Introductory Firmware Reverse Engineering

Canon Selphi printer

Aleksandar Nikolic, BSides PDX 2025

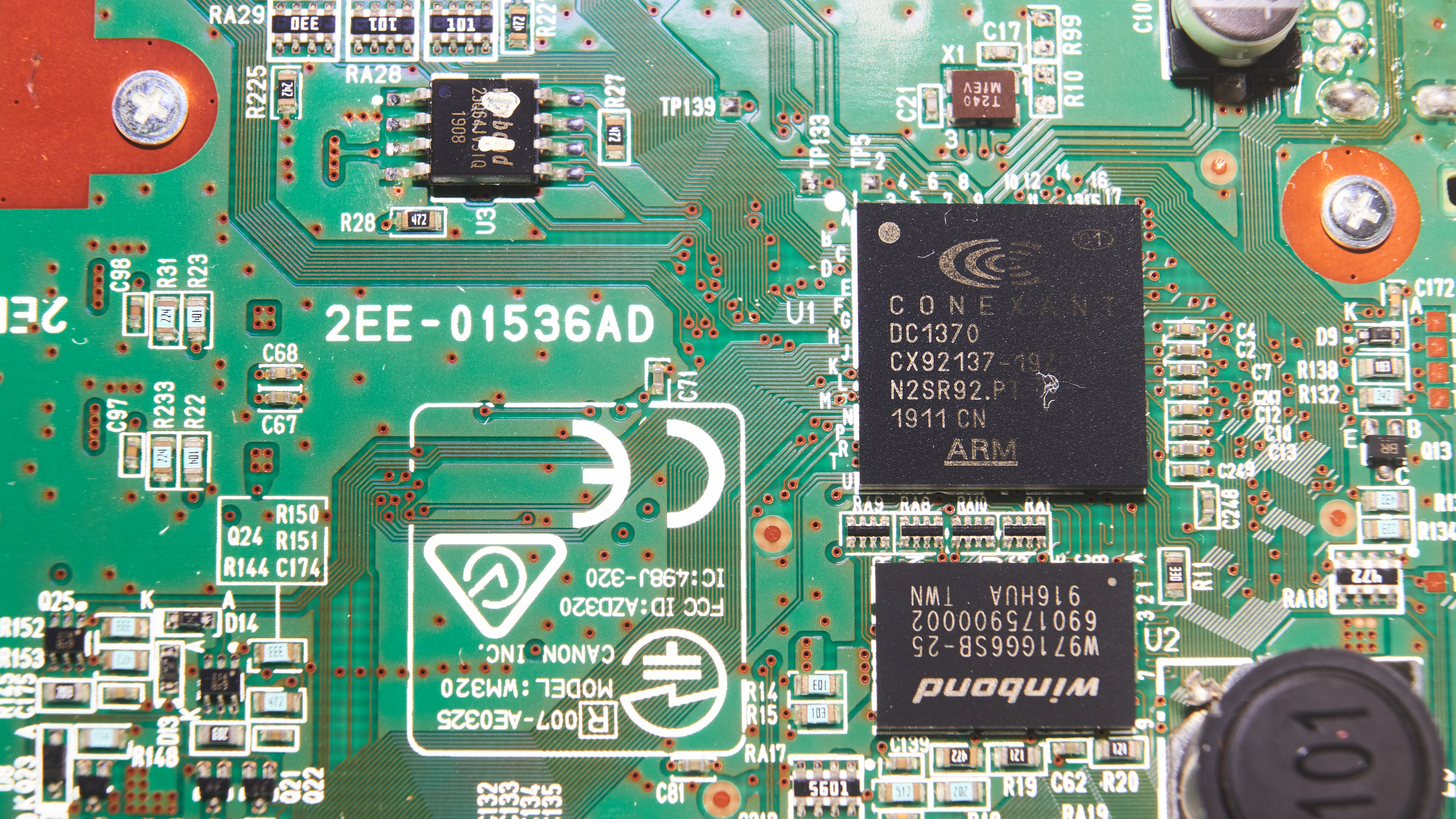
Tools to install

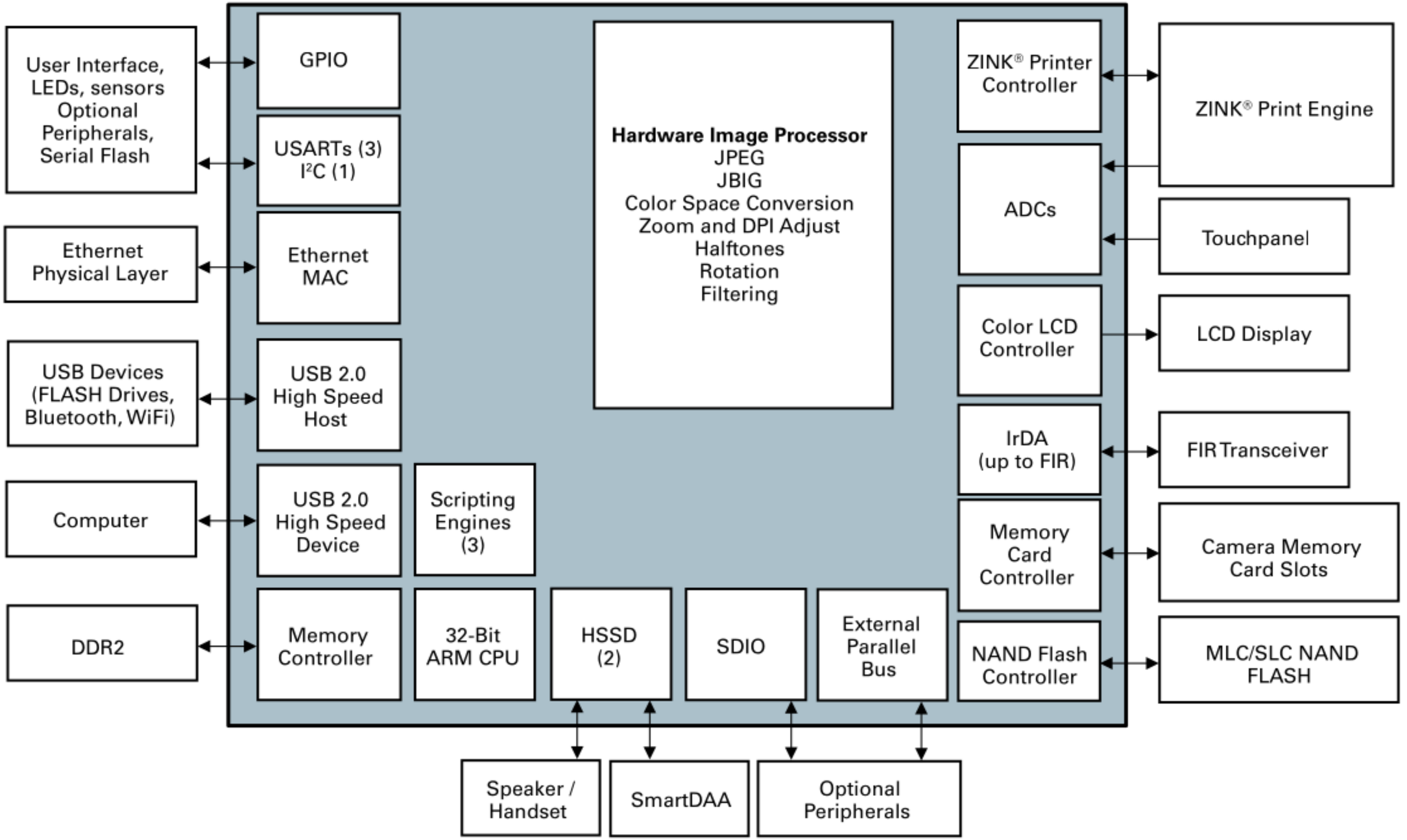
- Ghidra
  - https://github.com/NationalSecurityAgency/ghidra/releases/tag/Ghidra_11.4.2_build
  - May need JRE as well
- Binwalk
  - Probably through your distribution
- ImHex
  - https://imhex.werwolv.net/

Workshop materials

- https://github.com/ea/selphy_workshop
-


ON
HOME
MENU
OK
COPIES +
BACK
PRINT
SELPHY
CP1300
Wi Fi

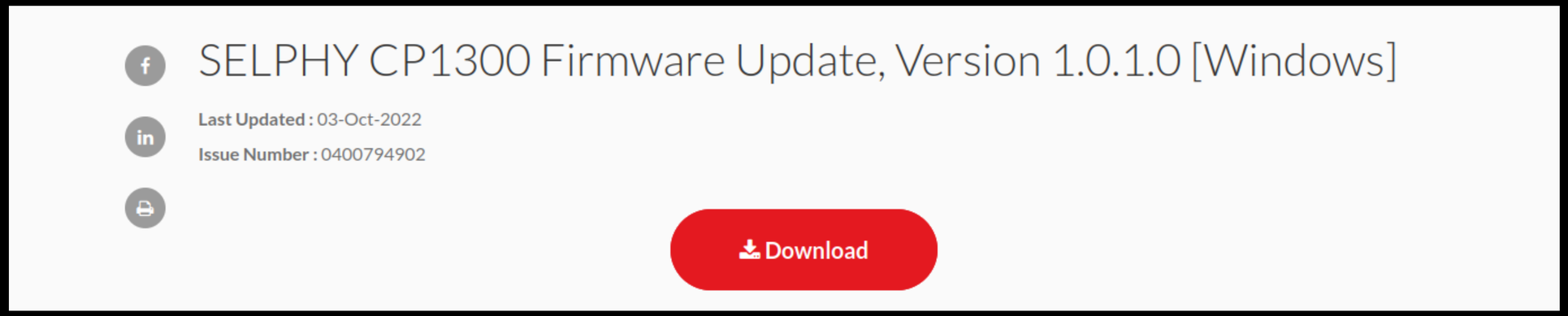

2EE-01536AD
CONEXANT
DC1370
CX92137-19
N2SR92.P1
1911 CN
ARM
winbond
W9716G6SB-25
916HUA TWN
FCC ID:AZDJ20
IC:498J-320
CANON INC.
MODEL:WM320
R007-AE0325
U1
U3
X1
T240
MIEV
TP139
TP133
TP5
RA29
RA28
RA17
RA18
R225
R28
R27
R10
R99
C17
C21
C98
C97
C68
C67
C71
C81
C172
D9
R138
R132
Q13
R150
Q24
R151
R144
C174
Q25
D14
R152
R153
R148
Q21
Q22
R14
R15
R19
C62
R20
C139
R11
C4
C2
C7
C12
C10
C13
C248
C249



CX92137 Block Diagram

Step 0: Dive right in

- Grab the firmware
- Run it through binwalk
- https://my.canon/en/support/0400794902
-



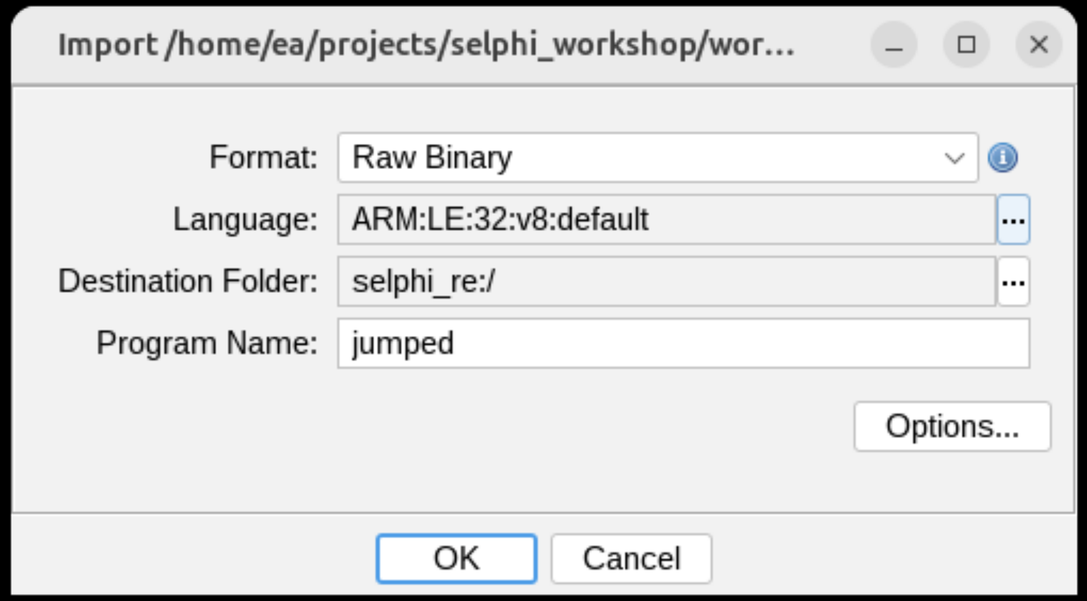
Binwalk

- Recognize known file types
- Recursively extract known archive file formats
- binwalk -e

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
183886        0x2CE4E         Copyright string: "Copyright (c) 1996-201
656892        0xA05FC         gzip compressed data, from NTFS filesyste
3941358       0x3C23EE        MySQL ISAM index file Version 7
3946106       0x3C367A        MySQL MISAM index file Version 7
3960252       0x3C6DBC        MySQL MISAM index file Version 2
3960455       0x3C6E87        MySQL MISAM index file Version 2
```

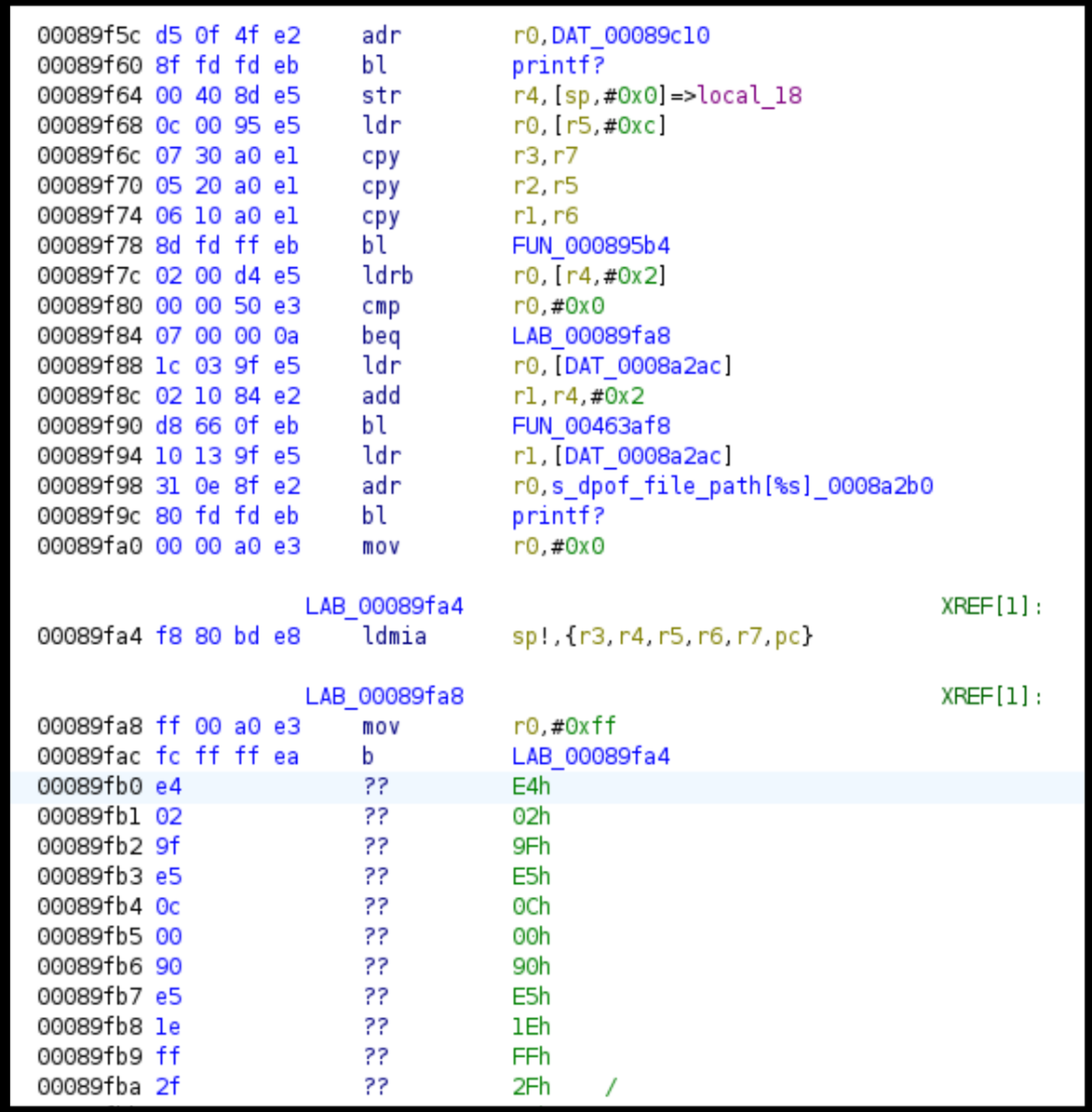
Step 1: Load into Ghidra

- We'll talk about details later...



Key terms: Disassembly

- Turning bytes into readable assembly code
- Non-trivial – often imperfect
- We don’t know what’s data and what’s code
-



Key terms: Decompiling

```c
main(int argc, char **argv){
        int sum_of_even = 0;
        int sum_of_odd = 0;
        for(int i = 0; i < atoi(argv[1]); i++){
                if(i % 2 == 0) sum_of_even += i;
                else sum_of_odd +=1;
        }
        printf("even %d, odd %d\n"
                        ,sum_of_even, sum_of_odd);
}
```

```
                        LAB_0040188f
0040188f 8b 45 fc       MOV     EAX,dword ptr [RBP + -0x4]
00401892 83 e0 01       AND     EAX,0x1
00401895 85 c0          TEST    EAX,EAX
00401897 75 08          JNZ     LAB_004018a1
00401899 8b 45 fc       MOV     EAX,dword ptr [RBP + -0x4]
0040189c 01 45 f4       ADD     dword ptr [RBP + -0xc],EAX
0040189f eb 04          JMP     LAB_004018a5

                        LAB_004018a1
004018a1 83 45 f8 01    ADD     dword ptr [RBP + -0x8],0x1

                        LAB_004018a5
004018a5 83 45 fc 01    ADD     dword ptr [RBP + -0x4],0x1

                        LAB_004018a9
004018a9 48 8b 45 e0    MOV     RAX,qword ptr [RBP + -0x20]
004018ad 48 83 c0 08    ADD     RAX,0x8
004018b1 48 8b 00       MOV     RAX,qword ptr [RAX]
004018b4 48 89 c7       MOV     RDI,RAX
004018b7 b8 00 00       MOV     EAX,0x0
         00 00
004018bc e8 4f 2c       CALL    FUN_00404510
```

```c
undefined8 FUN_00401865(undefined8 param_1,long param_2)

{
  int iVar1;
  int local_14;
  int local_10;
  uint local_c;

  local_14 = 0;
  local_10 = 0;
  local_c = 0;
  while( true ) {
    iVar1 = FUN_00404510(*(undefined8 *)(param_2 + 8));
    if (iVar1 <= (int)local_c) break;
    if ((local_c & 1) == 0) {
      local_14 = local_14 + local_c;
    }
    else {
      local_10 = local_10 + 1;
    }
    local_c = local_c + 1;
  }
  FUN_00405770("even %d, odd %d\n",local_14,local_10);
  return 0;
}
```

Key terms: Symbols

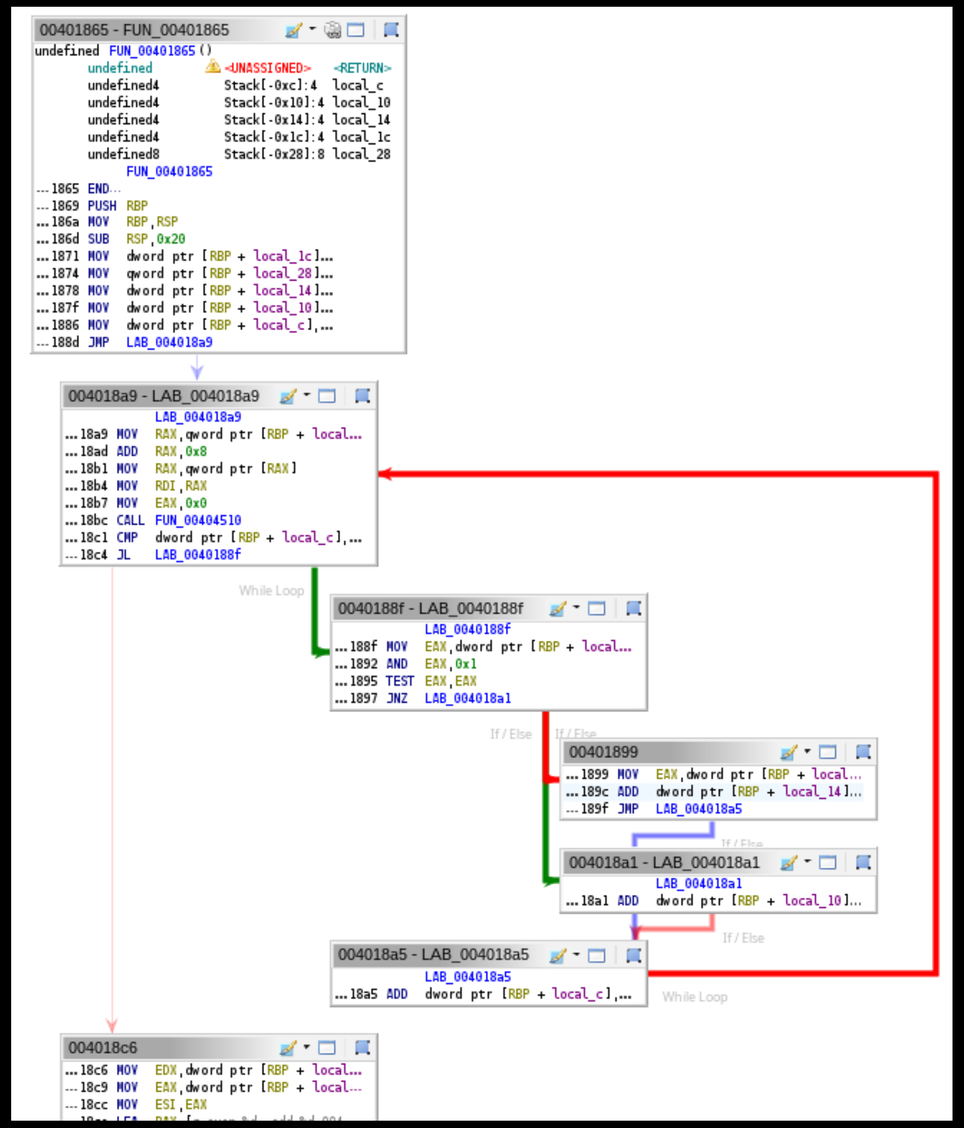
- Function names
- Variable names
- Type information
  - Type names, structure definitions, sizes…

```
const char *callReplyGetBigNumber(CallReply *rep, size_t *len) {
    callReplyParse(rep);
    if (rep->type != REDISMODULE_REPLY_BIG_NUMBER) return NULL;
    *len = rep->len;
    return rep->val.str;
}
```

- With firmware , we usually don’t have them

Key terms: basic blocks

- Functions are graphs of basic blocks
- Jump instructions form edges



Hopefully, the analysis is done.

Ghidra UI

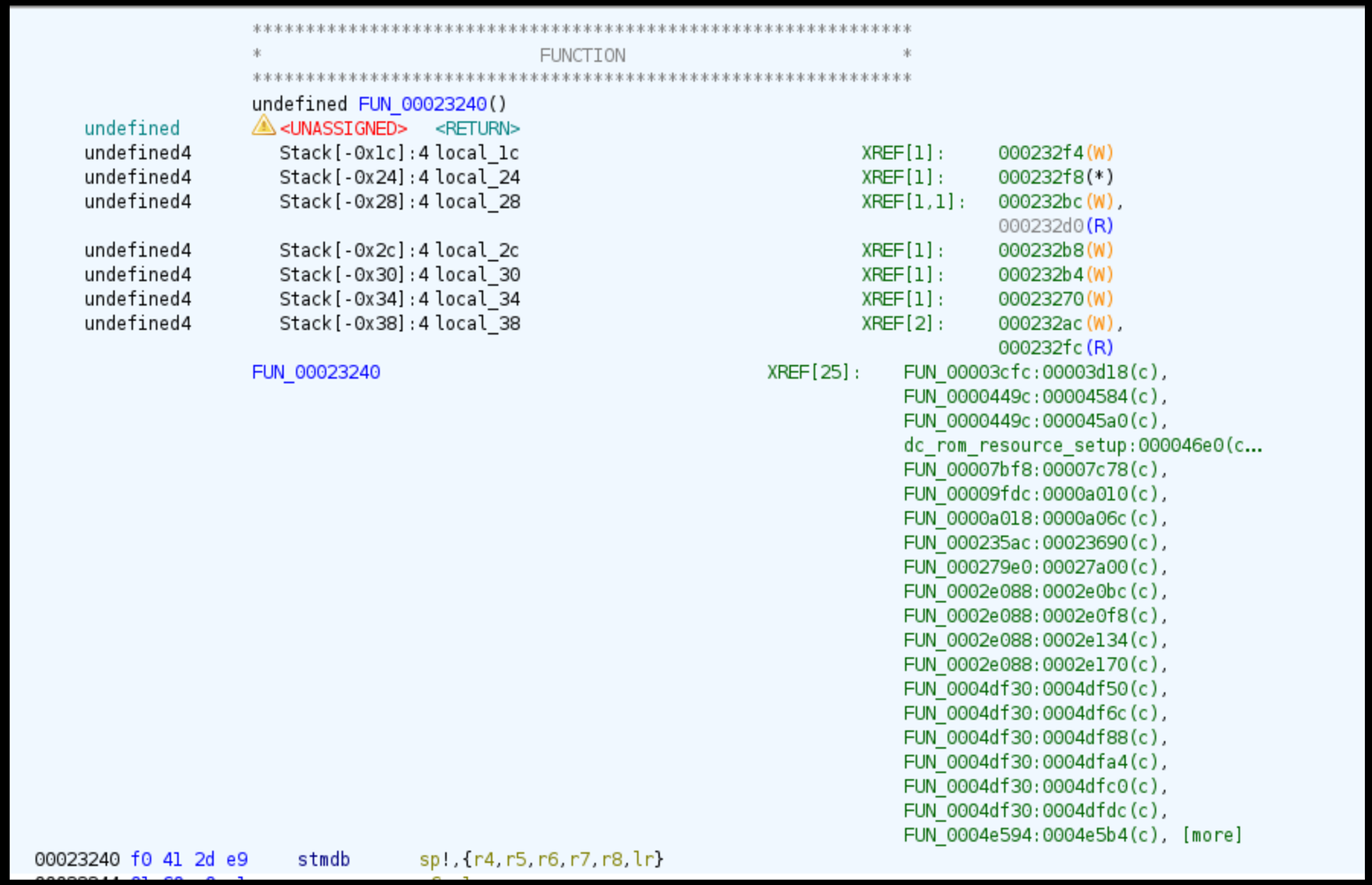
- Config and overview
- Emphasis on decompiler & XREFS

Key concepts: XREFS

- How we navigate through the binary
- Think in graphs!
- X has a relationship with Y
- Function A calls function B
- Instruction I uses memory address C
-
- Without reference resolution and propagation, a tool is useless

XREFs in Ghidra

- Not visible in decompiler



Recovering context

- Easy clues
  - Symbols?
  - Strings?
  - Well known functions
- Harder
  - Datasheets, architecture manuals

Naming things

- Adopt a convention and stick to it
- Function names – guess but append “?” at the end
- Variables – give hints about type – “psMsg”
-
- Keep guessing, refine as you get more info

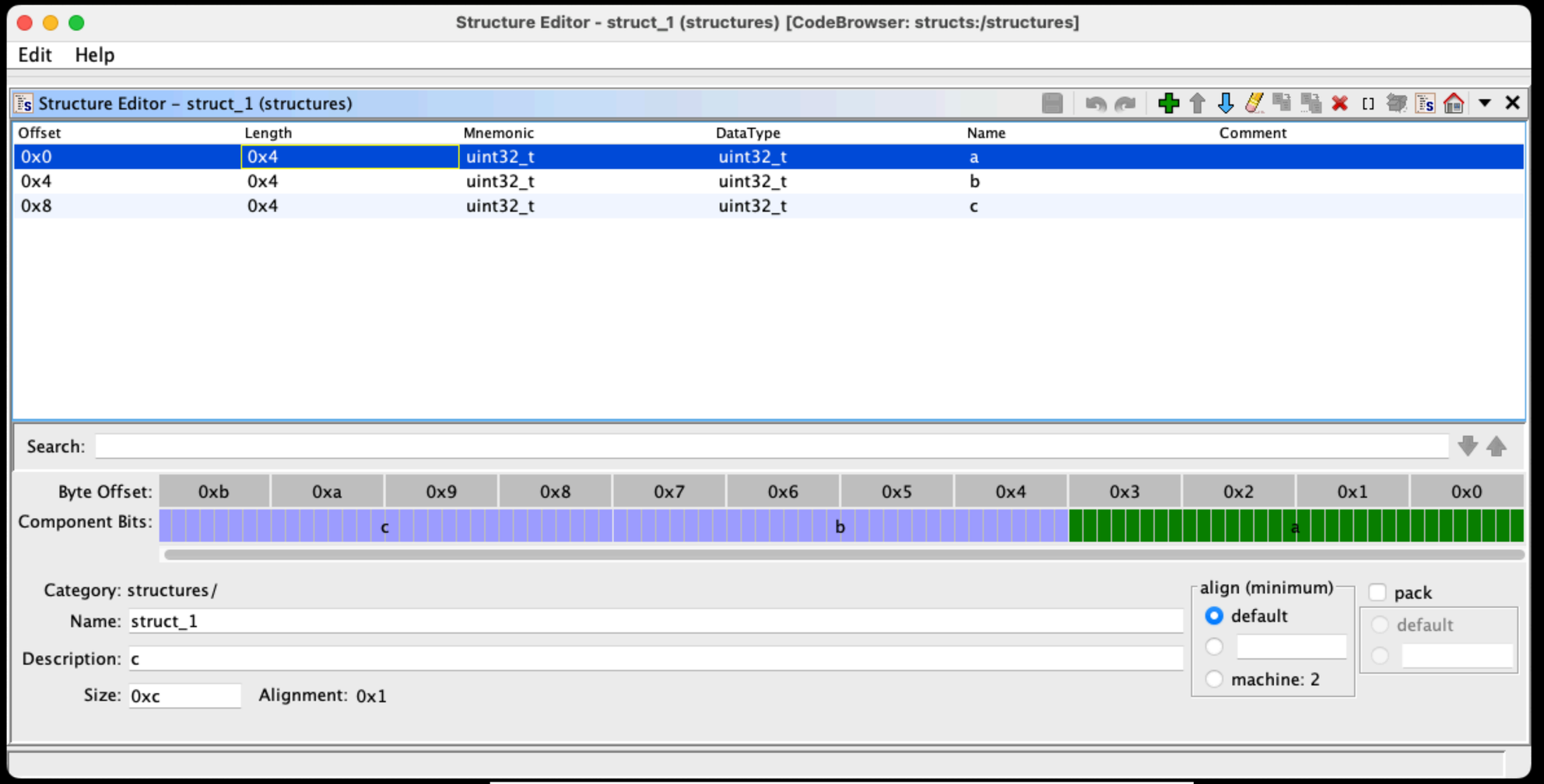
Key concept: Structures

- C Structs are everywhere
- Data accessed at offset into structure
- Crucial for references
- All languages map to them

```
struct msg_hdr {
        int size;
        int offset;
        int value;
};
```

```c
#include <stdio.h>

struct test_struct {
    int size;
    int offset;
    int value;
};


void print_struct(struct test_struct *a){

    printf("size %x, offset %x, value %d\n",
            a->size, a->offset, a->value);
}

int main(int argc, char **argv){
    struct test_struct a;
    a.size = 10;
    a.offset = 0xff;
    a.value = -4;
    print_struct(&a);
}
```

```c
undefined8 FUN_001011a6(void)

{
  long in_FS_OFFSET;
  undefined4 local_1c;
  undefined4 local_18;
  undefined4 local_14;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_1c = 10;
  local_18 = 0xff;
  local_14 = 0xfffffffc;
  FUN_00101169(&local_1c);
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not
    __stack_chk_fail();
  }
  return 0;
}
```




```
test local_1c;

lVar1 = *(long *)(in_FS_OFFSET + 0x28);
local_1c.a = 10;
local_1c.b = 0xff;
local_1c.c = 0xfffffffc;
FUN_00101169(&local_1c);
```

Let’s go digging through the FW.

Key concept: Memory map

- Not everything starts at 0
- Firmware load address?
- Flash memory?
- Peripherals?
-
- Ideally you have it, we don’t

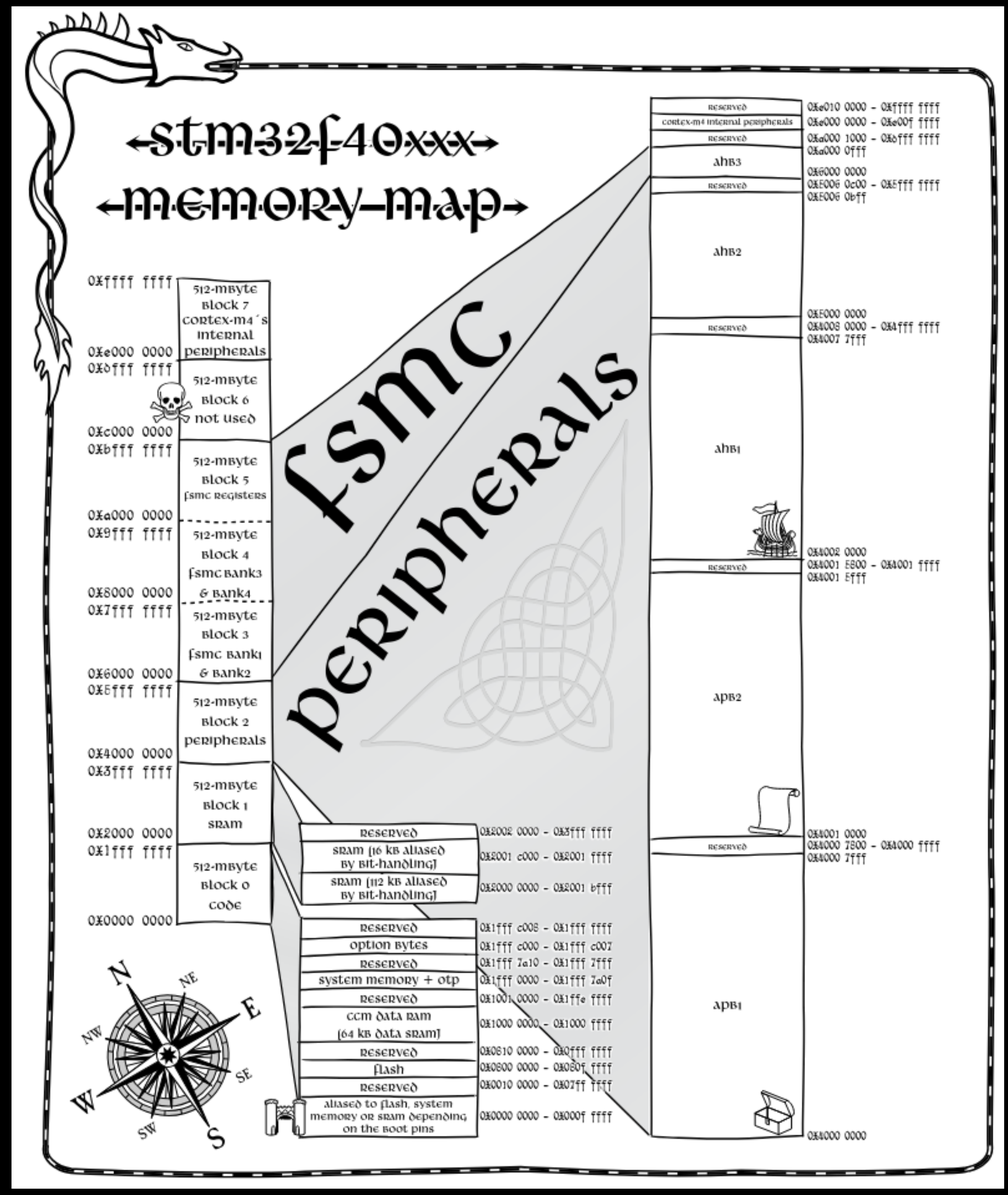


Image Credit POC||GTFO 0x11, A Tourist’s Phrasebook for Reversing Embedded ARM

How do we figure it out?

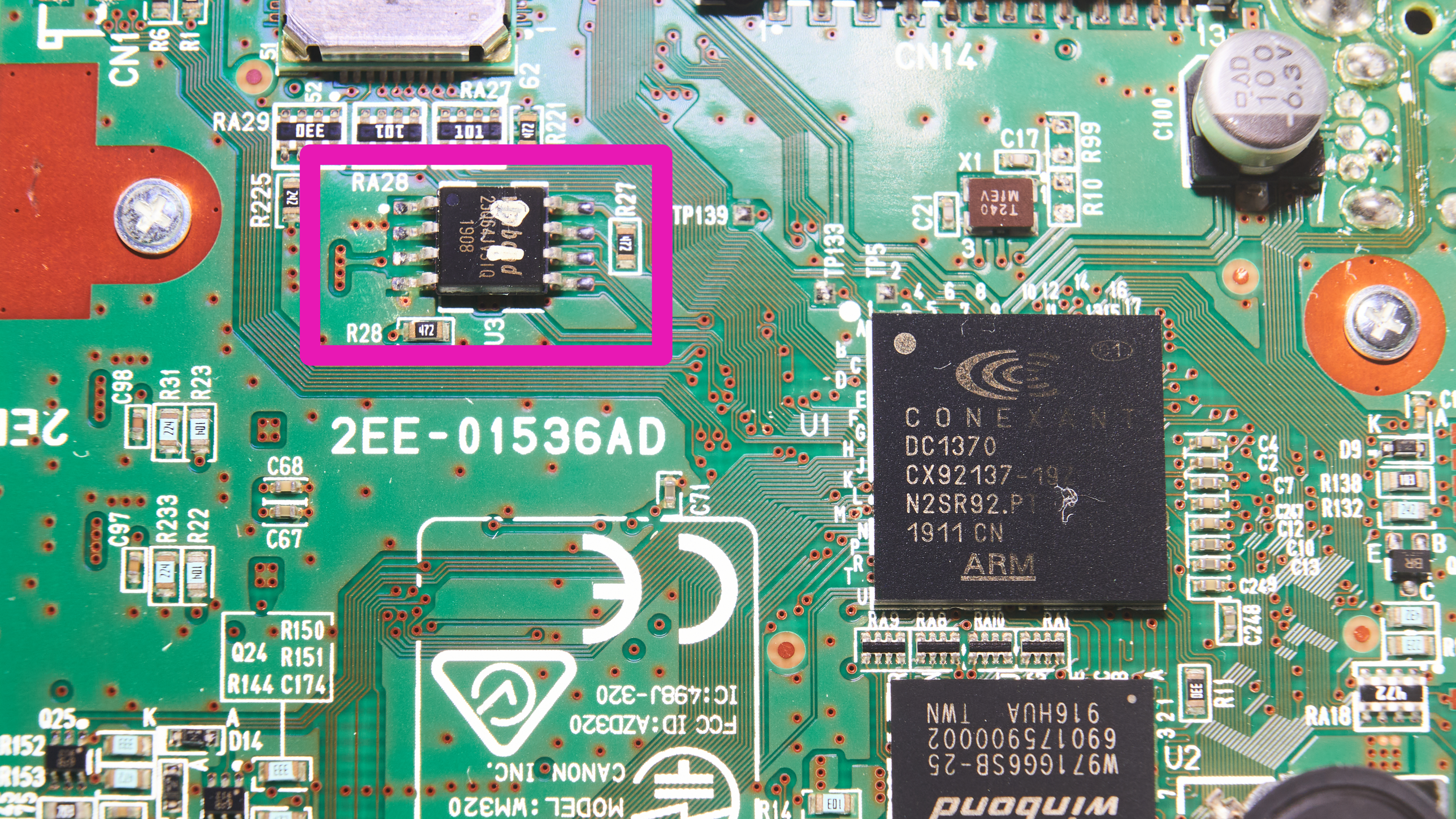
Firmware update might have a hint!

Firmware update parsing

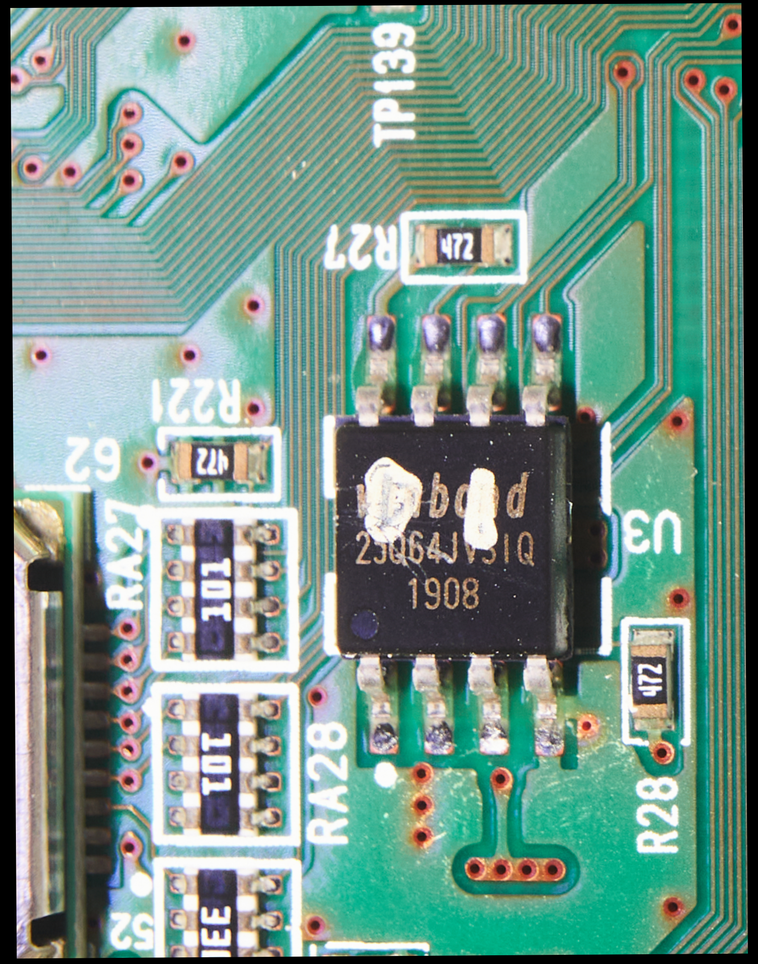
- Find the code
- Figure out the structure
- Match in hex editor
- Prototype parser
- Extract useful info

Lets look at the better mapping

Let’s talk about firmware dumping.


RA28
R28
472
U3
R27
TP139
TP133
TP5
U1
CONEXANT
DC1370
CX92137-19
N2SR92.P1
1911 CN
ARM
2EE-01536AD
CN14
CN1
RA27
RA29
R225
R221
C17
X1
C21
R10
R99
C100
C98
R31
R23
C97
R233
R22
C68
C67
C71
R150
Q24
R151
R144
C174
Q25
R152
R153
D14
IC:4981-320
FCC ID:AZD320
CANON INC.
MODEL:WM320
W9716G6SB-25
916HUA TWN
winbond
R14
C4
C7
C12
C10
C13
C249
C248
D9
R138
R132
RA18
R11

SPI Flash



W25Q64JV

winbond

3. PACKAGE TYPES AND PIN CONFIGURATIONS

3.1 Pin Configuration SOIC 150/208-mil

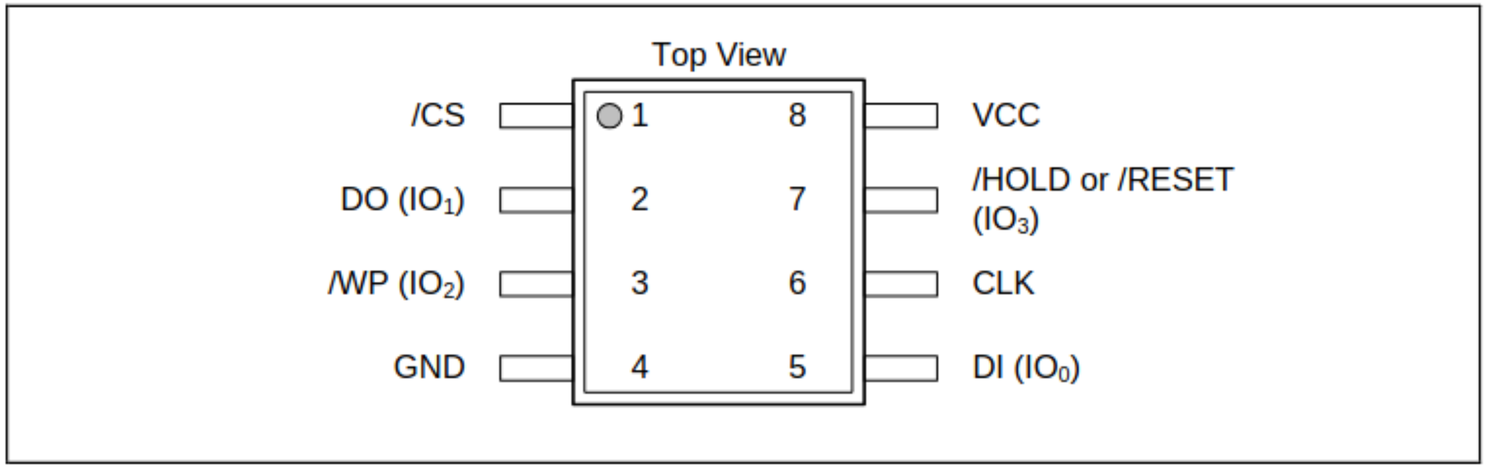
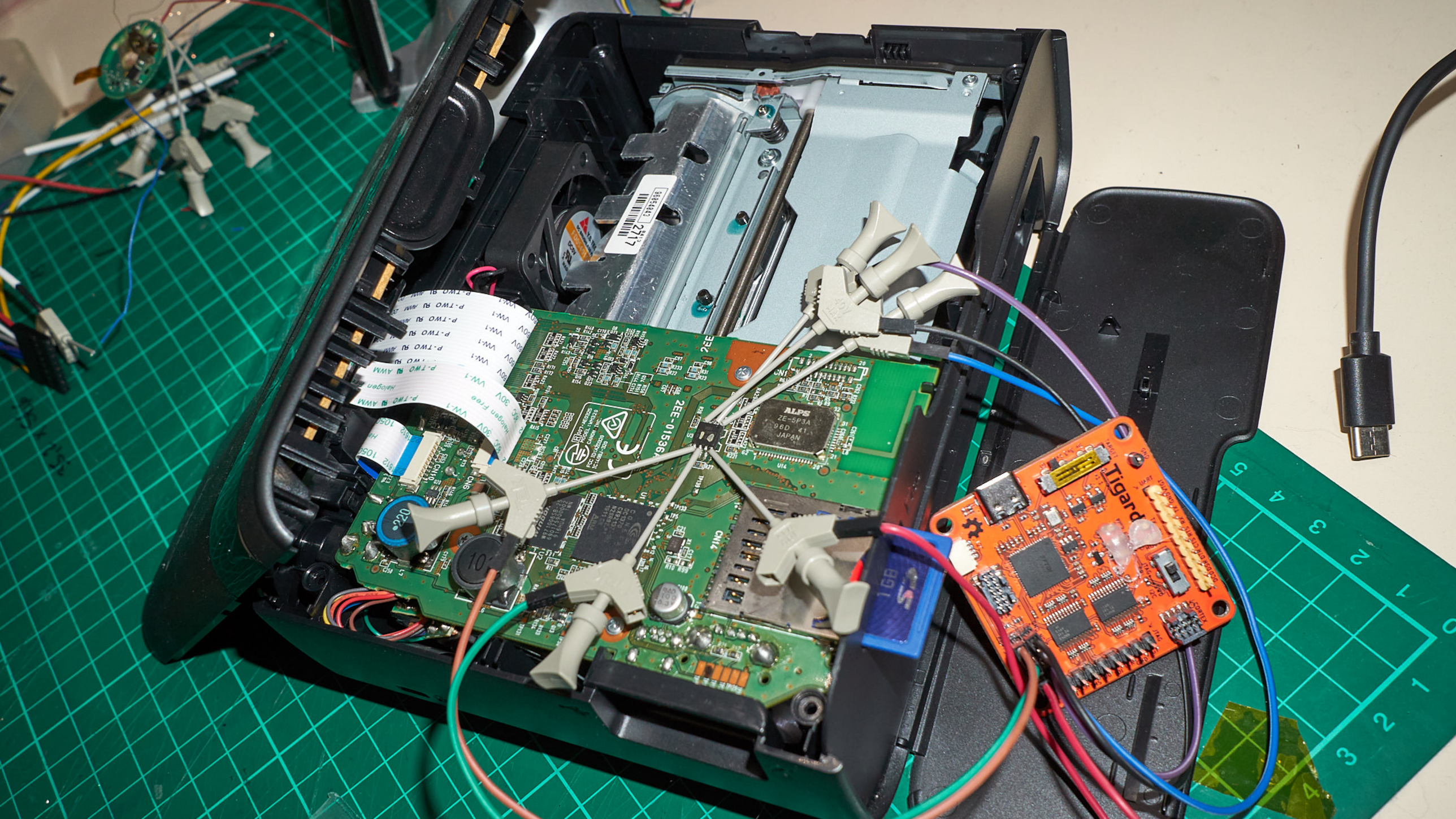
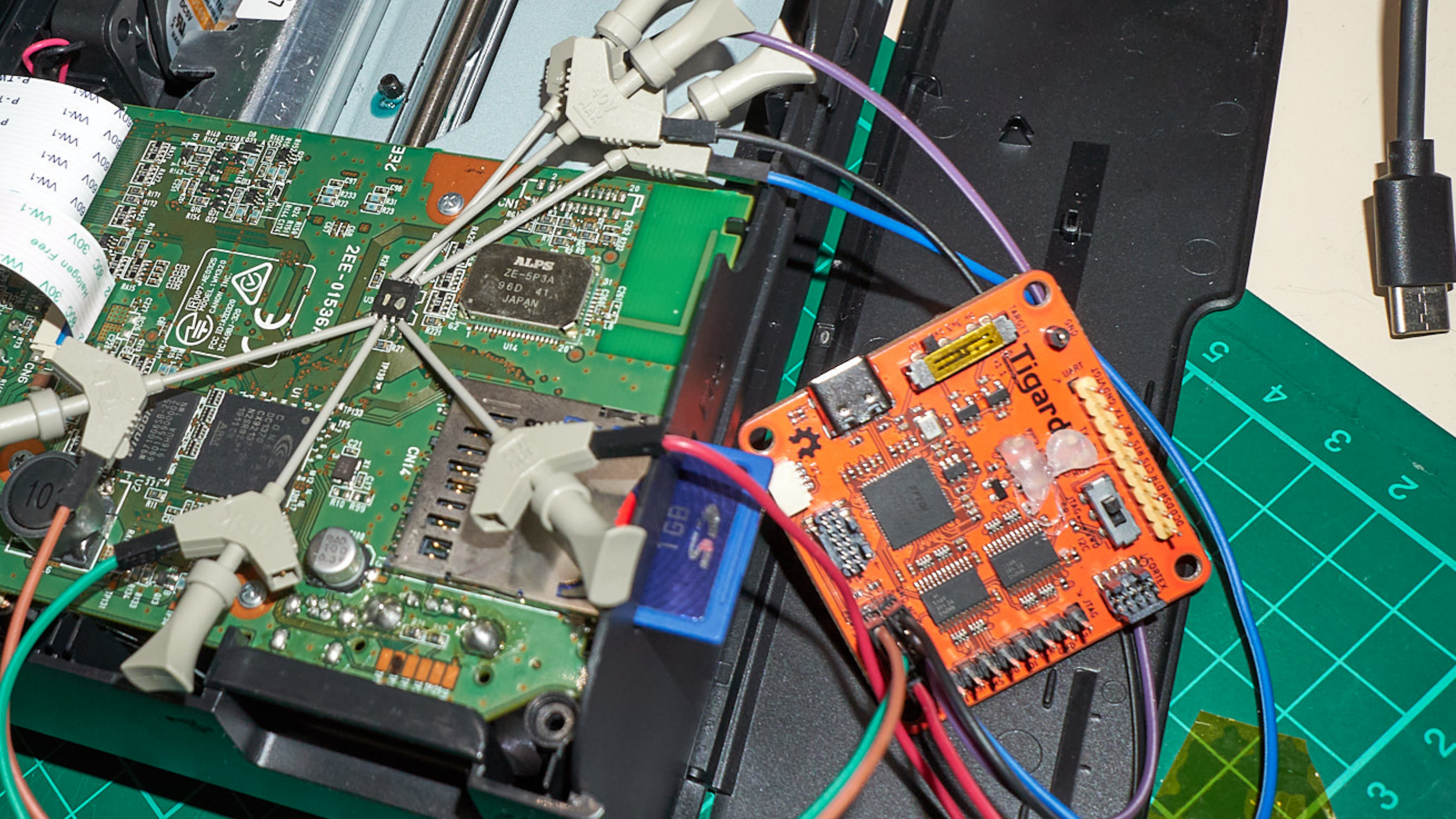


Figure 1a. W25Q64JV Pin Assignments, 8-pin SOIC 208-mil (Package Code SN,SS)

Reading SPI Flash

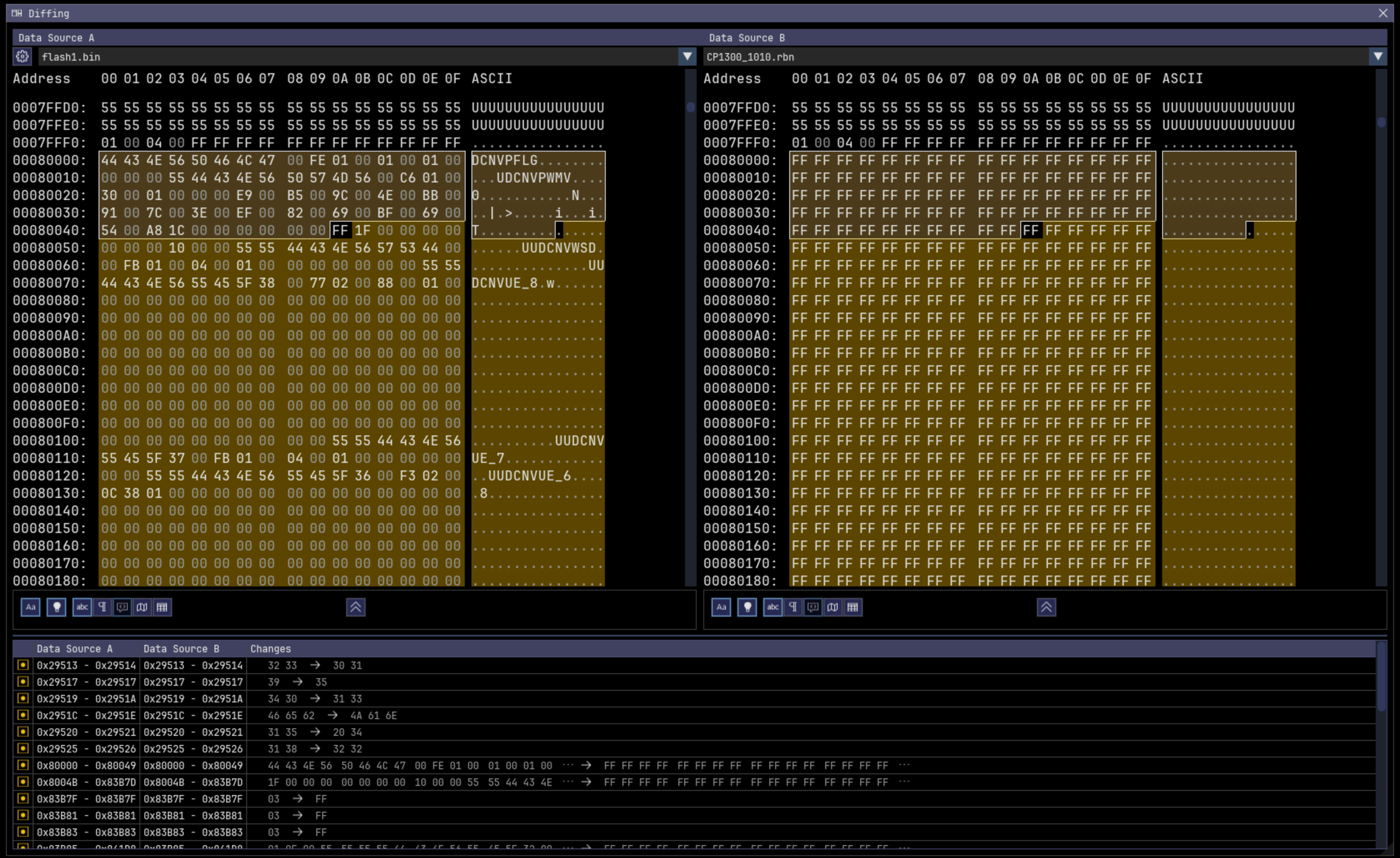
- SPI – serial peripheral interface
  - Relatively slow serial communication protocol
- Flashrom – implements reading most known chips
- Reader?
  - A Pi Pico will do
    - https://github.com/stacksmashing/pico-serprog
  - Or a Tigard
    - https://www.crowdsupply.com/securinghw/tigard
  -


ALPS
ZE-5P3A
96D 41
JAPAN
Tigard


ALPS
ZE-5P3A
96D 41
JAPAN
Tigard
1GB

Flash dump

Let’s explore the diff.


Diffing
Data Source A
flash1.bin
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
0007FFD0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
0007FFE0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
0007FFF0: 01 00 04 00 FF FF FF FF FF FF FF FF FF FF FF FF ....
00080000: 44 43 4E 56 50 46 4C 47 00 FE 01 00 01 00 01 00 DCNVPFLG........
00080010: 00 00 00 55 44 43 4E 56 50 57 4D 56 00 C6 01 00 ...UDCNVPWMV....
00080020: 30 00 01 00 00 00 E9 00 B5 00 9C 00 4E 00 BB 00 0...........N...
00080030: 91 00 7C 00 3E 00 EF 00 82 00 69 00 BF 00 69 00 ..|.>.....i...i.
00080040: 54 00 A8 1C 00 00 00 00 00 00 FF 1F 00 00 00 00 T...............
00080050: 00 00 00 10 00 00 55 55 44 43 4E 56 57 53 44 00 ......UUDCNVWSD.
00080060: 00 FB 01 00 04 00 01 00 00 00 00 00 00 00 55 55 ....UU
00080070: 44 43 4E 56 55 45 5F 38 00 77 02 00 88 00 01 00 DCNVUE_8.w......
00080080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00080090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000800A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000800B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000800C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000800D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000800E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000800F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00080100: 00 00 00 00 00 00 00 00 00 00 55 55 44 43 4E 56 ....UUDCNV
00080110: 55 45 5F 37 00 FB 01 00 04 00 01 00 00 00 00 00 UE_7............
00080120: 00 00 55 55 44 43 4E 56 55 45 5F 36 00 F3 02 00 ..UUDCNVUE_6....
00080130: 0C 38 01 00 00 00 00 00 00 00 00 00 00 00 00 00 .8..............
00080140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00080150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00080160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00080170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00080180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Data Source B
CP1300_1010.rbn
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
0007FFD0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
0007FFE0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
0007FFF0: 01 00 04 00 FF FF FF FF FF FF FF FF FF FF FF FF ....
00080000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
000800A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
000800B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
000800C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
000800D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
000800E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
000800F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080100: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080110: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080120: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080130: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080140: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080150: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080160: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080170: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
00080180: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ....
Data Source A
Data Source B
Changes
0x29513 - 0x29514 0x29513 - 0x29514 32 33 → 30 31
0x29517 - 0x29517 0x29517 - 0x29517 39 → 35
0x29519 - 0x2951A 0x29519 - 0x2951A 34 30 → 31 33
0x2951C - 0x2951E 0x2951C - 0x2951E 46 65 62 → 4A 61 6E
0x29520 - 0x29521 0x29520 - 0x29521 31 35 → 20 34
0x29525 - 0x29526 0x29525 - 0x29526 31 38 → 32 32
0x80000 - 0x80049 0x80000 - 0x80049 44 43 4E 56 50 46 4C 47 00 FE 01 00 01 00 01 00 ··· → FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ···
0x8004B - 0x83B7D 0x8004B - 0x83B7D 1F 00 00 00 00 00 00 00 10 00 00 55 55 44 43 4E ··· → FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ···
0x83B7F - 0x83B7F 0x83B7F - 0x83B7F 03 → FF
0x83B81 - 0x83B81 0x83B81 - 0x83B81 03 → FF
0x83B83 - 0x83B83 0x83B83 - 0x83B83 03 → FF

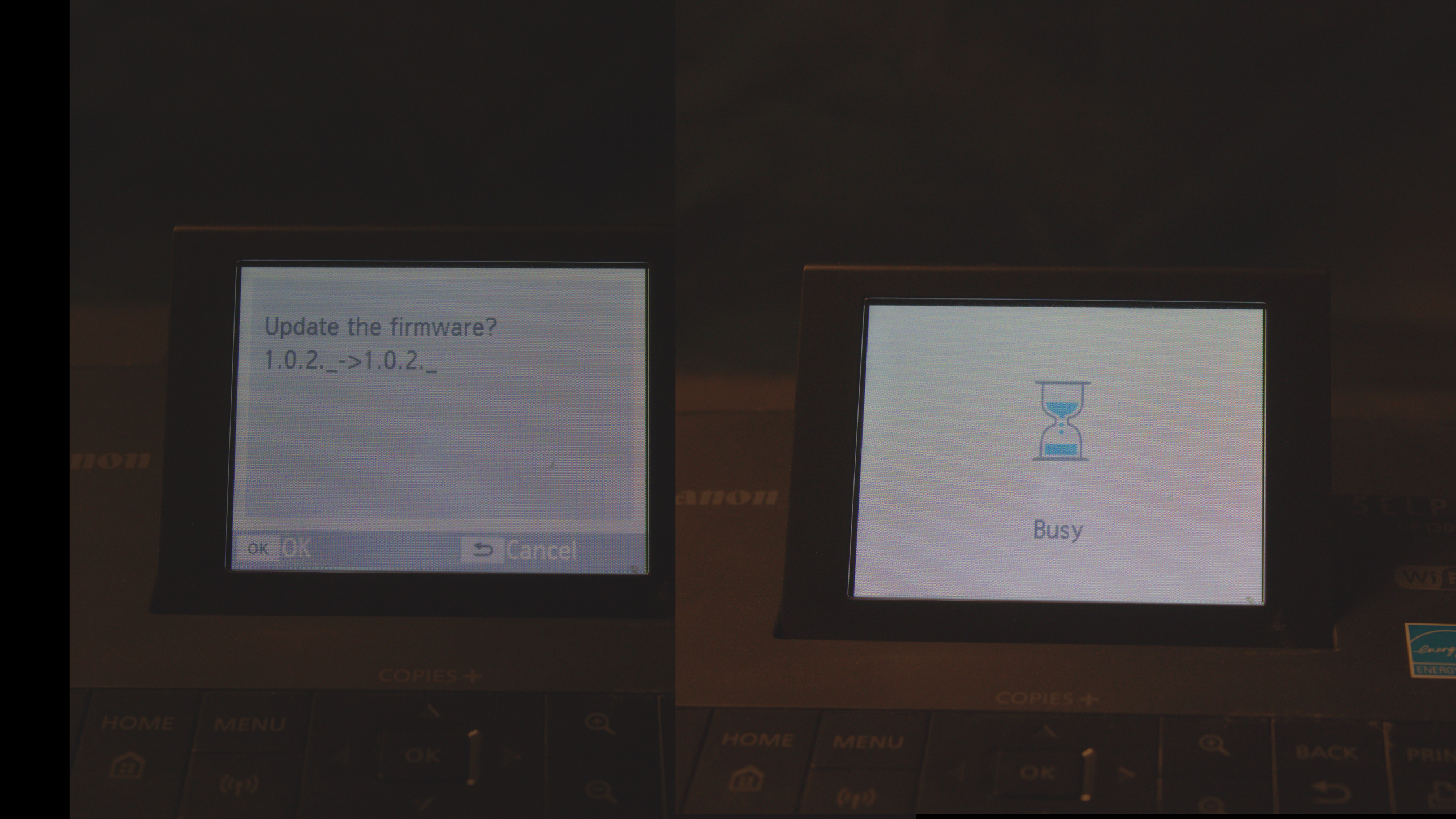
Firmware update

There’s always a checksum.

```
  uVar1 = thunk_FUN_06f6e614(param_1,0x7f4e18,uVar4);
  if (uVar1 != uVar4) {
    FUN_06ed5040();
    fmt = (char *)0x38ad0;
    goto LAB_06ed1bb4;
  }
  for (uVar1 = 0; uVar1 < uVar4; uVar1 = uVar1 + 4) {
    uVar2 = uVar2 ^ *(uint *)(uVar1 + 0x7f4e18);
  }
  uVar3 = uVar3 + uVar4;
} while( true );
```

```
romtab:
        magic: 1414676815
        rom_version: 0x15430002
        unk1: 0xa
        num_items: 52
        fw_idx: 0x1c
        fw_memory_address: 0x6e98000
        romtab_size: 5
        rom_size: 0x567d8c
        unk5: 0x1fc036ae
        unk6: 0x32db
        unk7: 0x80000000
```

```python
def get_checksum(filename, checksum_offset,romtab_offset):
    with open(filename, 'rb') as f:
        f.seek(checksum_offset)
        byte_block = f.read(4)
        orig_checksum = int.from_bytes(byte_block, 'little')
        f.seek(romtab_offset)
        result = 0
        for i in range(0, os.path.getsize(f.name), 4):
            byte_block = f.read(4)
            dword = int.from_bytes(byte_block, 'little')
            result ^= dword
        if result != 0:
            print("Correct cheksum should be: %x "%(result^orig_checksum))
        return result^orig_checksum
```


Update the firmware?
1.0.2._->1.0.2._
OK OK
Cancel
Busy
COPIES +
HOME
MENU
OK
BACK


Canon
Canon
ELPHY
CP1300
WiFi
ENERGY STAR
COPIES +
ON
HOME
MENU
OK
BACK
PRINT
not Simon
tones
Launch !
TP90

So that’s it?

- There’s one more tricky bit:

```
$ file FIRMWARE
FIRMWARE: gzip compressed data, last modified: Tue Jan  4 10:58:18 2022, from NTFS filesystem
$ ls -lh FIRMWARE
-rw-rw-r-- 1 ea ea 3.9M Sep  7 20:15 FIRMWARE
$ cp FIRMWARE FIRMWARE_unpack.gz
$ gunzip FIRMWARE_unpack.gz
$ ls -lh FIRMWARE_unpack
-rw-rw-r-- 1 ea ea 7.5M Sep  7 20:16 FIRMWARE_unpack
$ gzip FIRMWARE_unpack
$ ls -lh FIRMWARE_unpack.gz
-rw-rw-r-- 1 ea ea 3.5M Sep  7 20:16 FIRMWARE_unpack.gz
```

```
      FUN_00002420(fw_update,fw_update->field135_0x9c & 7);
      uVar3 = get_bits?(fw_update,0x10);
      *(undefined2 *)&fw_update->field_0xa8 = uVar3;
      FUN_000023cc(fw_update,0x10);
      FUN_00002420(fw_update,0x10);
      uVar2 = 1;
    }
LAB_000020cc:
    fw_update->field136_0xa0 = uVar2;
  }
  else {
    if (cVar1 != '\x01') {
      if (cVar1 != '\x02') {
        printf("INFLATE() error");
        do {
                    /* WARNING: Do nothing block with infinite loop */
        } while( true );
      }
      FUN_00002478(fw_update);
      uVar2 = 0;
      goto LAB_000020cc;
    }
    uVar4 = (uint)*(ushort *)&fw_update->field_0xa8;
    fw_update->field136_0xa0 = uVar4 != 0;
    if ((int)*(uint *)&fw_update->expected_size < (int)uVar4) {
      uVar4 = *(uint *)&fw_update->expected_size;
    }
    btype = fw_update->field3_0xc;
    uVar5 = uVar4 & 0xffff;
    fw_update->field3_0xc = btype + uVar5;
    *(uint *)&fw_update->expected_size = *(int *)&fw_update->expected_size - uVar5;
    *(short *)&fw_update->field_0xa8 = *(short *)&fw_update->field_0xa8 - (short)uVar4;
    pcVar6 = fw_update->load_address + btype;
    while (bVar7 = uVar5 != 0, uVar5 = uVar5 - 1 & 0xffff, bVar7) {
      cVar1 = FUN_00001fd0(fw_update->archive_data);
      *pcVar6 = cVar1;
      fw_update->archive_data = fw_update->archive_data + 1;
      pcVar6 = pcVar6 + 1;
    }
```


Canon
Canon
ELPHY
CP1300
WiFi
ENERGY STAR
COPIES +
ON
HOME
MENU
OK
BACK
PRINT
not Simon
tones
Launch 1
TP50

DEFLATE algorithm

- First bit: Last-block-in-stream marker:
  - `1`: This is the last block in the stream.
  - `0`: There are more blocks to process after this one.
- Second and third bits: Encoding method used for this block type:
  - `00`: A stored (a.k.a. raw or literal) section, between 0 and 65,535 bytes in length
  - `01`: A *static Huffman* compressed block, using a pre-agreed Huffman tree defined in the RFC
  - `10`: A *dynamic Huffman* compressed block, complete with the Huffman table supplied
  - `11`: Reserved—don't use.

Infgen

from Mark Adler

- https://github.com/madler/infgen

```
 $ ./infgen -h

infgen 3.5
Usage:

  infgen [-d[d]misc[c]q[q]rb[b]] input_path > output_path
  infgen [-d[d]misc[c]q[q]rb[b]] < input_path > output_path

    -d   Write raw dynamic header (code lengths in comments)
    -dd  Also show the bits for each element displayed
    -m   Show copied data after each match
    -i   Include detailed gzip / zlib header descriptions
    -s   Include deflate block statistics (as comments)
    -c   Color the output components (if terminal supports)
    -cc  Use high-intensity instead of standard colors
    -q   Do not write dynamic code lengths (comments or not)
    -qq  Do not write deflate stream description at all
    -r   Assume raw deflate data -- do not look for headers
    -b   Write compact binary format (only -r honored)
    -bb  Write compact binary format with bit counts
```

Original

```
 $  ./infgen -bb < ../flash_dump/FIRMWARE| ./infstats
1 stream
31 stored blocks: 557,307(4) -> 557,153 (1.000)
  data: μ = 17,972.7, σ = 1,027.9, in [16,383..19,160]
136 fixed blocks: 3,470,468(6) -> 7,234,335 (0.480)
  lits: 1,491,990(2) -> 1,432,053 (1.042)
  mats: 1,978,308(4) -> 5,802,282 (0.341)
  bits: μ = 204,145.2, σ = 25,244.1, in [42,869..241,189]
  syms: μ = 16,286.6, σ = 1,124.1, in [3,274..16,383]
  lens: μ = 7.4, σ = 16.8, in [3..258]
  data: μ = 53,193.6, σ = 27,264.0, in [19,211..233,466]
```

Modified

```
 $  ./infgen -bb < ../flash_dump/mod/FIRMWARE_test.gz| ./infstats
1 stream
167 dynamic blocks: 3,612,441(5) -> 7,791,488 (0.464)
  lits: 1,871,128(7) -> 1,919,958 (0.975)
  mats: 1,723,913(3) -> 5,871,530 (0.294)
  bits: μ = 173,051.1, σ = 32,053.9, in [42,876..213,484]
  head: μ = 820.3, σ = 210.9, in [249..1,133]
  syms: μ = 16,306.4, σ = 989.4, in [3,597..16,383]
  lens: μ = 7.3, σ = 16.5, in [3..258]
  data: μ = 46,655.6, σ = 28,197.2, in [16,383..236,264]
```

So we can modify firmware

Telnet?

How to draw an owl

1.

2.

1. Draw some circles

2. Draw the rest of the fucking owl

```
 $ nc -c 192.168.1.207 59223
**** Welcome to SELPHY TELNET Server ****

login password=alps-alsi
SELPHY_Command>iperf --help
Usage: iperf  [-s | -c host] [options]
       iperf  --help
Client / Server:
  -p <port>  server port to listen on/connect to
  -u         use UDP
```

Contact

- https://bsky.app/profile/fuzzyaleks.bsky.social
- https://infosec.exchange/@FuzzyAleks
- https://github.com/ea/
- https://blog.29b.net/
-