NAHUEL RIVA
Embedded & Hardware Team / ncriva@quarkslab.com

Old New Things: An examination of the Philips TriMedia architecture

quarkslab
SECURING EVERY BIT OF YOUR DATA

WARNING!

PARENTAL ADVISORY

ADULT LANGUAGE

SENSITIVE SUBJECT MATTER

Agenda

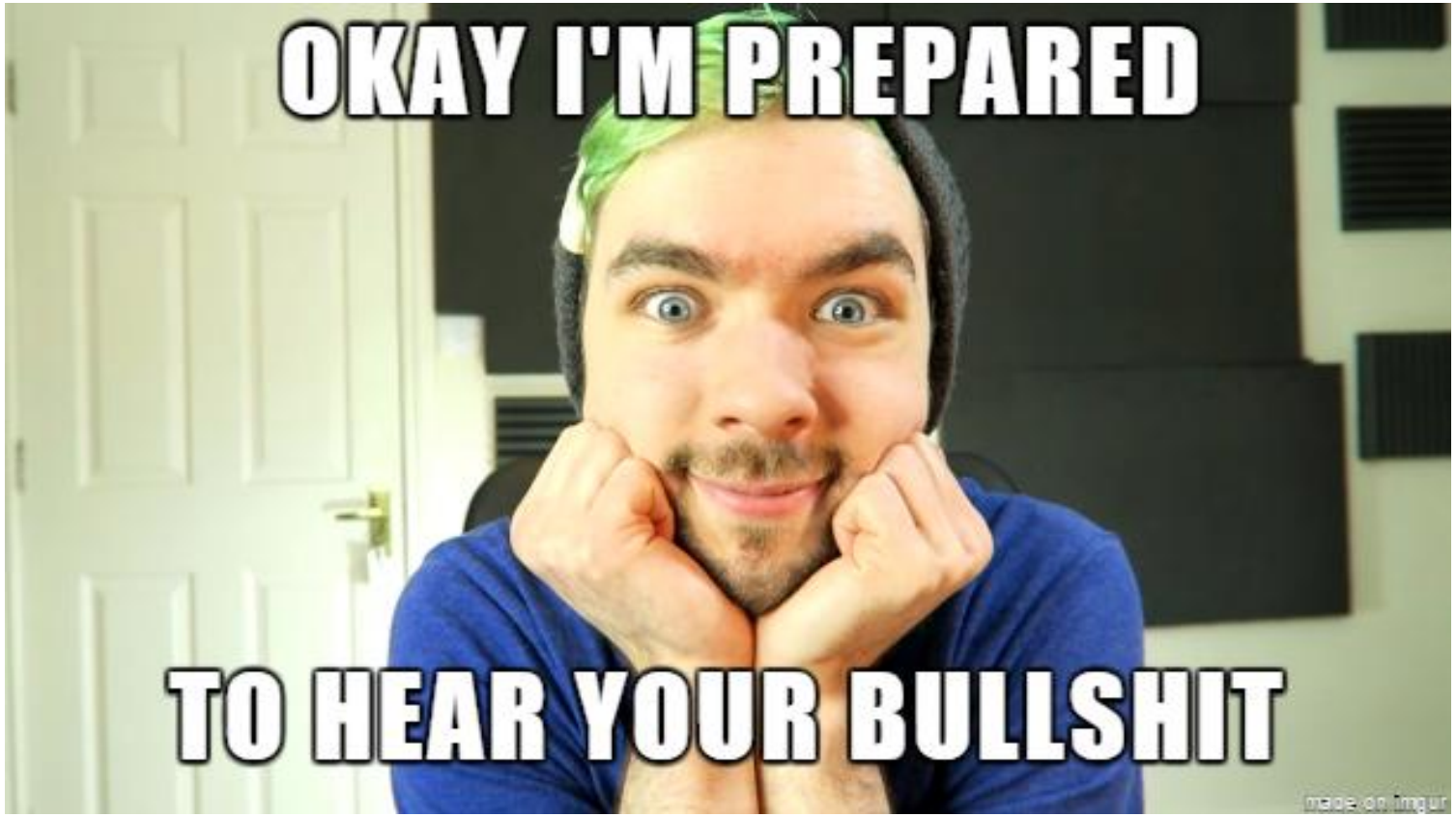
- Introduction
  - Presentation
  - Tell the story …
- Previous work
- Analysis of the device
  - Basic firmware analysis
  - Exploring the attack surface
  - Hardware exploration
- The Philips TriMedia CPU
  - History
  - Sources of information
    - Tools and hardware
  - The PNX1300EH CPU
  - ASM and instruction set
  - Compression scheme
  - Disassembling the code
- TODO
- Conclusion

Introduction

- Nahuel Riva from General Pico, La Pampa, Argentina
- Currently working on the Embedded Hardware Team @ Quarkslab
- Previously @ Core Security doing Exploit Development for 10 years
- Before Core, I used to break software protections
- I like infosec, reverse engineering, ASM, dogs, Asado, beer, tattoos, bodybuilding from the 70's
- A lot of other things …

Introduction

Tell the story …



Previous work

- There’s not much work done about TriMedia (publicy available, of course)
- The only resource I found was at https://hackingbtbusinesshub.wordpress.com/ (not available anymore since 2015)
  - Valuable information about 2Wire routers hacking
- https://blog.quarkslab.com/reverse-engineering-a-philips-trimedia-cpu-based-ip-camera-part-1.html
- https://blog.quarkslab.com/reverse-engineering-a-philips-trimedia-cpu-based-ip-camera-part-2.html

Analysis of the device

Quarkslab

SECURING EVERY BIT OF YOUR DATA

The device

- D-Link DCS-5300 IP camera
  - PAN/TILT zoom (360)
  - Microphone
  - Motion detection
  - Remote control
  - Ethernet, WiFi
  - Audio and video support
  - Remote administration
  - UPnP, DDNS, FTP, Telnet
  - Etc

Basic firmware analysis

```
fastix@bulin:~/dcs-5300$ binwalk dcs5300_firmware_105.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
978627        0xEEEC3         HTML document header
979192        0xEF0F8         HTML document footer
979238        0xEF126         HTML document header
995907        0xF3243         HTML document header
996327        0xF33E7         HTML document footer
997041        0xF36B1         HTML document header
997325        0xF37CD         HTML document footer
998169        0xF3B19         HTML document header
1000439       0xF43F7         HTML document footer
1000448       0xF4400         HTML document header
```

```
1060247       0x102D97        HTML document footer
1060755       0x102F93        XML document, version: "1.0"
1096400       0x10BAD0        Unix path: /usr/local/etc/zoneinfo
1103113       0x10D509        Base64 standard index table
1103407       0x10D62F        HTML document header
1103549       0x10D6BD        HTML document footer
1103557       0x10D6C5        HTML document header
1103691       0x10D74B        HTML document footer
1103699       0x10D753        HTML document header
1103857       0x10D7F1        HTML document footer
1106328       0x10E198        Microsoft Cabinet archive data, 278748 bytes, 1 file
1385220       0x152304        Certificate in DER format (x509 v3), header length: 4, sequence length: 964
1386188       0x1526CC        Certificate in DER format (x509 v3), header length: 4, sequence length: 1023
1387215       0x152ACF        Certificate in DER format (x509 v3), header length: 4, sequence length: 1215
1388434       0x152F92        Certificate in DER format (x509 v3), header length: 4, sequence length: 1269
1388593       0x153031        Digi International firmware, load address: 0x204D6963, entry point: 0x66742053,
1390788       0x1538C4        JPEG image data, JFIF standard 1.02
1392148       0x153E14        GIF image data, version "89a", 1 x 1
1392191       0x153E3F        GIF image data, version "89a", 1022 x 124
1424806       0x15BDA6        GIF image data, version "89a", 584 x 8
1426129       0x15C2D1        GIF image data, version "89a", 96 x 96
```

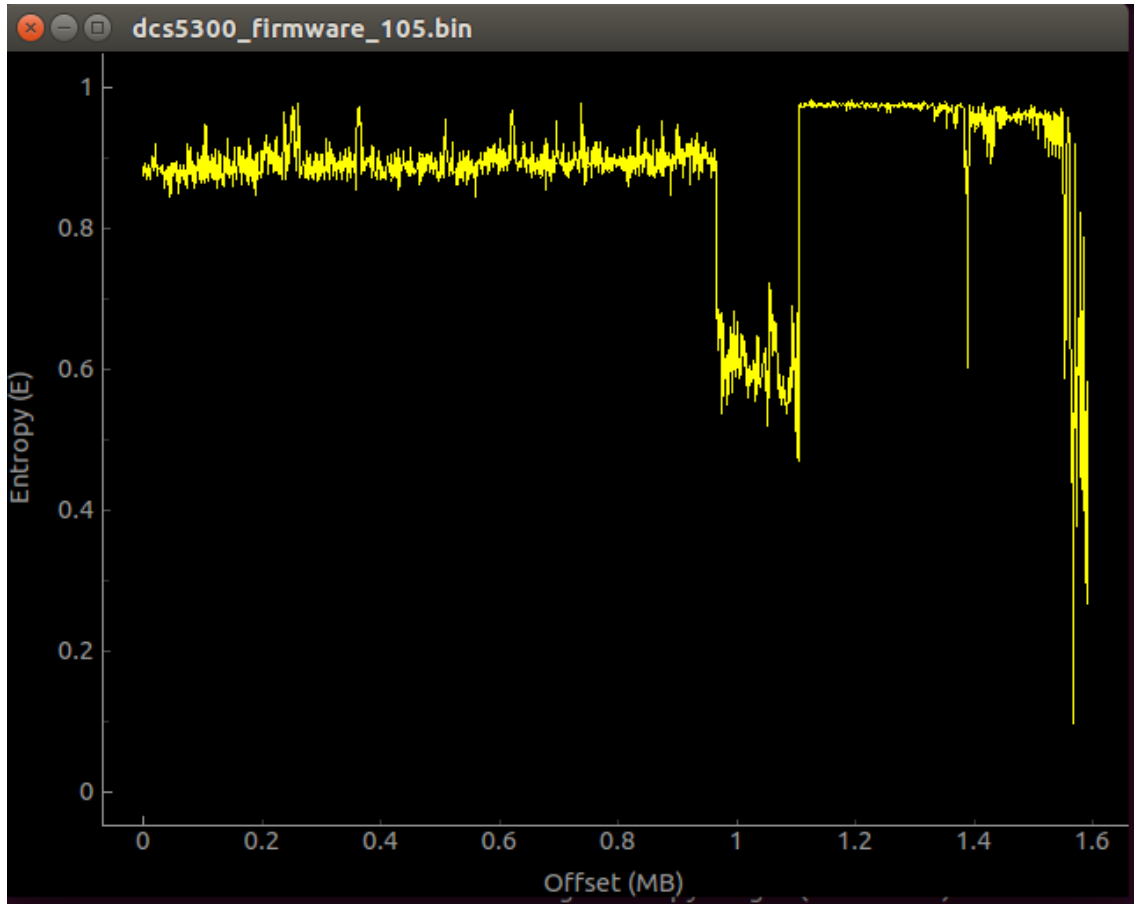
Basic firmware analysis

- Firmware version: 1.05 (the one installed on the camera)
  - Latest available version is 1.06
- Binwalk results:
  - binwalk was capable of recognizing some files starting at offset **0xEEEC3**
    - What about the first MB of data?
  - Lots of HTML, XML, GIF, JPEG files (for the Web admin interface)
  - A CAB file with an ActiveX (useful for IE)
  - Some certs
  - "Digi International firmware" false positive
  - No bootloader or kernel image?
    - Generally, they are located at the beginning of the firmware image, not in this case

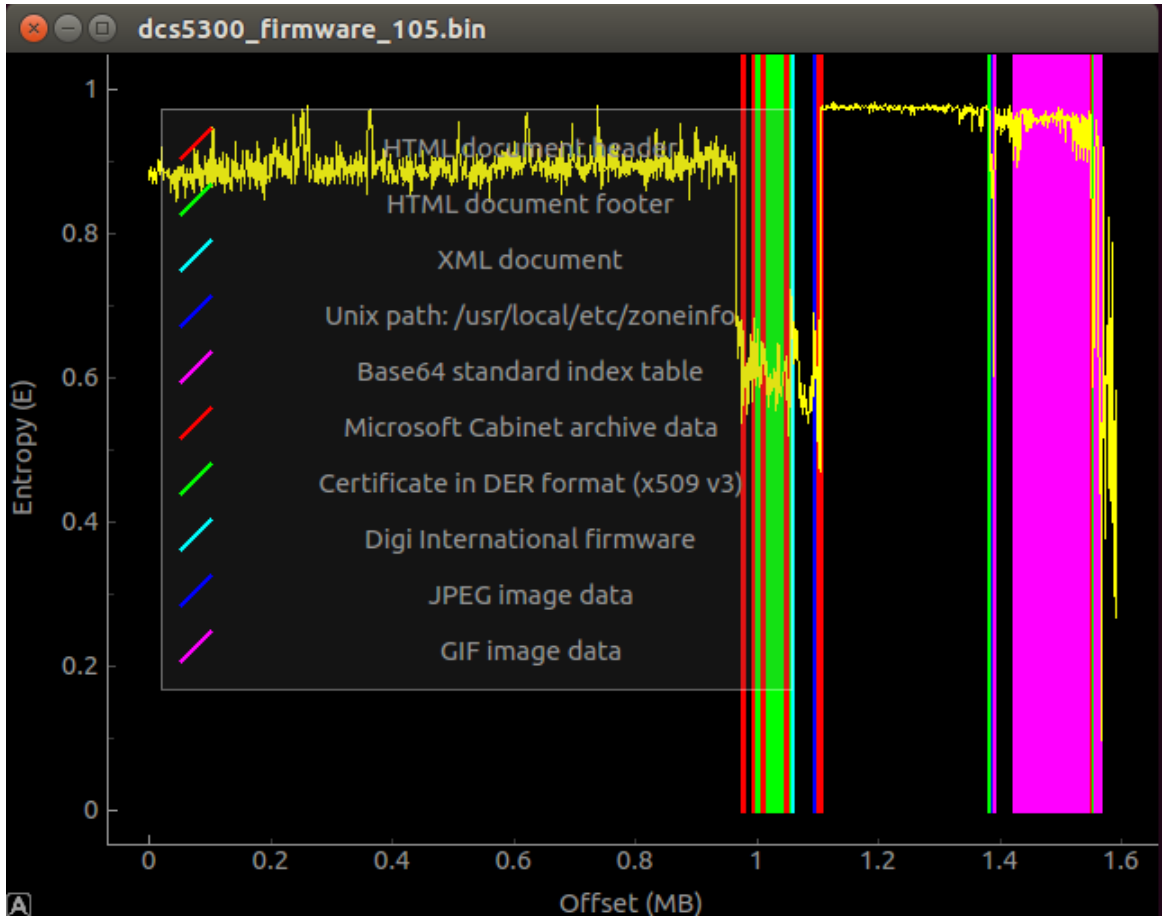
Basic firmware analysis

Basic firmware analysis

- Entropy analysis only



- Entropy analysis with signatures



Basic firmware analysis

- First MB of data has high entropy
  - Can be compressed or encrypted (we'll see later)
  - Couldn't identify any common compression algorithm nor any cipher
  - Definitely, not a « common » firmware structure

Exploring the attack surface

nmap -sV 192.168.1.0/24

Running services:

- http (80 tcp)
- ftp (21 tcp)
- telnet (23 tcp)

- commplex-link (5001 tcp)
- rfe (5002 tcp)
- filemaker (5003 tcp)

```
Nmap scan report for 192.168.1.114
Host is up (0.0010s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE        VERSION
21/tcp   open  ftp
23/tcp   open  telnet
80/tcp   open  http           D-Link Internet Camera
5001/tcp open  commplex-link?
5002/tcp open  rfe?
5003/tcp open  filemaker?
3 services unrecognized despite returning data. If you know the service/ver
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port21-TCP:V=7.60%I=7%D=7/24%Time=5B575F77%P=x86_64-pc-linux-gnu%r(NULL
SF:,20,"220\x20DCS-5300\x20FTP\x20server\x20ready\.\r\n")%r(GenericLines,3
SF:4,"220\x20DCS-5300\x20FTP\x20server\x20ready\.\r\n530\x20access\x20deni
SF:ed\.\r\n")%r(Help,34,"220\x20DCS-5300\x20FTP\x20server\x20ready\.\r\n53
SF:0\x20access\x20denied\.\r\n")%r(SMBProgNeg,34,"220\x20DCS-5300\x20FTP\x
SF:20server\x20ready\.\r\n530\x20access\x20denied\.\r\n");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port23-TCP:V=7.60%I=7%D=7/24%Time=5B575F77%P=x86_64-pc-linux-gnu%r(NULL
SF:,26,"\xff\xfd\x18DCS-5300\x20Telnet\x20Daemon\r\nPassword\x20:\x20")%r(
SF:GenericLines,26,"\xff\xfd\x18DCS-5300\x20Telnet\x20Daemon\r\nPassword\x
SF:20:\x20")%r(tn3270,26,"\xff\xfd\x18DCS-5300\x20Telnet\x20Daemon\r\nPass
SF:word\x20:\x20")%r(GetRequest,31,"\xff\xfd\x18DCS-5300\x20Telnet\x20Daem
SF:on\r\nPassword\x20:\x20Password\x20:\x20")%r(RPCCheck,26,"\xff\xfd\x18D
SF:CS-5300\x20Telnet\x20Daemon\r\nPassword\x20:\x20")%r(Help,31,"\xff\xfd\
SF:x18DCS-5300\x20Telnet\x20Daemon\r\nPassword\x20:\x20Password\x20:\x20")
SF:%r(SIPOptions,94,"\xff\xfd\x18DCS-5300\x20Telnet\x20Daemon\r\nPassword\
SF:x20:\x20Password\x20:\x20Password\x20:\x20Password\x20:\x20Password\x20
SF::\x20Password\x20:\x20Password\x20:\x20Password\x20:\x20Password\x20:\x
SF:20Password\x20:\x20Password\x20:\x20")%r(NCP,26,"\xff\xfd\x18DCS-5300\x
SF:20Telnet\x20Daemon\r\nPassword\x20:\x20");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port80-TCP:V=7.60%I=7%D=7/24%Time=5B575F77%P=x86_64-pc-linux-gnu%r(GetR
SF:equest,10D,"HTTP/1\.1\x20401\x20Unauthorized\r\nWWW-Authenticate:\x20Ba
SF:sic\x20realm=\"DCS-5300\"\r\nContent-Type:\x20text/html\r\nServer:\x20D
SF:-Link\x20Internet\x20Camera\r\n\r\n<HTML>\n<HEAD>\n<TITLE>Protected\x20
SF:Object</TITLE></HEAD><BODY>\n<H1>Protected\x20Object</H1>This\x20object
SF:\x20on\x20the\x20server\x20is\x20protected\.<P>\n</BODY></HTML>")%r(RTS
SF:PRequest,6E,"HTTP/1\.1\x20405\x20Method\x20Not\x20Allowed\r\nAllow:\x20
SF:GET,\x20HEAD,\x20POST\r\nContent-Length:\x200\r\nServer:\x20D-Link\x20I
SF:nternet\x20Camera\r\n\r\n")%r(SIPOptions,6E,"HTTP/1\.1\x20405\x20Method
SF:\x20Not\x20Allowed\r\nAllow:\x20GET,\x20HEAD,\x20POST\r\nContent-Length
SF::\x200\r\nServer:\x20D-Link\x20Internet\x20Camera\r\n\r\n");
```

Exploring the attack surface - Telnet

- Password is « admin »
  - Can’t be changed
- Commands:
  - Debug
  - Clear
  - Reset
  - No shell? :/

```
fastix@bulin:~/dcs-5300$ telnet 192.168.1.114
Trying 192.168.1.114...
Connected to 192.168.1.114.
Escape character is '^]'.
DCS-5300 Telnet Daemon
Password : admin
Authorized and start service
help
Supported commands :
        debug    : Dump debug information
        dinote   : Dump changed input status
        stop     : Stop dumping debug info and input status
        diquery  : Dump current input status
        do1=h    : Set output 1 to high
        do1=l    : Set output 1 to low
        erase graph : Erase all graphics
        erase homepage : Erase custom homepage
        lock     : Lock network settings
        unlock   : Reset network settings
        clear    : Restore factory settings
        reset    : Restart system
        save     : Save parameters
```

Exploring the attack surface - Telnet

- Output of « dump » command:
  - MAC address
  - IP address
  - DNS servers
  - Status of Telnet, FTP and Web servers
- Undocumented commands:
  - fanon
  - fanoff
  - newweb (creates new root folder)
  - suicide (restart the camera)

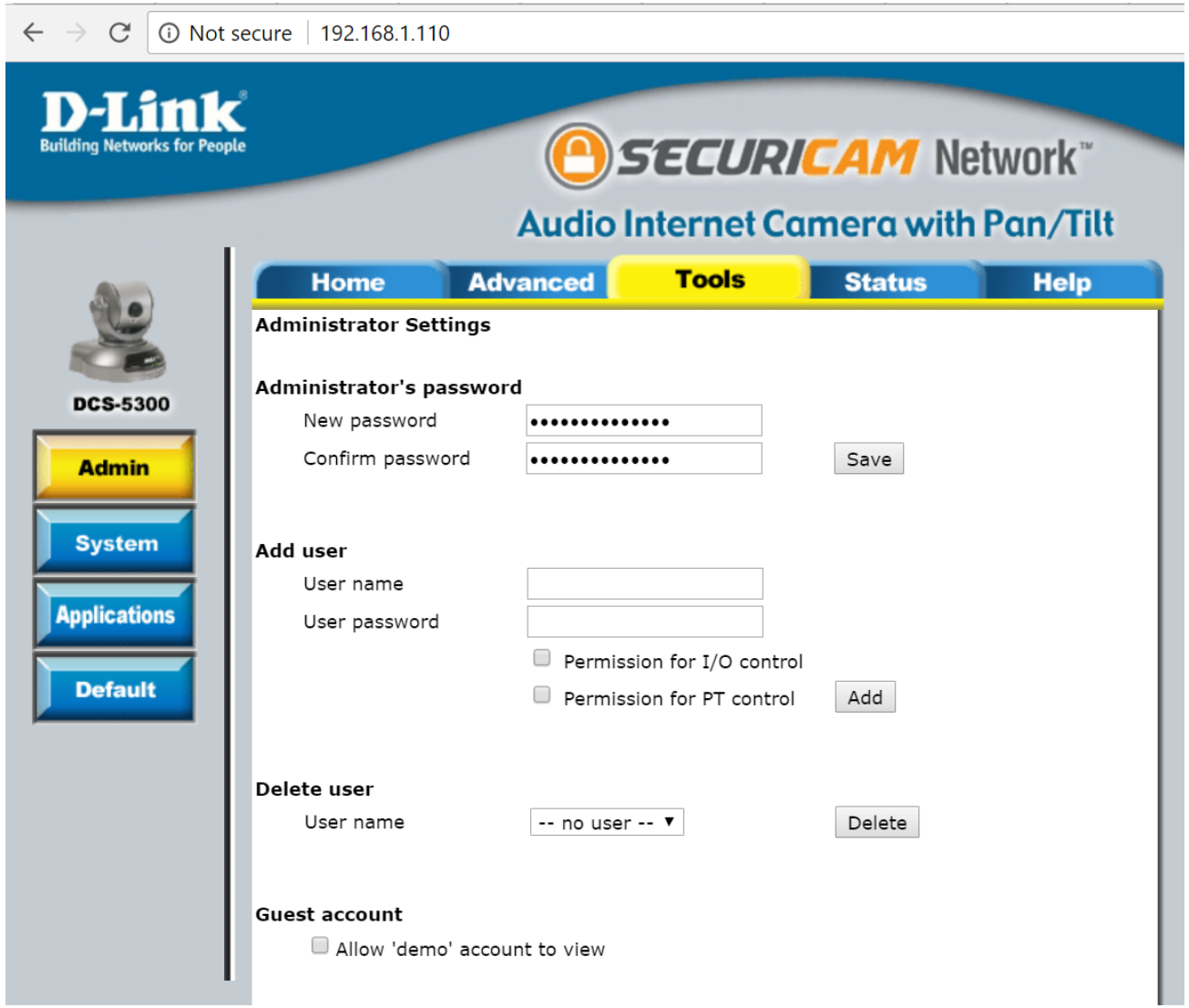
```
TLN: Start debugging
SYS: MAC address = 00-0D-88-7E-35-B9
SYS: Update FLASH!
ETH: Activate Ethernet
ETH: Ethernet link speed is 100Mbps.
SYS: Ethernet is chosen
SYS: -----NET INFO-----
SYS: Host IP=192.168.1.114
SYS: Subnet Mask=255.255.255.0
SYS: Default gateway=192.168.1.1
SYS: Primary DNS server=186.56.20.66
SYS: Secondary DNS server=186.56.20.67
SYS: Video modulation is NTSC
SYS: No logo
TLN: Server starts up
FTP: Server starts up
SYS: No background
SYS: No custom homepage
H263 Encode Task start!
Audio Encode Task start!
comm res0:0
h263_control.dwBitRate=356000
WWW: Server starts up
Reboot Timer started
Reboot after 86400 sec = 1day 00hr:00min:00sec
SYS: System starts at 2018/02/27 22:55:30 in local time
[UPNPMiniServer] Bind at port 13396
UPnP started:0
TLN: Stop debugging
 -STOP-
```

Exploring the attack surface – FTP

- Only accesible folder is the « root »
- Some interesting files:
  - flash.bin (entire filesystem), only « write » permissions
  - system.log (activiy log), only « read » permissions
  - config.ini (configuration of the camera, includes Web panel credentials)

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Gro...
..					
config.ini	2,935	Configuration settings	11/7/2017 9:11:...	-rw-rw-rw-	ftp ftp
flash.bin	1,592,337	BIN File	11/7/2017 9:01:...	--w--w--w-	ftp ftp
logo.gif	0	GIF File	11/7/2017 9:01:...	-rw-rw-rw-	ftp ftp
system.log	707	Text Document	11/7/2017 9:11:...	-r--r--r--	ftp ftp
user.htm	0	Chrome HTML Document	11/7/2017 9:01:...	-rw-rw-rw-	ftp ftp
video.jpg	0	JPG File	11/7/2017 9:11:...	-r--r--r--	ftp ftp
vpos.jpg	0	JPG File	1/1/2002	-r--r--r--	ftp ftp
vpre.jpg	0	JPG File	1/1/2002	-r--r--r--	ftp ftp
vtrg.jpg	0	JPG File	1/1/2002	-r--r--r--	ftp ftp
wallppr.jpg	62	JPG File	11/7/2017 9:01:...	-rw-rw-rw-	ftp ftp

Exploring the attack surface – HTTP



Exploring the attack surface – HTTP

- Default Administrator username is « admin » with blank pwd
- Guest account disabled by default
  - Gives access to video stream via snapshots
  - Guest account username is « demo » with blank pwd
- User-Agent reported by the Web server is « VVTK »
  - Could it be the same Web server used in Vivotek IP cameras?
  - Yes, in fact, they share (almost) the same CGI API :P
- Some of the CGIs were affected by vulns in the past (getparam.cgi, CVE-2013-1594)
  - However, these CGIs require authentication to be invoked in the Dlink camera
  - Another vuln: CSRF in /setup/security.cgi (CVE-2012-5319), requires authentication

Exploring the attack surface – Mysterious services

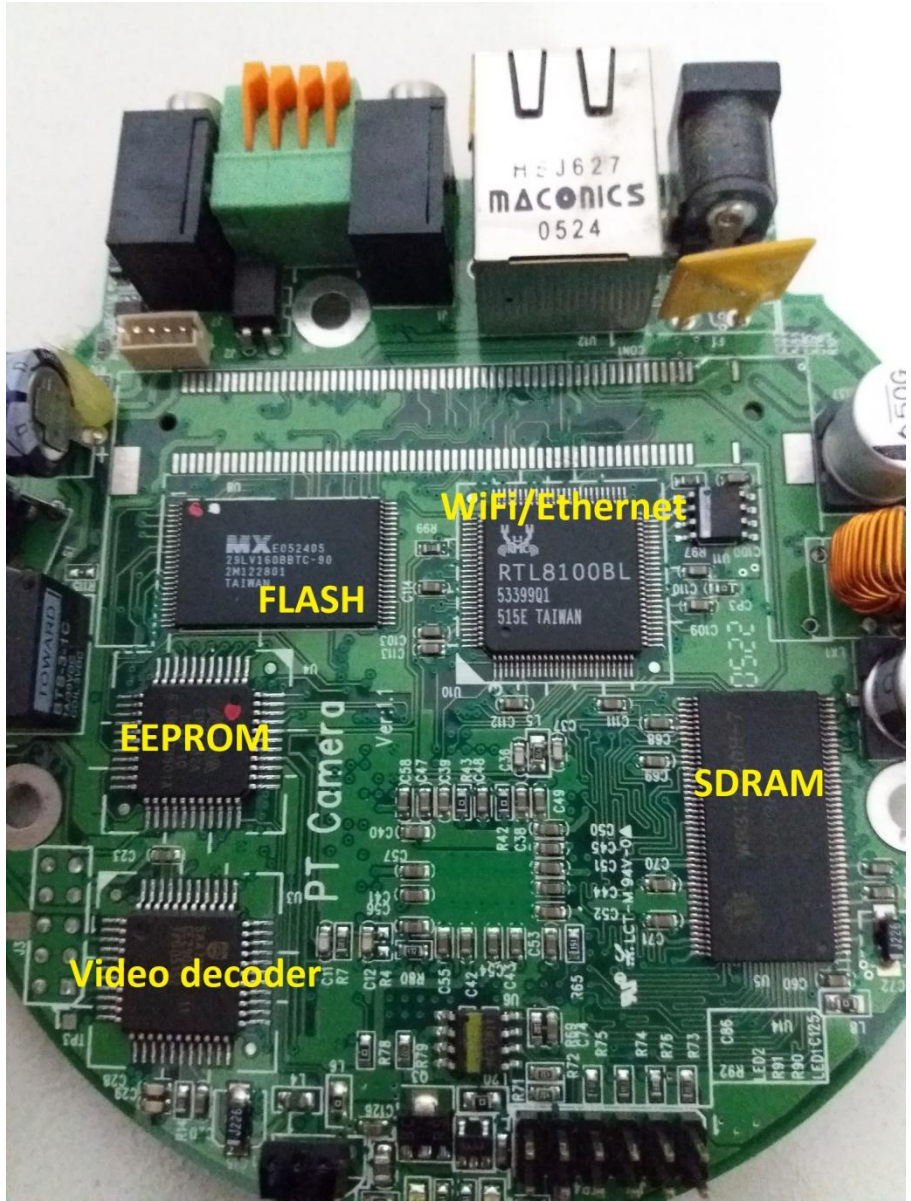
- Radio Free Ethernet (RFE) (TCP 5002) is a network audio broadcasting system. It consists of programs and tools that allow packets of audio data to be transmitted around a network. The system is best understood by using the analogy of traditional radio broadcasting [1]
  - The camera has an incorporated microphone so this service is probably used to broadcast the captured audio
- The filemaker (TCP 5003) service is used to stream live video, a kind of RTP/RTSP service. With the D-ViewCam application, we can access the audio and video transmitted by the camera. The transmission is done via HTTP by requesting the video.vam file.
- The commplex-link (TCP 5001) service is used for sync up the audio and video transmitted by the camera [2]

[1] http://baselinesystems.com/mediafiles/pdf/Ethernet_Radio_Config_Guide.pdf
[2] http://www.dlink.co.in/nl/dcs-over-internet.pdf

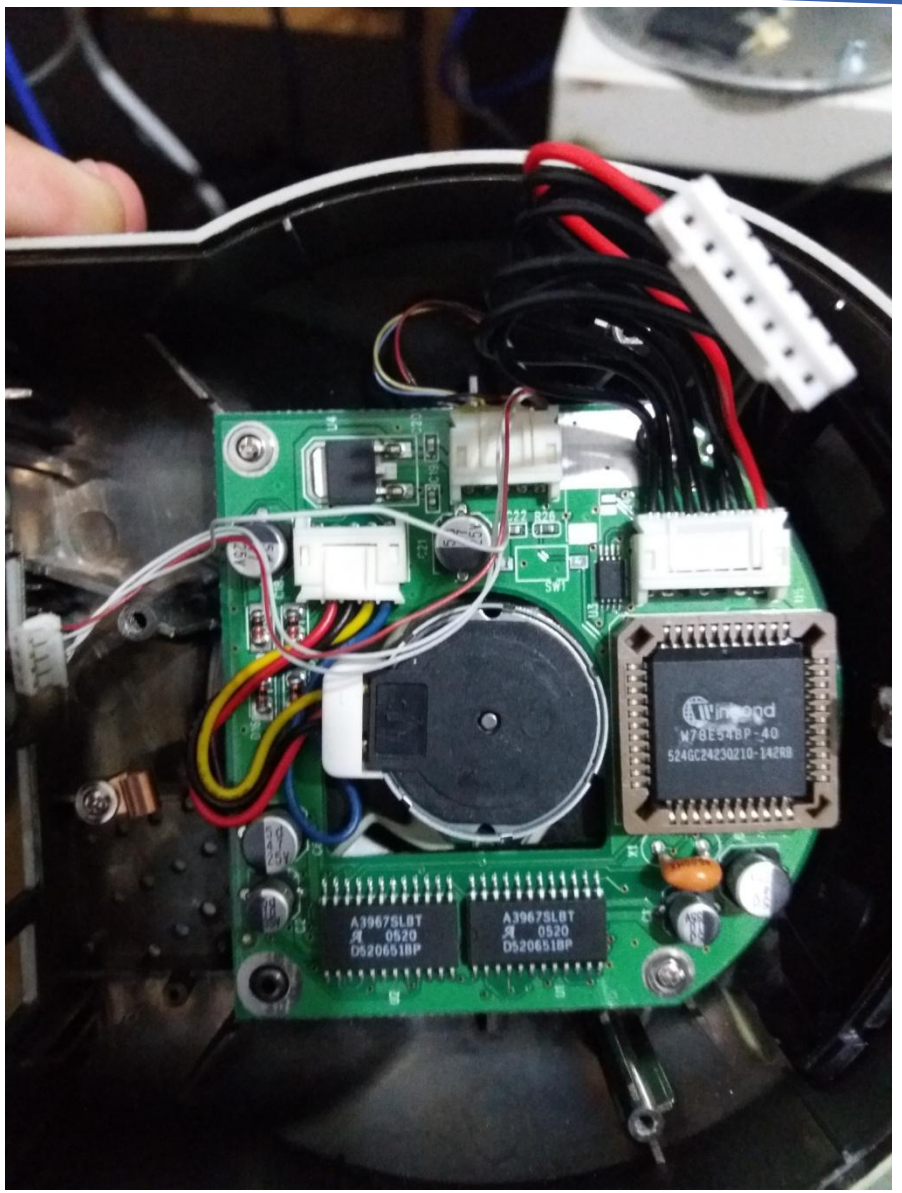
Exploring the hardware

- Camera has two PCBs:
  - One with main components (CPU, RAM, Flash, etc)
    - CPU: **Philips TriMedia PNX1300EH** (32 bit processor)
    - SDRAM: Winbond W9812G2DH-7 SDRAM (128 MB)
    - Flash memory: 16 MB MX29LV160BBTC-90
    - WiFi/Ethernet: RTL8100BL
    - EEPROM: Altera EPM3032A
    - Video decoder: Philips SAA7113H
  - Another one with PTZ functionallity
    - Winbond W78E54BP-40
    - Two Allegro A3967SLBT chips

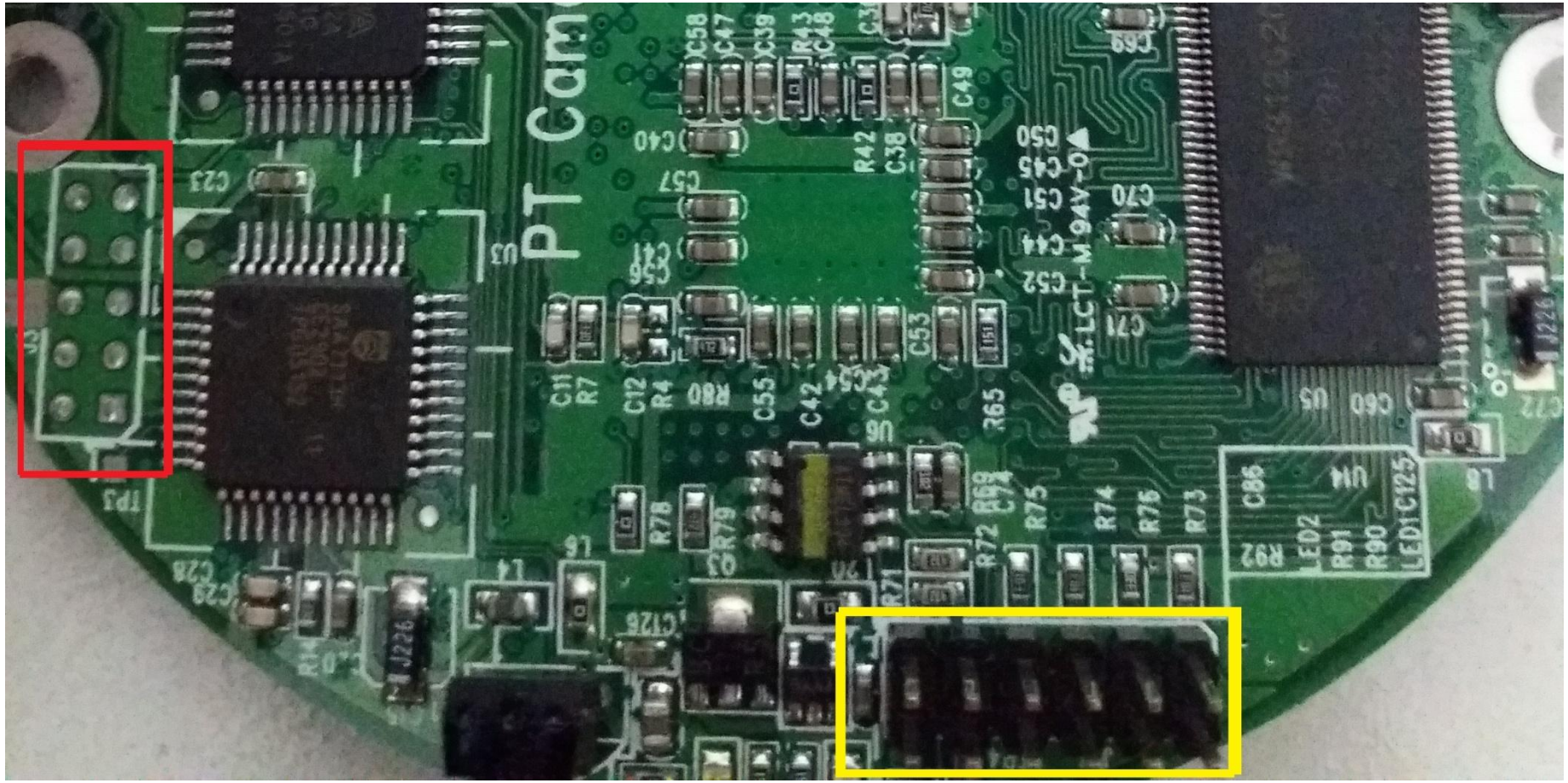
Exploring the hardware – 1st PCB - Front & back



Exploring the hardware – 2nd PCB



Exploring the hardware - Pinout

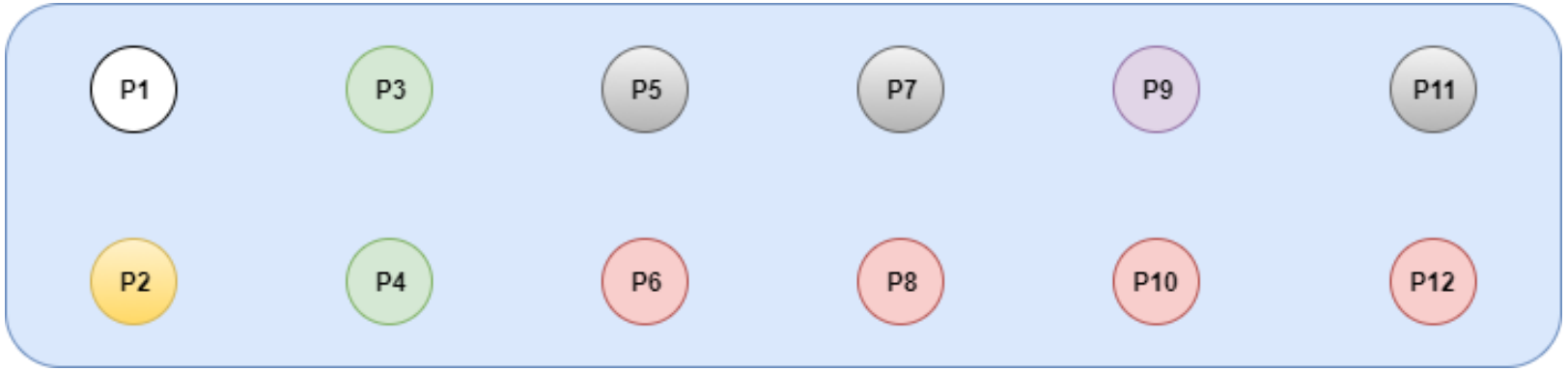


Exploring the hardware - Pinout

- Pinout
  - In chapter 18 "JTAG Functional Specification“ of the specs, we can see that the TriMedia CPU has a JTAG interface
    - Monument Data Systems offers a JTAG PCI debugger
    - Philips also offers one but needs a proprietary JTAG cable (It is a USB device that connects to the PC via a NetChip NET-2282 peripheral controller. The programmer boots from its own EEPROM and is driven by a Philips PNX1502 CPU with NOR flash and DRAM memory) (*)
  - 10 pins marked in red
    - Are a little bit far from the CPU, maybe not so interesting
  - 12 pins marked in yellow
    - More close to the main CPU (some traces seem to be in that direction)
  - So, I tried to identify the pinout with .. a voltmeter :/
    - Not a very clever idea

(*) Quote from asbokid:
http://web.archive.org/web/20140620185105/http://hackingbtbusinesshub.wordpress.com:80/2011/12/16/the-proprietary-trimedia-jtag-tools/

Exploring the hardware - Pinout



- P1: Voltage fluctuation: 0,94 - 1, 14 - 2,x - 3,46 (TX?)
- P2: Voltage fluctuation: 2,x - 3,46 (after a few seconds, remains steady at 3,46) (RX?)
- P3/P4: Fixed voltage: 3,29
- P5/P7/P11: Voltage fluctuation: 3,27 - 3,29 - 3,30
- P9: Fixed voltage: 1,79
- P6/P8/P10/P12: GND

Exploring the hardware - Pinout

Exploring the hardware - Pinout

- Right tool? An Oscilloscope
- With a scope you can check the basic nature of a signal (VCC, GND,Pulled-up line, numeric signal) and guess its parameters (max voltage, frequency, etc).
- Didn’t have any at that time
- It’s possible to build a cheap scope with a BusPirate or a Raspi & Arduino
  - BusPirate: The bandwith and sampling rate aren't good (5720 samples per second). In practice, gives a maximum measurable frequency at about 1kHz
  - https://www.raspberrypi.org/blog/build-oscilloscope-raspberry-pi-arduino/

The Philips TriMedia CPU

The Philips TriMedia CPU – History & Features

- Originally manufactured by Philips, currently known as NXP (Nexperia) semi-conductors
  - Were used in automotive, mobile, IoT and networking solutions
- Mostly used for DSP (Digital Signal Processing)
- Are VLIW (Very Long Instruction Word) processors
  - Can execute multiple operations in parallel
  - 5 to 8 issue slots filled with up to 45 functional units
  - 128 32-bit general purpose registers
- In 2000, there were some efforts to create a 64 bit version of the TriMedia CPU
- In 2010, TriMedia group was terminated
- In 2016, Qualcomm wanted to acquired NXP ($47 billion): https://www.qualcomm.com/news/releases/2016/10/27/qualcomm-acquire-nxp
- In 2018, Qualcomm walked away of buying NXP: https://www.cnbc.com/2018/07/25/qualcomm-is-preparing-to-give-up-on-nxp.html

https://en.wikipedia.org/wiki/TriMedia_(mediaprocessor)

The Philips TriMedia CPU- Additional info

- TriMedia micro-processors run a RTOS (Real Time Operating System) known as **pSOS**
  - There have been some efforts to port the 2.6 Linux kernel branch to run on TriMedia CPU
- There's an official SDK to program TM CPUs (C++ and TM ASM)
- Only available for big companies such as DLink or 2Wire which make specific development for TM CPUs
- There are some TM compliant SDKs
  - Streaming Networks has IADK (Integrated Application Development Kit), specifically designed for PNX1300 family series
- There are development boards equipped with TriMedia CPUs
  - TriREF development board offered by Streaming Networks
  - Costs around US $5,000.00
- Other options can be found on Ebay

The Philips TriMedia CPU – Sources of information

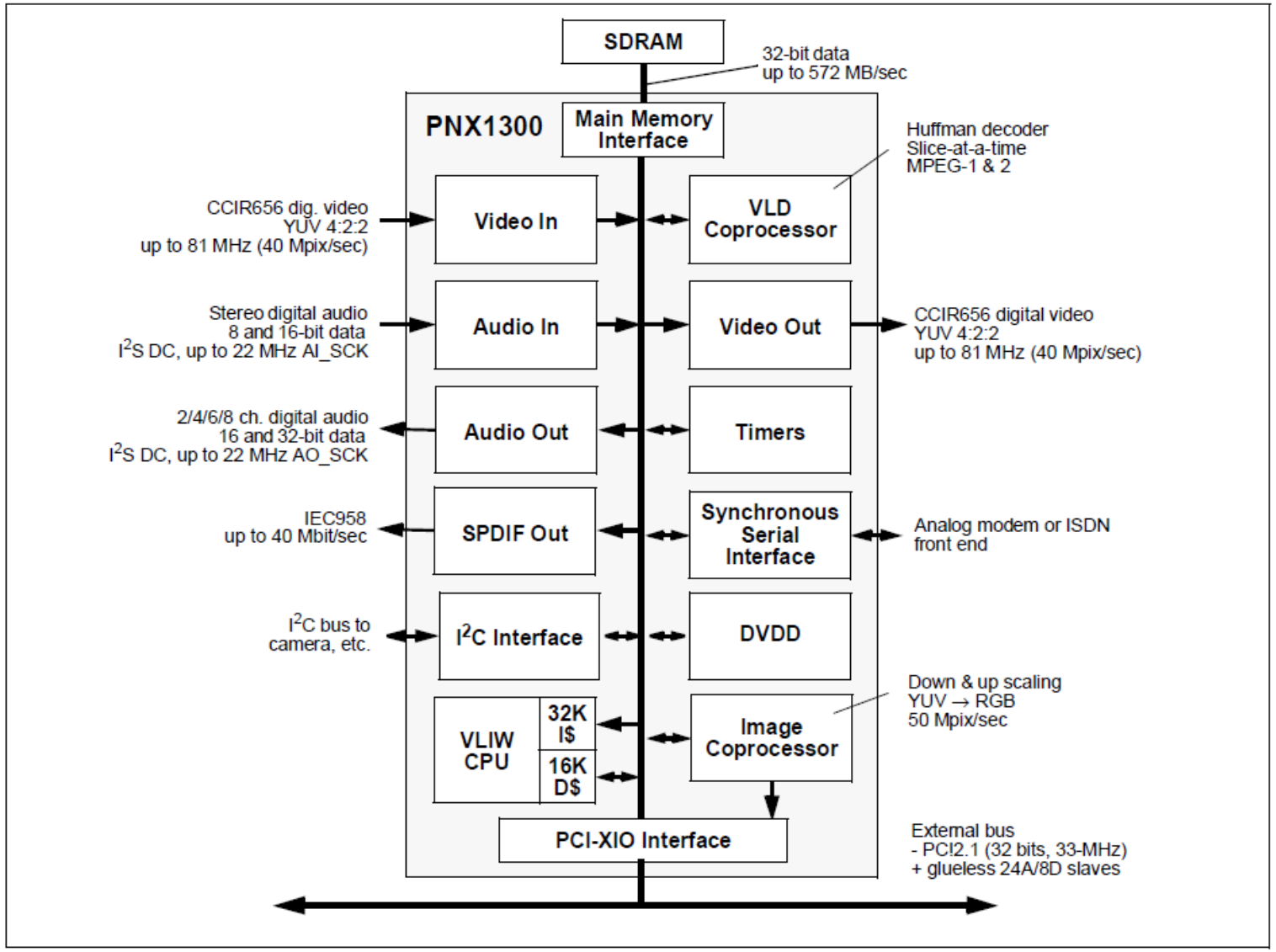
- The “good” and “official” information is private and tools are expensive
  - However, there are some datasheets and US patents around the Internet
- http://hackingbtbusinesshub.wordpress.com was a good resource
  - Not available anymore, anyway some copies can be found on web.archive.org
  - Very nice work reversing 2Write routers based on TriMedia CPUs
- Wrote 2wiglet (based on urjtag) and tm32dis (disassembler)
- IDA has support for TriMedia CPUs since its 4.x version, specifically, its 4.21 version
  - Only available upon special request (in fact, is not available anymore)
- I created a repo with all the TriMedia information and tools I could find during the research
  - https://github.com/crackinglandia/trimedia

The PNX1300EH CPU

- VLIW architecture
  - Other VLIW archs are SHARC & the C6000 processors (*) with more than 5 issue slots
- Maximum speed: 143-MHz at 2.5V
- General-purpose 32-bit CPU
- 128 32-bit general purpose registers
- Implements some media standards such as MPEG-1 and MPEG-2
- 5 issue slots with 27 functional units

(*) https://en.wikipedia.org/wiki/Very_long_instruction_word

The PNX1300EH CPU – Block diagram



The PNX1300EH CPU – Register model

- 128 32-bit general purposes registers
- From r0 to r127
- r0 and r1 have fixed values, 0 & 1, respectively. Used as boolean
- The programmer is not allowed to write to r0 & r1
- PC is the program counter
- Four special registers:
  - PCSW (Program Control and Status Word)
  - DPC (Destination Program Counter)
  - SPC (Source Program Counter)
  - CCOUNT (Counts clock cycles since reset)

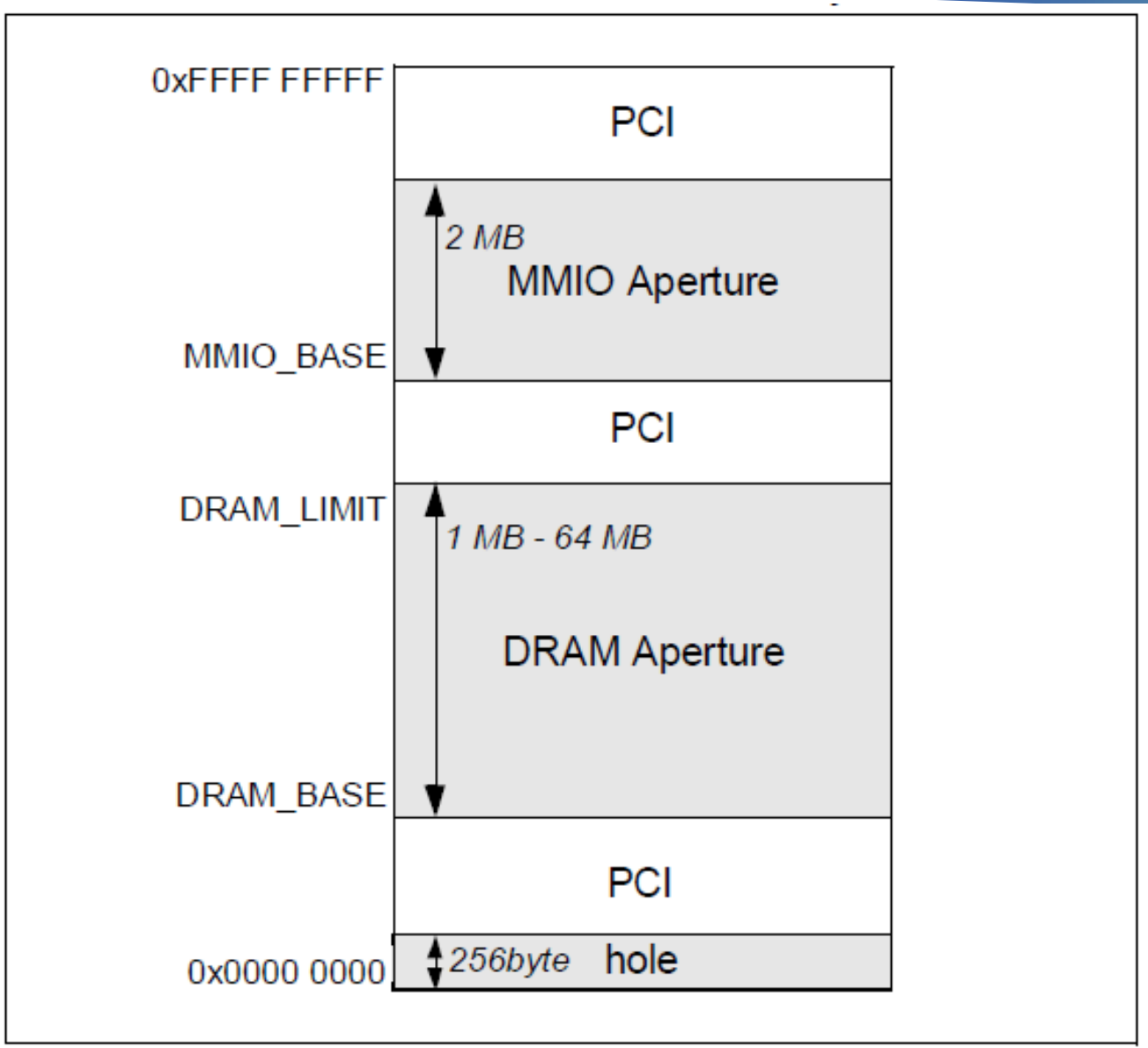
The PNX1300EH CPU – Register model

- The PCSW register is used as a flag register (like EFLAGS in x86/x64)
- The DPC and SPC are registers used for exception processing
- The CCOUNT register, the only 64-bit register in the PNX1300, is used for cycle counting. It counts clock cycles since the last RESET event.
- The PNX1300 CPU has a switchable bytesex (per unit)
  - The switch is done by software by writing the BSX flag (bit size) into the PCSW register
  - This means that little and big endian byte ordering can be found on the same execution
  - load and store operations observe the BSX flag in the PCSW register in order to know if the operation should be done in little or big endian order
  - If the BSX flag is equal to 1, then little endian byte ordering is used. If the BSX flag is 0, then big endian byte ordering is used

The PNX1300EH CPU – Memory & MMIO

- The PNX1300 defines four apertures in a 32 bit address space:
  - The memory hole: mapped from address 0 to 0xFF, thus is 256 bytes size
  - DRAM: mapped from DRAM_BASE to the address in the DRAM_LIMIT registers (max size 64 MB)
  - MMIO: starting at MMIO_BASE and is a fixed 2-MB size
  - PCI: space not marked as DRAM, MMIO or memory hole
- Values for DRAM and MMIO are set at boot time by the BIOS

The PNX1300EH CPU – Memory & MMIO



The TriMedia ASM & Instruction set

- The ASM language used by TriMedia can be described like this:

"The Trimedia processor is a VLIW machine with multiple functional units, where you can get up to 5 operations in a single instruction word. The assembler for that is a multi-pass beast that is as close to magic as I've ever encountered in an assembler."

The TriMedia ASM & Instruction set

Just to give you an example:

```
1  (* instruction 0    : 224 bits (28 bytes) long *)
2  (* offset          : 0x00000000 *)
3  (* bytes           : 00 18 4c 0c c0 80 c0 81 c3 80 c0 b5 c0 81 02 00 12 00 8c 00 20 90 40 40 40 20 a0 d0 *)
4  (* format bytes    : 0x0018 & 0xff03 = 0x0000, format in little endian bit order: 00 00 00 00 00  *)
5     IF r1   uimm(0x61a618) -> r0,                (* 42 bits: 0 02 30 c0 0c 4c *)
6     IF r7   ijmpi(0x90030001),                   (* 42 bits: 2 40 81 81 c0 80 *)
7     IF r2   fadd r67 r1 -> r32,                  (* 42 bits: 1 01 02 c0 80 c3 *)
8     IF r10  bitand r64 r3 -> r16,                (* 42 bits: 0 81 02 02 81 c0 *)
9     IF r1   uimm(0xd0060024) -> r0;              (* 42 bits: 3 42 83 00 12 00 *)
```

The TriMedia ASM & Instruction set

- Operation != Instruction -> each TM instruction has 5 operations
- As many other ASM languages, TM different types of operations:
  - **Mathematical operations** (integer and floating point) E.g: iadd, isub, imul, fdiv
  - **Logical** E.g: bitand, bitor, bitxor, bitinv
  - **Load/Store** E.g: alloc, allocd, ild16, ld32, st8
  - **Control-flow** (branches) E.g: ijmpf, ijmpt, ijmpi
  - **Multimedia & DSP** E.g: imax, imin, quadumax, h_dspidualabs
  - **Sign** E.g: sex16, sex8
- All operations can be executed in parallel, up to five
- Each operation has its own functional unit (a predefined slot in the CPU that dispatches certain group of operations)
  - i.e. Mathematical integer operations go to the ‘ALU' unit, control flow operations go to the 'branch' unit, immediate operations go to the 'const' unit, etc

The TriMedia ASM & Instruction set

- (Almost) All operations can be (optionally) “guarded” (except for iimm and uimm)
- A guarded operation executes conditionally, depending on the value in the 'guard' register
- Any of the 127 registers (except for those that have a special use) can be used to “guard” an operation
- Example:

IF R23 iadd R14 R10 → R13

- This should be taken to mean

if R23 then R13 ← R14 + R10

The PNX1300EH CPU – Addressing modes

- Addressing modes, these can be summarized in the following table:

Mode	Suffix	Applies to	Name
R[i] + scaled(#j)	d	Load & Store	Displacement
R[i] + R[k]	r	Load only	Index
R[i] + scaled(R[k])	x	Load only	Scaled index

- R[i] indicates one of the general purpose registers
- The 'i' and 'k' parameter can have a value between 0 to 127
- The 'j' parameter can be between -64 and 63
- Example: ‘ld32d(–8) r3’ loads a 32-bit value from address (r3 – 8)

The TriMedia ASM – Compression scheme

- Sources of information:
  - US PATENT 5787302 Software for producing VLIW instruction compression
  - TM32 disassembler tool: https://sourceforge.net/projects/tm32dis/

- Reminder:
  - Instruction != Operation
  - We can have up to 5 operations in one instruction
  - Each operation can belong to any of the 27 different functional units
  - Each operation can be dispatched in any of the 5 issue slots

The TriMedia ASM – Compression scheme

- General considerations:
  - The length of each instruction will vary depending on the size of each operation
  - Operations size can be 26, 34 or 42 bits
  - Operations can be guarded or unguarded
  - Operations can be zeroary, unary or binary (0, 1 or 2 operands)
  - Operations can be resultless
  - Operations can contain immediate parameters (7 or 32 bits)
  - Operations are all compressed except for the branch op

The TriMedia ASM – Compression scheme

- Format bits:
  - 2 bits that accompany an operation
  - Provide additional information about the operation (listed in the previous slide)
  - Are located in the prior instruction
- 5 operations per instruction: that means that we have a 10 format bits in total (2 for each of the 5 operation), thus, one byte plus 2 bits are used
- In general: 2*N format bits for a N-issue slot machine
  - Bits are organized in N groups of 2 bits

The TriMedia ASM – Compression scheme

- Format bits are referred in matrix notation as Format[j] where j is the bit number
- Bits Format[2i] and Format[2i+1] give format information about issue slot i, where 0<=i<=N

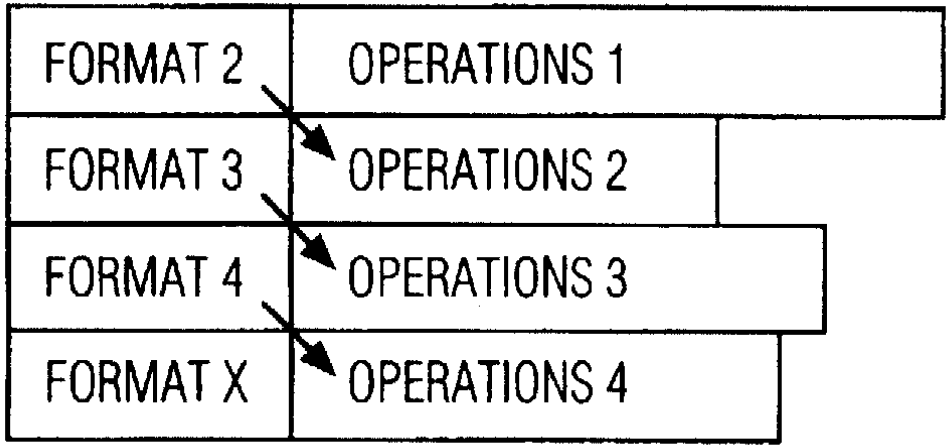
Format (2i) (lsb)	Format (2i+1) (msb)	Meaning
0	0	Issue slot i is used and an operation for it is available in the instruction. The operation size is 26 bits. The size of the extension is 0 bytes
1	0	Issue slot i is used and an operation for it is available in the instruction. The operation size is 34 bits. The size of the extension is 1 byte
0	1	Issue slot it is used and an operation for it is available in the instruction. The operation size is 42 bits. The size of the extension is 2 bytes
1	1	Issue slot is unused and no operation for it is included in the instruction

- For example, if Format = {1, 1, 1, 1, 1, 0, 1, 0, 1, 0}, then the instruction contains three 34 bits operations ({1,0}, {1,0}, {1,0}).

The TriMedia ASM – Compression scheme

- Operations can have 26, 34 and 42 bits
  - 26-bit operations are broken up into a 2-bit part to be stored with the format bits and a 24-bit part
  - 34-bit operations are broken up into a 2-bit part, 24-bit part and one byte extension
  - 42-bit operations are broken up into a 2-bit part, a 24-bit part and two byte extensions
- Extension bytes are used to extend the size of the operation from the basic 26 bit to 34 or 42 bit, if needed

The TriMedia ASM – Compression scheme



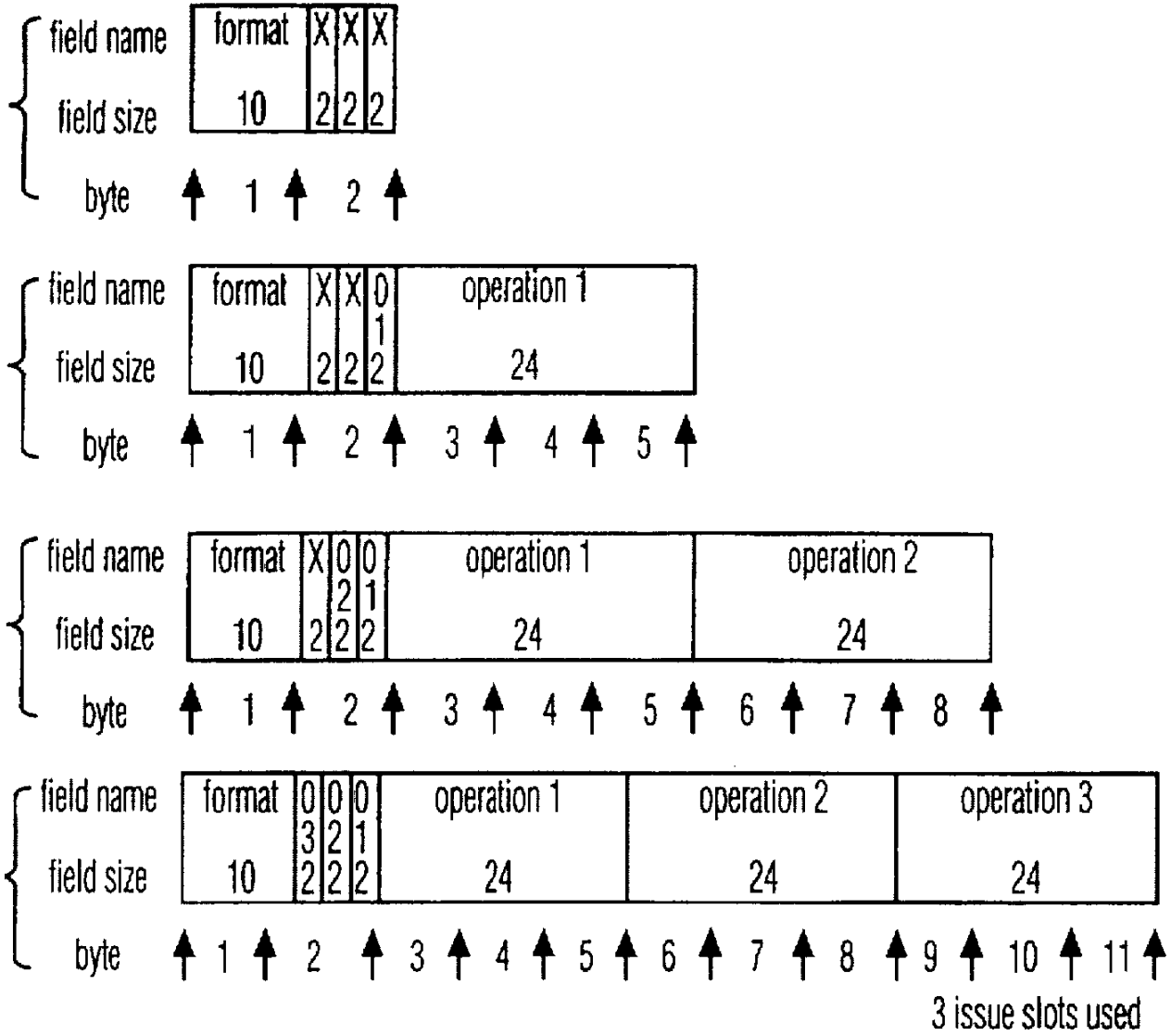
INSTRUCTION 1 - BRANCH TARGET, UNCOMPRESSED

INSTRUCTION 2 - COMPRESSED

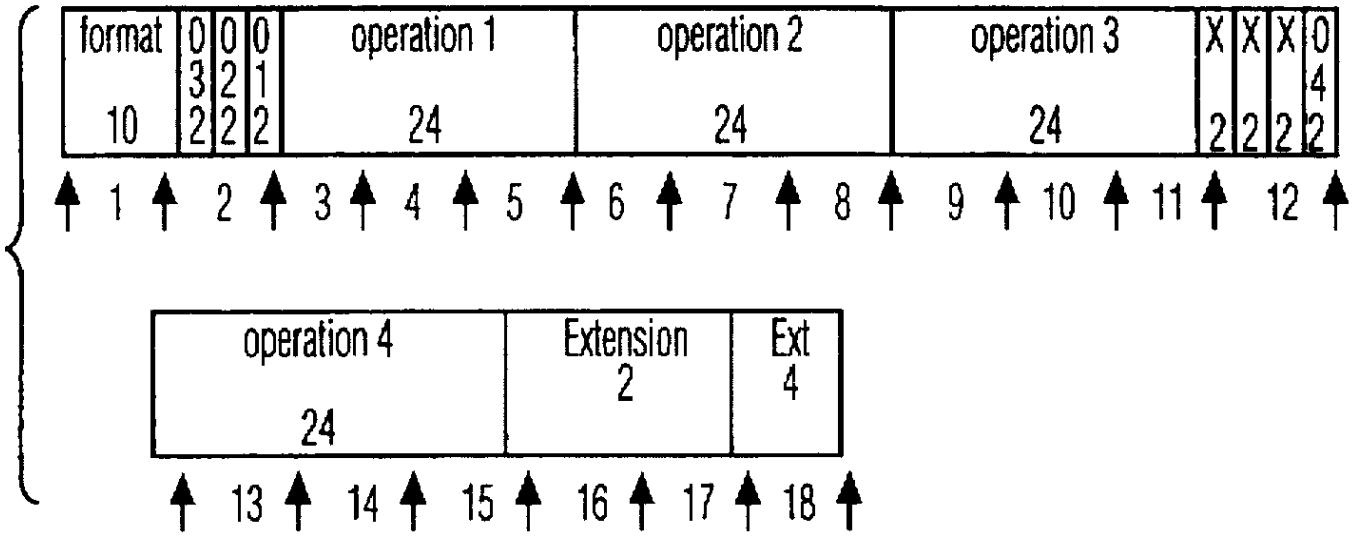
INSTRUCTION 3 - COMPRESSED

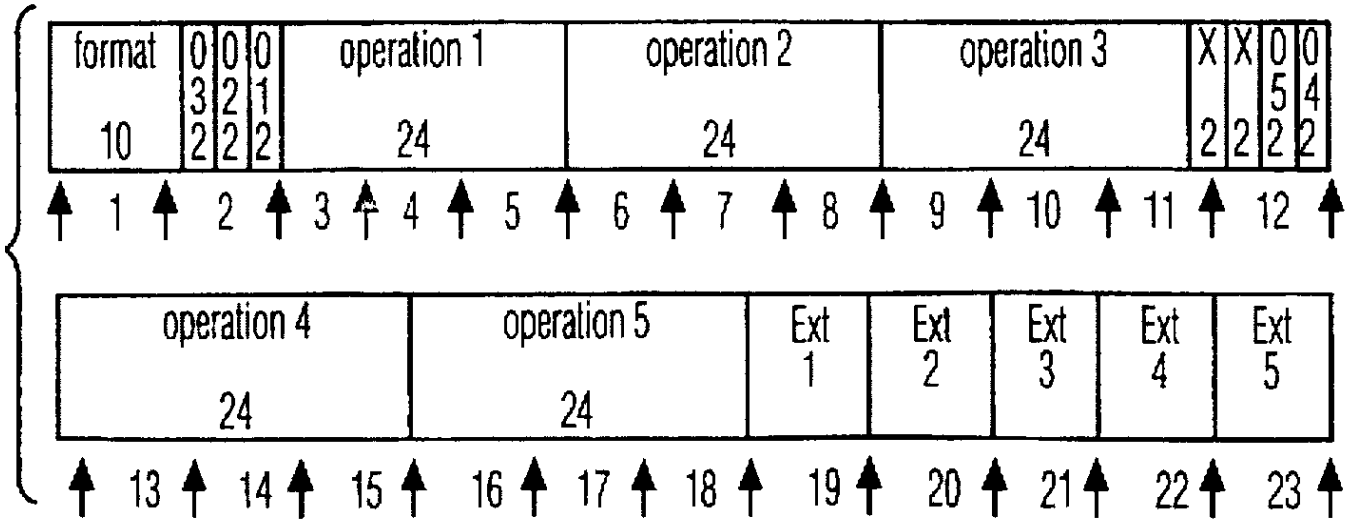
INSTRUCTION 4 - COMPRESSED

The TriMedia ASM – Compression scheme



The TriMedia ASM – Compression scheme





The TriMedia ASM – Disassembling

- Execution flow is divided into "Decision Trees" (dtree)
- Each Decision Tree has a given number of instructions
- Decision Tree is the scheduling unit or building block for a TriMedia VLIW core, it can be seen as a function in a high-level language
- The beginning of a dtree is indicated with the format bytes 0xAA02
  - These bytes are used to encode a branch target instruction
- Four basic steps:
  - Get instruction length
  - Get operation size
  - Unpack operation
  - Decode operation

The TriMedia ASM – Disassembling

Disassembling a TM instruction:

- Get instruction length
  - Sum of sizes of operations (subtract format bits). If len(ins) > 3, add 8 bits for formats field of second group
- Get operation size
  - Given by formats bits
- Unpack operation
  - Black Magic!
- Decode operation
  - More Black Magic!

The TriMedia ASM – Disassembling

- Now, remember the 1 MB of data at the beginning of the firmware image?
- dd if=dcs5300_firmware_105.bin of=almost_first_mb.bin bs=1 count=978627
- tm32dis.exe -i almost_first_mb.bin > output.txt

The TriMedia ASM – Disassembling

```
(* instruction 0    : 224 bits (28 bytes) long *)
(* offset           : 0x00000000 *)
(* bytes            : 00 18 4c 0c c0 80 c0 81 c3 80 c0 b5 c0 81 02 00 12 00 8c 00 20 90 40 40 40 20 a0 d0 *)
(* format bytes     : 0x0018 & 0xff03 = 0x0000, format in little endian bit order: 00 00 00 00 00  *)
   IF r1   uimm(0x61a618) -> r0,                (* 42 bits: 0 02 30 c0 0c 4c *)
   IF r7   ijmpi(0x90030001),                   (* 42 bits: 2 40 81 81 c0 80 *)
   IF r2   fadd r67 r1 -> r32,                  (* 42 bits: 1 01 02 c0 80 c3 *)
   IF r10  bitand r64 r3 -> r16,                (* 42 bits: 0 81 02 02 81 c0 *)
   IF r1   uimm(0xd0060024) -> r0;              (* 42 bits: 3 42 83 00 12 00 *)

(* instruction 1    : 144 bits (18 bytes) long *)
(* offset           : 0x0000001c *)
(* bytes            : e0 60 41 40 02 c0 c5 65 d0 41 18 08 10 80 42 5f 20 60 *)
(* format bytes     : 0xe060 & 0xff03 = 0xe000, format in little endian bit order: 00 00 01 11 00  *)
   IF r1   uld8d(0) r65 -> r9,                  (* 26 bits: 1 02 40 41 *)
   IF r1   asl r64 r11 -> r23,                  (* 26 bits: 2 65 c5 c0 *)
   IF r1   igtri(3) r80 -> r97,                 (* 26 bits: 0 18 41 d0 *)
   IF r1   ilesi(0) r16 -> r10,                 (* 26 bits: 0 42 80 10 *)
   26: ILLEGAL OP! = ineqi;                     (* 26 bits: 0 60 20 5f *)

(* instruction 2    : 136 bits (17 bytes) long *)
(* offset           : 0x0000002e *)
(* bytes            : 00 20 60 22 26 3c a0 a4 c0 c0 80 82 c0 89 a8 a5 24 *)
(* format bytes     : 0x0020 & 0xff03 = 0x0000, format in little endian bit order: 00 00 00 00 00  *)
   IF r1   igeqi(-60) r96 -> r24,               (* 26 bits: 0 26 22 60 *)
   IF r18  ifixrz r60 -> r64,                   (* 26 bits: 2 a4 a0 3c *)
   IF r1   uimm(0x24292001) -> r3,              (* 42 bits: 0 92 94 80 c0 c0 *)
   IF r1   nop,                                 (*  0 bits: *)
   IF r34  ifixrz r64 -> r19;                   (* 26 bits: 2 a8 89 c0 *)
```

The TriMedia ASM – Disassembling

TODO

- A plugin for IDA to disassemble and navigate TM code would be great (work in progress project)
- More docs
- More affordable and open tools for amateurs (development boards, SDK, hardware debuggers)
  - Reminds me of Halvar Flake's "Closed, heterogenous platforms and the (defensive) reverse engineers dilemma" talk (*)

(*) https://www.sstic.org/2018/presentation/2018_ouverture/

Conclusion

Conclusion

- Now, really …
  - I was a little bit stressed but reversing a TriMedia based device was a real challenge
    - Given by a lack of tools and docs
    - Also because of the complexity of the architecture itself
  - Learnt lots of new things
  - I'll probably end up looking for more devices with TriMedia CPUs ☺

Quarkslab

SECURING EVERY BIT OF YOUR DATA