Like the Bus Pirate or similar



Source: fluke.com



Source: gwinstek.com



Source: salaea.com



Source: seedstudio.com

Logic Analyzers as Protocol Decoders

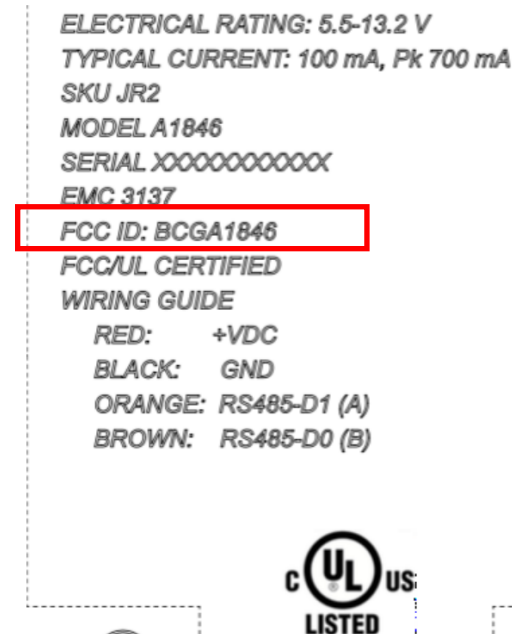
- Many logic analyzers now include protocol decoding
 - I2C, SPI, asynchronous serial, CAN and more
- These can save days of effort in trying to decode the target IDs or chip selects when looking at hardware
- More expensive units can decode PCIe and other high-speed buses



Source: salaea.com

Do Your Research

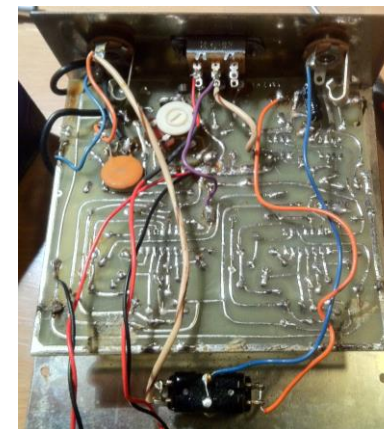
- Who makes the device?
 - Is there an ODM? If so, who is it?
- Is there an FCC registration?
 - Pull the registration info
 - There may be one radio used by many similar devices, one of which has more information available
- Are there patents involved?
 - Patents are often public record, too
- Who are the patent holders?
 - Information about them may prove useful



Source: fccid.io

Opening the Case

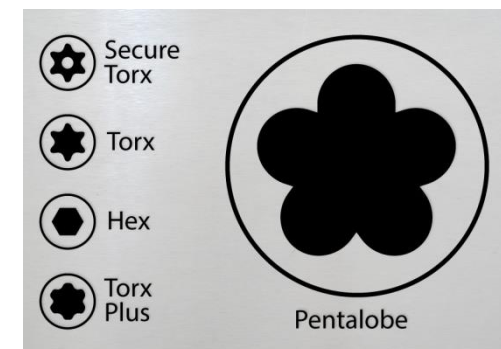
- Just opening the case can often be a challenge
 - Manufacturers want to keep you from seeing what they've done – both good and bad
- Techniques to keep the casual user out of their hardware include:
 - Using special screws
 - Special adhesives
 - Adding anti-tamper sensors
 - Encasing the device in epoxy
 - Also called “potting” the device



Source: ehx.com



Source: teacoinc.com



Source: bobmackay.com

Dealing with Anti-Tamper Switches and Potting

- A well-equipped RE shop will have an X-Ray inspection capability
 - It lets you know what's in the box that may be waiting to trigger
- Anti-tamper case switches can be used to zero-out flash
 - LN2 does a great job at slowing the switches down enough to keep them from triggering until you can control them
- Potting comes in several varieties including polyester and epoxy resins
 - Hard and soft types that use different techniques to remove
 - Oftentimes, heating in an oven will make them pliable
 - Make sure the temperature is less than the melting point of the ROHS solder and plastic connectors (watch for toxic fumes!)
 - Solvents like WD-40, dichloromethane, nitric or sulfuric acid or isopropanol may be required
 - You may need special permits for some of these (again, watch for toxic fumes!)
 - When all else fails, use “cut and scrape” techniques
 - Utility knife and Dremel



Source: safetysign.com



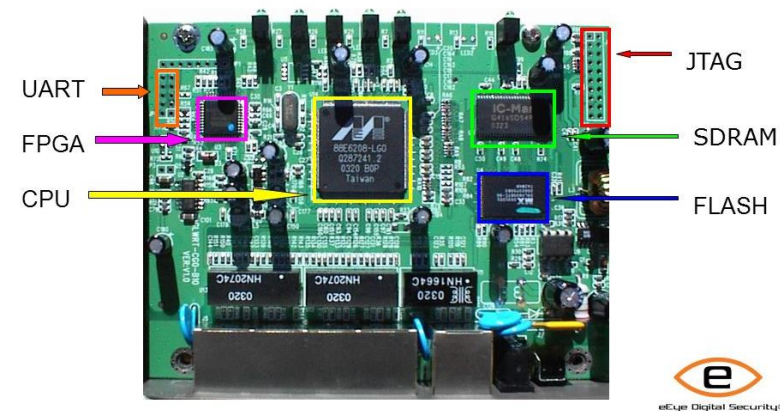
Source: landainternational.com



Examining the Device

- Once you've got it out of the case, take a close look at the device to see if you can identify the parts that are being used
 - For those you can identify, try to obtain the data sheets from the manufacturer
 - Easier said than done in many cases
- Use teardown sites like iFixit.com to see if they've already done a teardown to help you understand the components

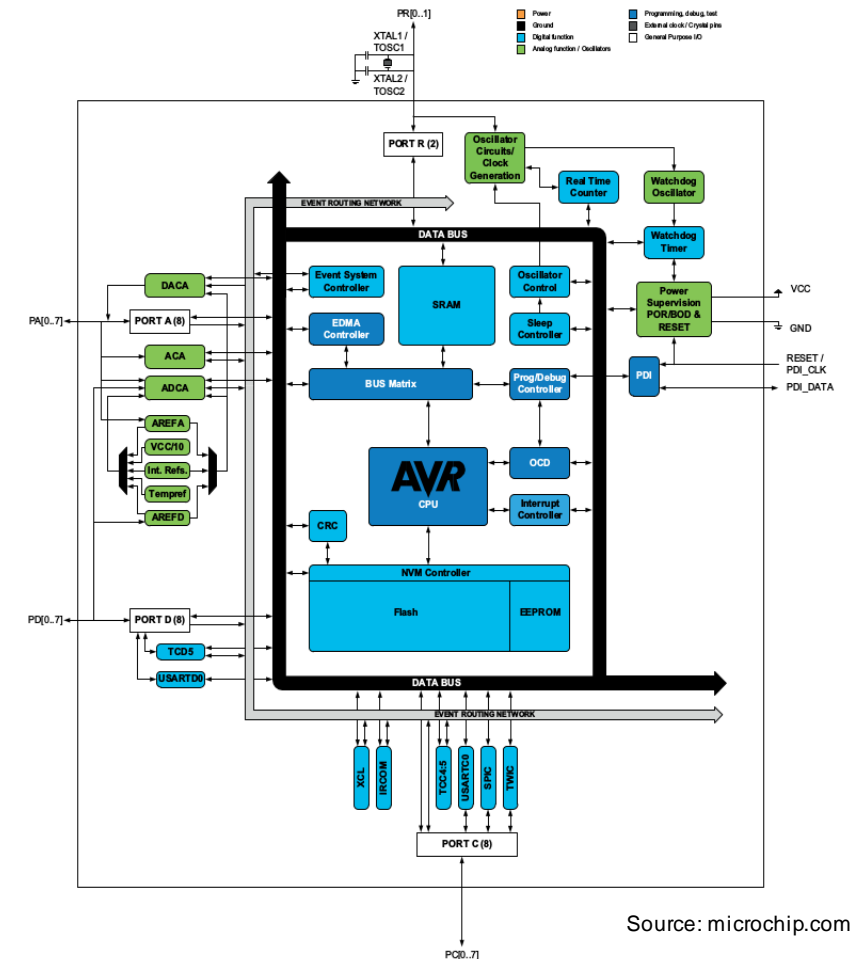
The Internals – D-Link 604 Router



Source: beyondtrust.com

Example Data Sheet

- Accessing the data sheets can help understand the capabilities of the part that may be of use to you
 - E.g., knowing there are 2 U(S)ARTS or that the device supports SPI flash, etc.
 - Logic voltage levels
- The data sheets may also outline what software algorithms are available for the part
 - Like that the part supports a 16-bit CRC engine or has built-in communications protocols



Source: microchip.com

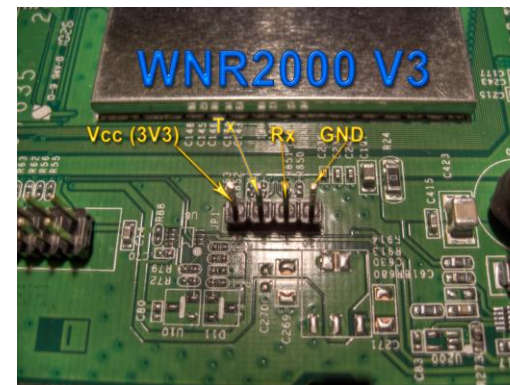
Repopulating Interfaces

- Many manufacturers will depopulate debug and serial interfaces
 - JTAG interfaces often have a familiar look, but serial ports can be elusive
- Use a VOM to measure the voltages
 - Find a good ground on the board to use as a reference
- Be careful about voltage logic levels
 - Using 5V on a 3.3V device will release the magic blue smoke
- Fast signals may not be measurable on a VOM
 - Use the DSO for these
- Or, use a signal tester like the Bus Pirate to determine what kind of signal it is
 - Good tutorials for this at http://dangerousprototypes.com/docs/Bus_Pirate_101_tutorial

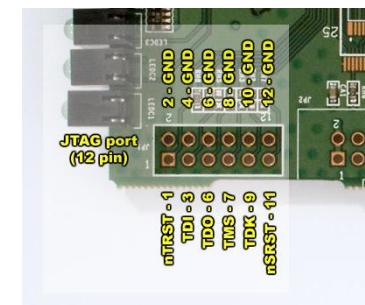


Repopulating Interfaces #2

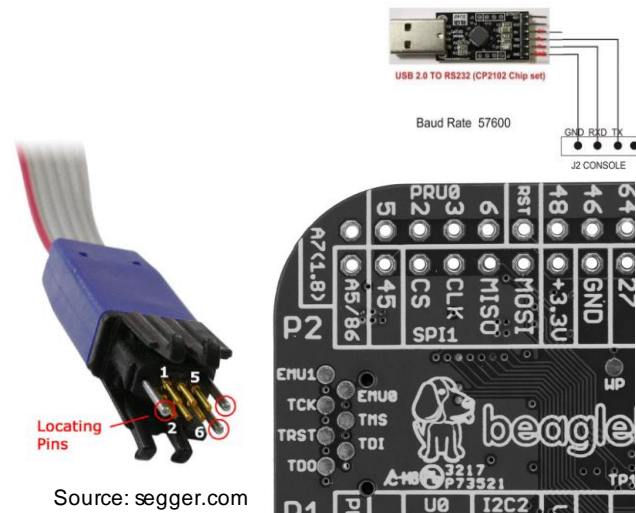
- You might get lucky and the manufacturer has left the solder mask on the board indicating what type of device it was
 - UART, SERIAL, JTAG
- As pin counts go, a typical serial interface requires 3 pins
 - TX, RX & GND
- JTAG requires 5 pins
 - TDI, TDO, TCK, TMS and TRST
 - Often in a 10, 14 or 20-pin configuration
- Serial Wire Debug (SWD) requires only 2 pins
 - Generally targeted at ARM μ Cs but can be extended to larger Cortex-A parts
 - SWO adds one more pin



Source: ;iatoss.com



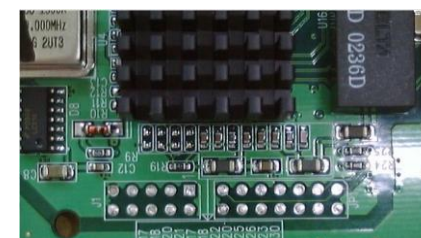
Source: openwrt.org



Source: segger.com



Source: siliceo.es



Source: zibotronicx.com

Why Repopulate the Interfaces?

- For serial ports, the goal is to be able to watch the boot cycle
 - This will provide clues as to what OS it's running and if there is a way to break into the boot cycle without having the development credentials
- For JTAG/SWD, it's about being able to read the firmware out of the boot flash
- Once you have the firmware, you can start the RE of the boot code
 - What boot firmware is it using?
 - Is there a device tree blob?
 - Needed if you're going to update the OS
 - What OS is it using?
 - Which version?

Other Ways to get the OS Image

- If the device is/was being maintained, go to the manufacturer's website and see if there is update firmware available for download
 - If so, download it and let's have a look...
- Depending on the vendor, you might be able to download the update directly, or you may have to go through the device itself to get the update



Firmware Upgrade

Visit upgrade.actiontec.com for upgrade support, upgrade options and information.

Current Version: 40.21.24

Upgrade From the Internet:



Automatic Check Disabled

Check at URL <https://upgrade.actiontec.com/MI42>

Check Now

Status: OK

Internet Version: No new version available

Force Upgrade

Upgrade From a Computer in the Network:



Select an updated Wireless Broadband Router firmware file from a computer's hard drive or CD on the network

Upgrade Now

Once You Have the Image...

- Example: Actiontec MI424WR-GEN3I

```
$ file MI424WR-GEN3I.rmt
MI424WR-GEN3I.rmt: data
$ binwalk MI424WR-GEN3I.rmt
```

DECIMAL	HEXADECIMAL	DESCRIPTION
163	0xA3	uImage header, header size: 64 bytes, header CRC: 0x70F0614A, created: 2014-02-03 21:18:38, image size: 6506664 bytes, Data Address: 0x1000000, Entry Point: 0x1000000, data CRC: 0x42627A27, OS: Linux, CPU: ARM, image type: OS Kernel Image, compression type: none, image name: "OpenRG"
12591	0x312F	gzip compressed data, maximum compression, from Unix, last modified: 2014-02-03 21:18:36
2159504	0x20F390	LANCOM WWAN firmware

- Now, we know quite a bit about what's going on the device:
 - We know that it's an ARM-based Linux device with U-Boot as the boot loader
 - Given the date, this may be pre-device tree
- Using `dd`, we can dissect the image



Can you guess what these are?

Now, Separate the Pieces

```
$ dd if=MI424WR-GEN3I.rmt of=uboot bs=1 skip=163 count=12428
12428+0 records in
12428+0 records out
12428 bytes (12 kB, 12 KiB) copied, 0.0407348 s, 305 kB/s
```

```
$ dd if=MI424WR-GEN3I.rmt of=os.gz bs=1 skip=12591 count=2146913
2146913+0 records in
2146913+0 records out
2146913 bytes (2.1 MB, 2.0 MiB) copied, 2.19151 s, 980 kB/s
```

```
$ dd if=MI424WR-GEN3I.rmt of=LANCOM.FW bs=1 skip=2159504
4347387+0 records in
4347387+0 records out
4347387 bytes (4.3 MB, 4.1 MiB) copied, 4.4023 s, 988 kB/s
```


Start Poking Around...

- Now, let's decompress the OS image and see what we've got:

```
$ gunzip os.gz
$ binwalk os
```

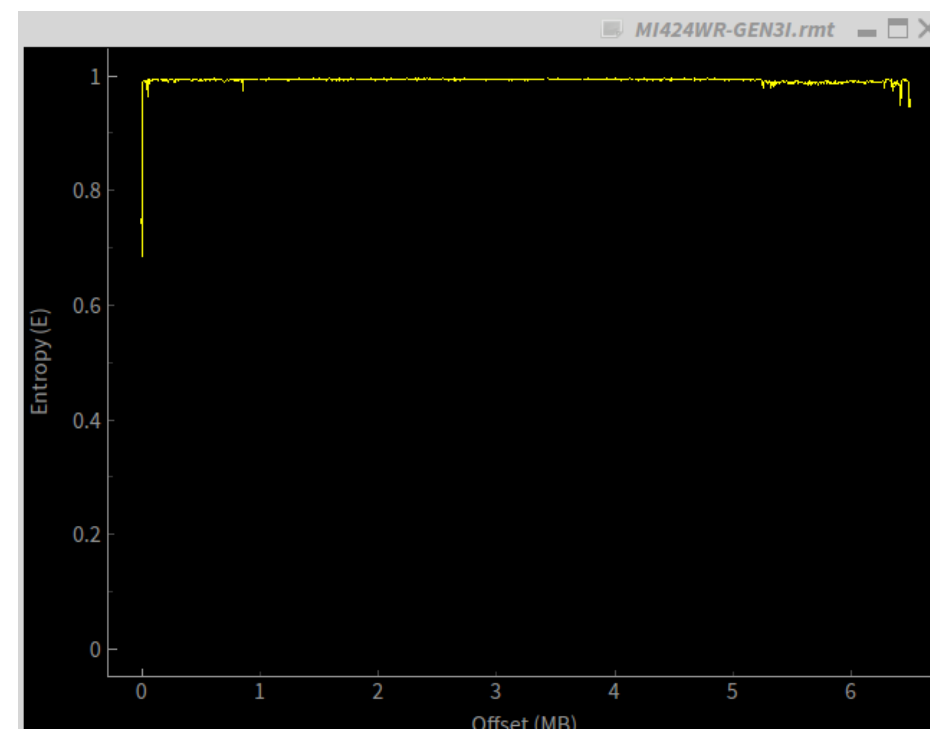
DECIMAL	HEXADECIMAL	DESCRIPTION
72832	0x11C80	gzip compressed data, maximum compression, from Unix, last modified: 2014-02-03 21:12:45
888832	0xD9000	CramFS filesystem, little endian, size: 3866624 version 2 sorted_dirs CRC 0x0A9D581F, edition 0, 549 blocks, 451 files
4755456	0x489000	CramFS filesystem, little endian, size: 589824 version 2 sorted_dirs CRC 0x2DF0DBD1, edition 0, 52 blocks, 34 files
5455117	0x533D0D	Certificate in DER format (x509 v3), header length: 4, sequence length: 1284
6546105	0x63E2B9	Certificate in DER format (x509 v3), header length: 4, sequence length: 5436
6546181	0x63E305	Certificate in DER format (x509 v3), header length: 4, sequence length: 5436
6606637	0x64CF2D	Certificate in DER format (x509 v3), header length: 4, sequence length: 4099
6622237	0x650C1D	Certificate in DER format (x509 v3), header length: 4, sequence length: 3
6638405	0x654B45	Certificate in DER format (x509 v3), header length: 4, sequence length: 3
6917709	0x698E4D	Certificate in DER format (x509 v3), header length: 4, sequence length: 5512
6951609	0x6A12B9	Certificate in DER format (x509 v3), header length: 4, sequence length: 5568
6960481	0x6A3561	Certificate in DER format (x509 v3), header length: 4, sequence length: 5552
6960525	0x6A358D	Certificate in DER format (x509 v3), header length: 4, sequence length: 5548
6960569	0x6A35B9	Certificate in DER format (x509 v3), header length: 4, sequence length: 1476
7378672	0x7096F0	Linux kernel version "2.6.16.14feroceon #1 Mon Feb 3 13:18:27 PST 2014"
7385656	0x70B238	CRC32 polynomial table, little endian
7386771	0x70B693	Copyright string: "Copyright 1995-1998 Mark Adler "
7401144	0x70EEB8	Unix path: /home/bhr/Rev-I/Verizon/tag-bhr-revI-ipv6-40-21-10-3/bhr/rg/os/linux-2.6/init/main.c
7403432	0x70F7A8	Unix path: /home/bhr/Rev-I/Verizon/tag-bhr-revI-ipv6-40-21-10-3/bhr/rg/os/linux-2.6/arch/arm/kernel/irq.c
7405796	0x7100E4	Unix path: /home/bhr/Rev-I/Verizon/tag-bhr-revI-ipv6-40-21-10-3/bhr/rg/os/linux-2.6/arch/arm/kernel/traps.c

...

General Approach to RE a Binary

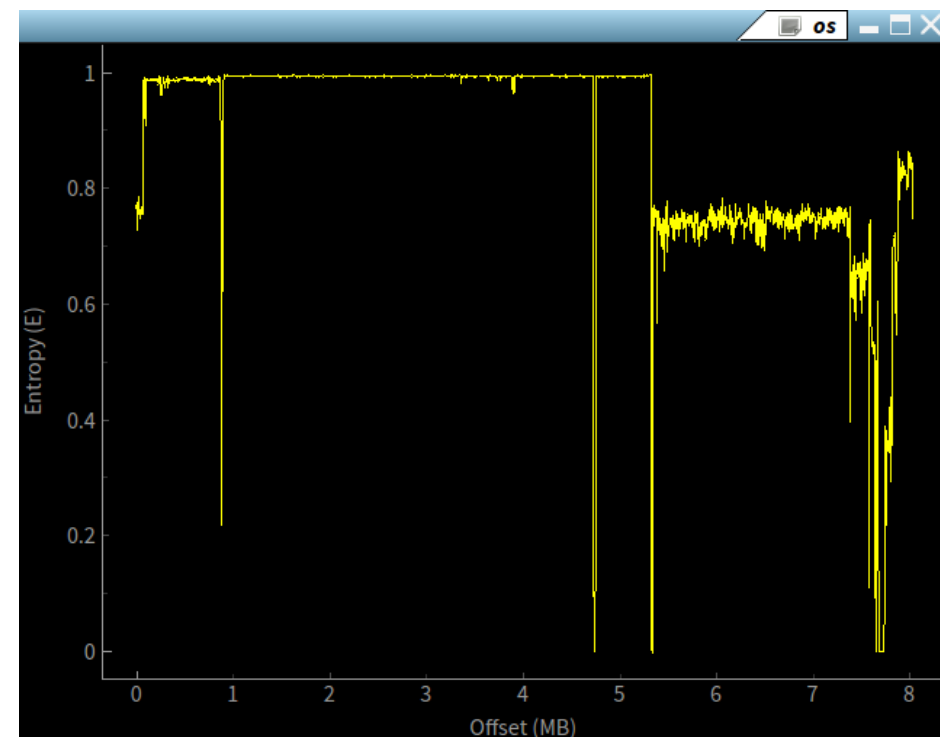
- We got really lucky with this example 😊
 - Most firmware is not so forgiving
- The general approach is to first try to assess the binary
 - Look at the entropy of the binary using **binwalk**
 - Entropy near 1 means it's either compressed or encrypted
- Use **strings** to look for printable character sequences:

```
$ binwalk -E MI424WR-GEN3I.rmt  
$ strings MI424WR-GEN3I.rmt | more  
start section  
rg_hw: FEROCION  
dist: FEROCION  
vendor: VERIZON  
prod_version: 4.7.5.3.31.2.19  
version: 40705  
ext_ver: 40.21.10.3
```



Next Step...

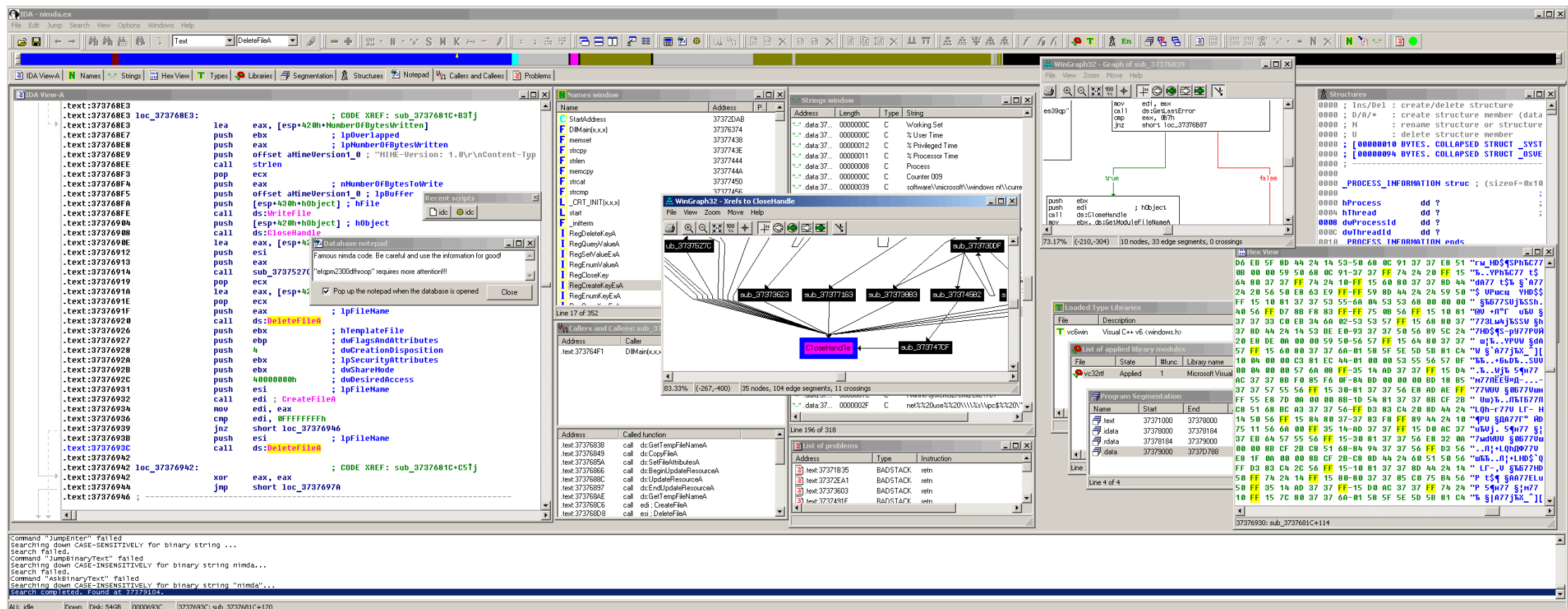
- Once you think you've got it separated out a bit, take another look into the entropy
 - After decompressing...
- We know from the **binwalk**, that there are 2 CRAMFS images on the front
 - We see that in the entropy
- We can separate them out and mount them using loopback mounts to take a peek inside
 - Left for another time...



Looking Closer at the Binaries

- There are a number of additional tools for looking at binaries
 - Binutils like **objcopy**, **objdump**
 - Remember that **objdump** has a disassemble option
 - ELF utilities like **readelf**
- Use tools like **strace** and **ltrace** to watch the execution
 - Run the applications in a VM or chroot jail if you're unsure of what they do
- Disassemblers such as the ERESI project
 - <https://github.com/thorkill/eresi>
- Professional tool chains/disassemblers like IDA Pro
 - <https://www.hex-rays.com/products/ida/>
- You can invest a lot of time and \$\$\$ in this if you really want to

IDA Pro



Always use Protection...

- Never run a foreign binary on your test platform without taking significant precautions
- Using QEMU is a good start
 - Support for most of the common CPU varieties
- Or, use a VM like KVM to keep the application bottled up
- At a minimum, use a chroot/LXC session to show a small distro of files that might make the application wake up
- Capture the run with strace/ltrace to see what functions it's using
 - Note any anomalous behavior

Always use Protection... #2

- Alternatively, transfer the application to a small platform like a BeagleBone or Raspberry Pi to run.
 - If something goes wrong, pull the plug
- You can always reformat the SD card and start again
 - Or, can you? 😊
- In general, don't run with IP enabled (pull the Ethernet cable) or go into IPTables and block all outbound traffic until you have a warm fuzzy that it's OK
- If you believe that the application isn't doing anything odd, advance the time by a month, quarter, 6 months and a year to see if something wakes up to beacon out to the bad guys

Example RE of a Protocol

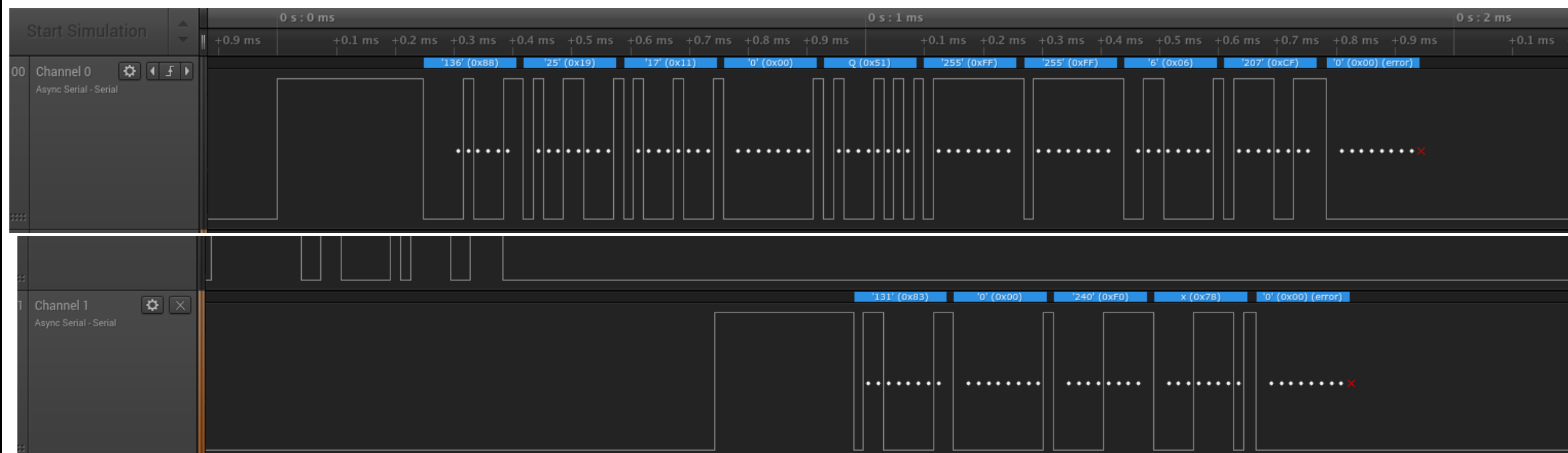
- In some cases, we're not as interested in the device itself as we are the communication protocols
 - E.g., Ethernet cabling does not immediately imply they're using Ethernet
 - We have to check it to make sure before plugging it into our test boards
- Serial protocols are significantly difficult to RE
 - Dealing with an ancient technology that few folks still understand
- The use of logic analyzers with protocol decoders is your savior here

Check your voltages

- There are several options for serial protocols when it comes to voltages
 - 5V, 3.3V, 1.8V at a minimum
- Serial ports traditionally have power on either the TX or the RX lines that you can test for voltages
 - VOM might work OK if the voltages are steady
- If there's any weirdness, you might have to resort to the oscilloscope
 - Use the high-voltage probe just to be safe
- See if you can get the device to send any data and watch the output



Try to Capture Some Data



- The bit width told us it was ~56Kbs, but there appeared to be some drift

Tinker with the Protocol Decode

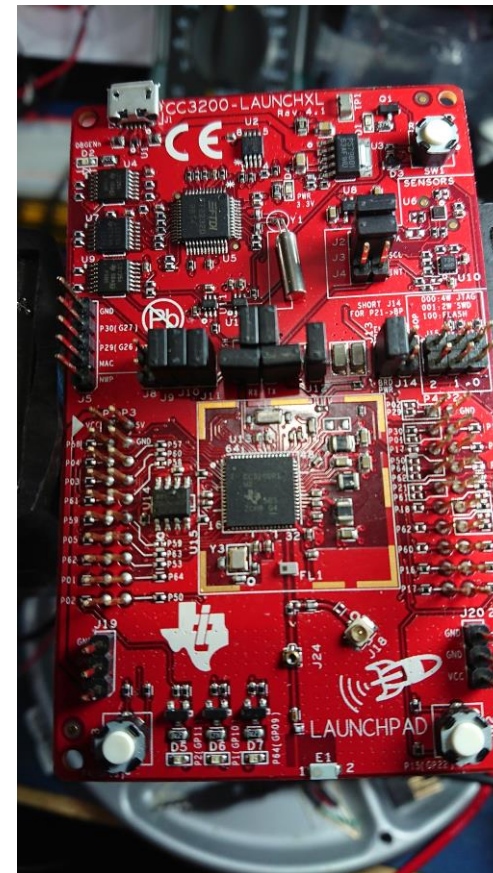
- If your logic analyzer supports multiple protocols, try switching the interpretation of the data into different protocols to see if anything makes sense
- How many pins appear to have a signal on them?
 - 2 pins might be a simple serial line
- Look at the protocol capture to see if there appears to be a clock
 - Could mean I2C to other simple bus
- SPI is another possibility
- We were afraid it was bit-banged
- In this case, it was serial – with a twist

Strange Quirk to the Signal

- Because the voltages were at 0V most of the time and then came alive, it looked like one of the protocols that does a Break-after-Mark
- Closer examination showed that it didn't follow the typical B-A-M pattern
- It turned out that the developers were trying to save battery power by turning the device off between signaling and used a high voltage (looked like a BREAK signal) to wake up the circuitry and then start clocking data
- This means that if you want to inject a new command, you'll have to follow the same pattern or the device battery will die quickly

Using a Microcontroller to do testing

- Microcontrollers are great in that we have I2C, SPI, serial, PWM and GPIO
- The micro we used was a TI CC3200
 - Right serial voltage level at 3.3V
- This also gave us Wi-Fi and JTAG so we could have some options for interfacing with the micro
- We opted to use the Arduino-like Energia as the control software
 - Simple to work with



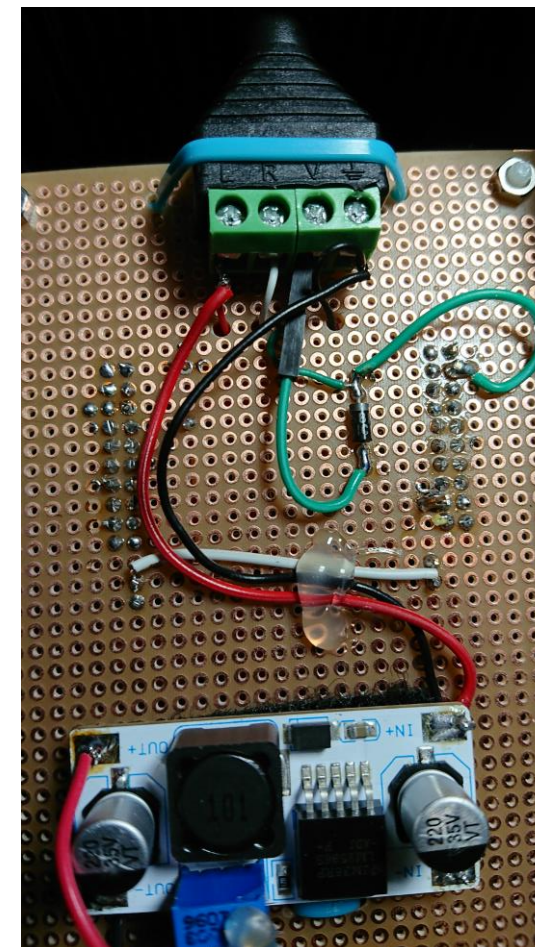
First Try...

- But the first try to inject a command failed miserably
 - We knew we needed a diode to keep the signal from going back into the micro
- We took a look at the o'scope and we were not pleased
- The voltage was cut in half and the edges looked horrible



Different Diode Fixed it

- What we needed was a high-speed schottky diode
 - Handles faster voltage transitions
- And, we needed to power the micro and keep the signal on the same reference ground
- We were able to tap into the power being supplied by the device itself through a buck transformer to power the micro
 - It kept us on the same reference ground
- This allowed us to inject commands and succeed!



Summary

- Reverse engineering is an incredibly challenging problem
 - Lots of reasons you might want to do it
- Make sure you gather your tools
 - Hardware and software tools
- Understand what your goals are and when to declare victory
- If you really like to do this kind of thing, the Repair Movement could use your help
 - Commercial RE is also a thing
- Understand the legal implications of what you're doing in the local jurisdiction
 - Just because you own the device does not mean you can do anything you want

Questions?

