Dualboot

Firsov Nikita

Raccoon Security

Moscow, August 25, 2022

ABOUT ME

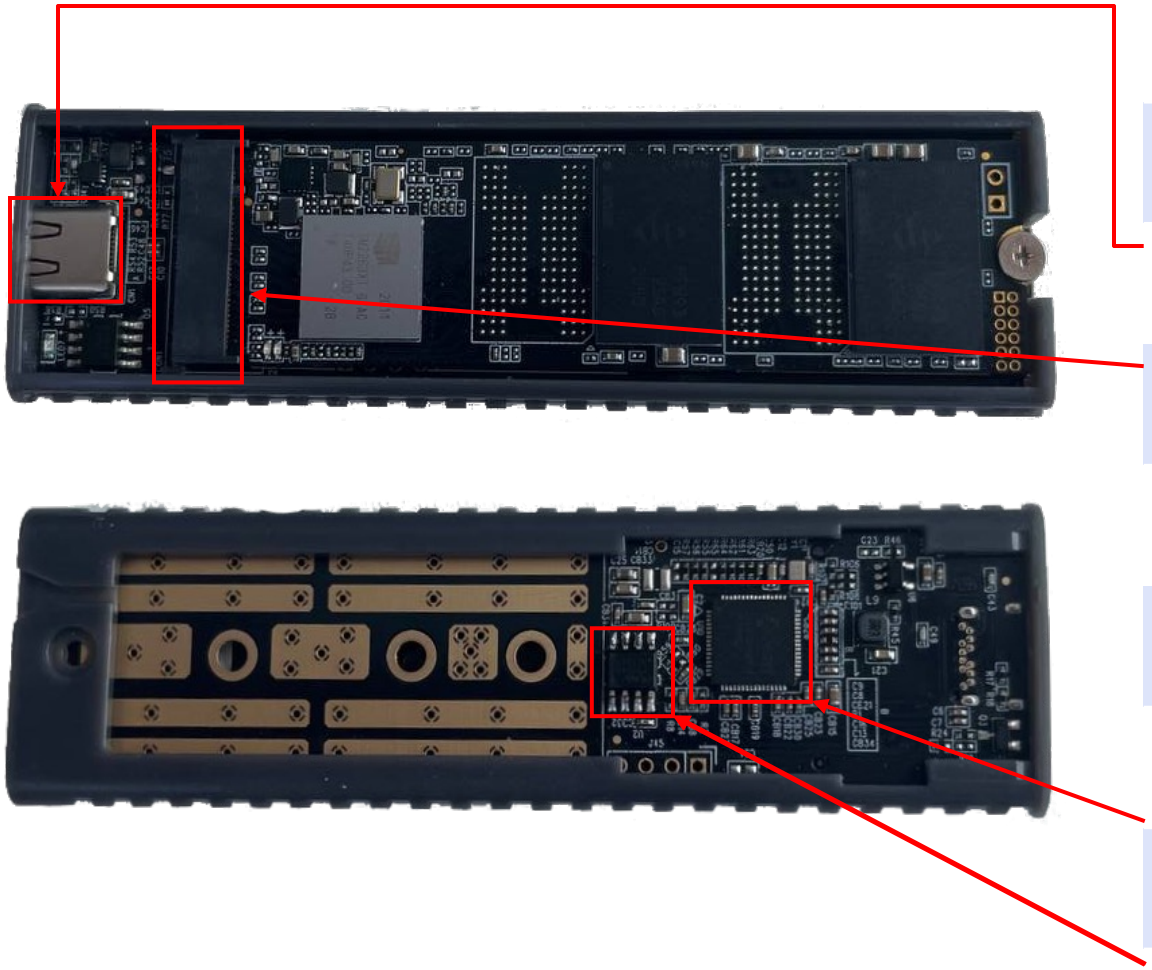
Firsov
Nikita
Raccoon
Security

AGENDA

1. Intro
2. Get Firmware
3. CRC32
4. Dualboot

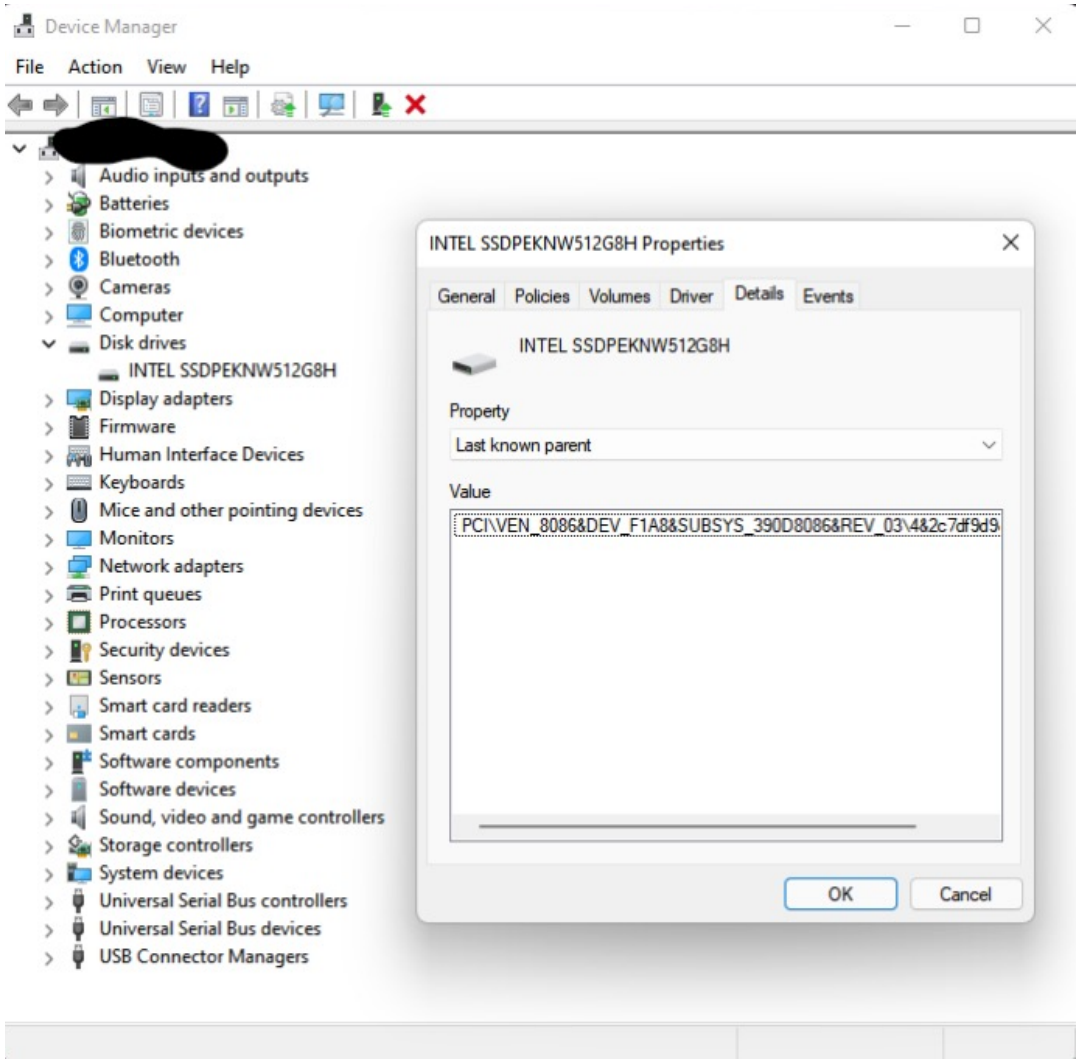
- NVMe-USB ADAPTER AGESTAR

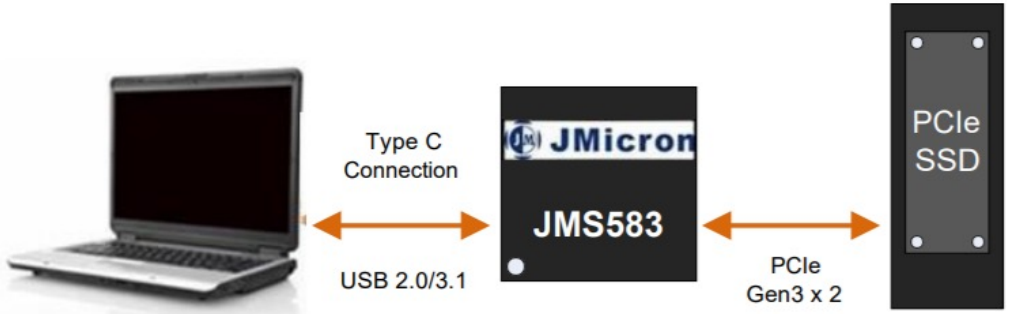
INSIDE AGESTAR 31UBNV1C



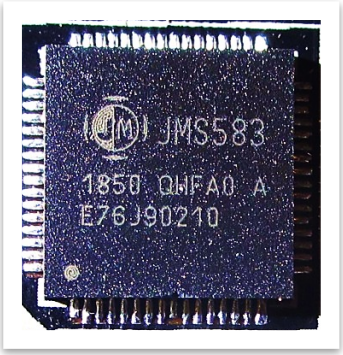
Brand	AGESTAR
Model	31UBNV1C
External interface	USB 3.1 Type-C
Interface	NVMe
Form-factor	M.2
Key	B-key
Size	2230/2242/2260/2280
Controller	JMS583
SPI flash	P25D40H

PCI DRIVE

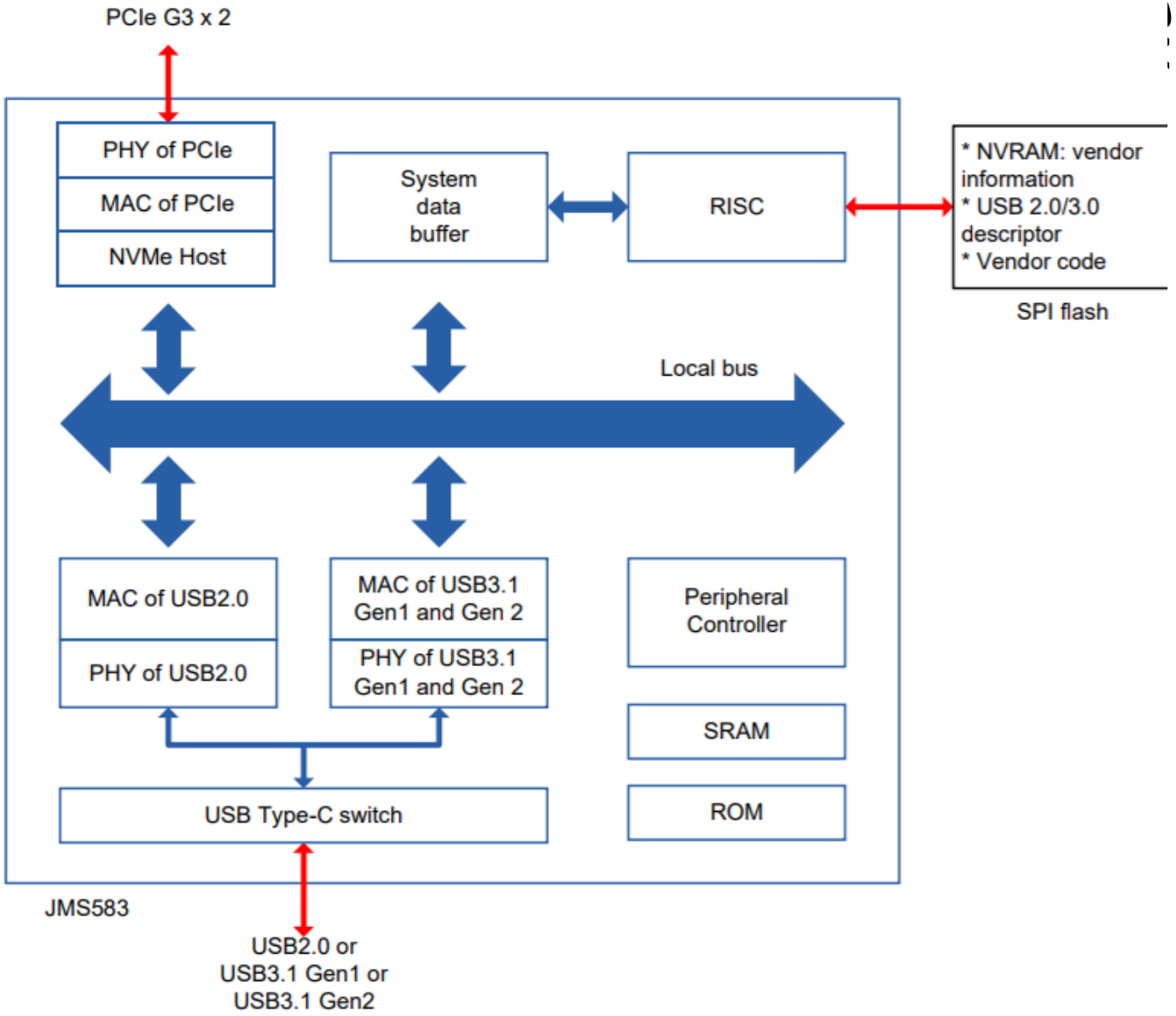

Device Manager
File Action View Help
[redacted]
Audio inputs and outputs
Batteries
Biometric devices
Bluetooth
Cameras
Computer
Disk drives
INTEL SSDPEKNW512G8H
Display adapters
Firmware
Human Interface Devices
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Print queues
Processors
Security devices
Sensors
Smart card readers
Smart cards
Software components
Software devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
Universal Serial Bus devices
USB Connector Managers
INTEL SSDPEKNW512G8H Properties
General Policies Volumes Driver Details Events
INTEL SSDPEKNW512G8H
Property
Last known parent
Value
PCI\VEN_8086&DEV_F1A8&SUBSYS_390D8086&REV_03\4&2c7df9d9
OK
Cancel


Type C
Connection
USB 2.0/3.1
JMicron
JMS583
PCIe
Gen3 x 2
PCIe
SSD

JMS583

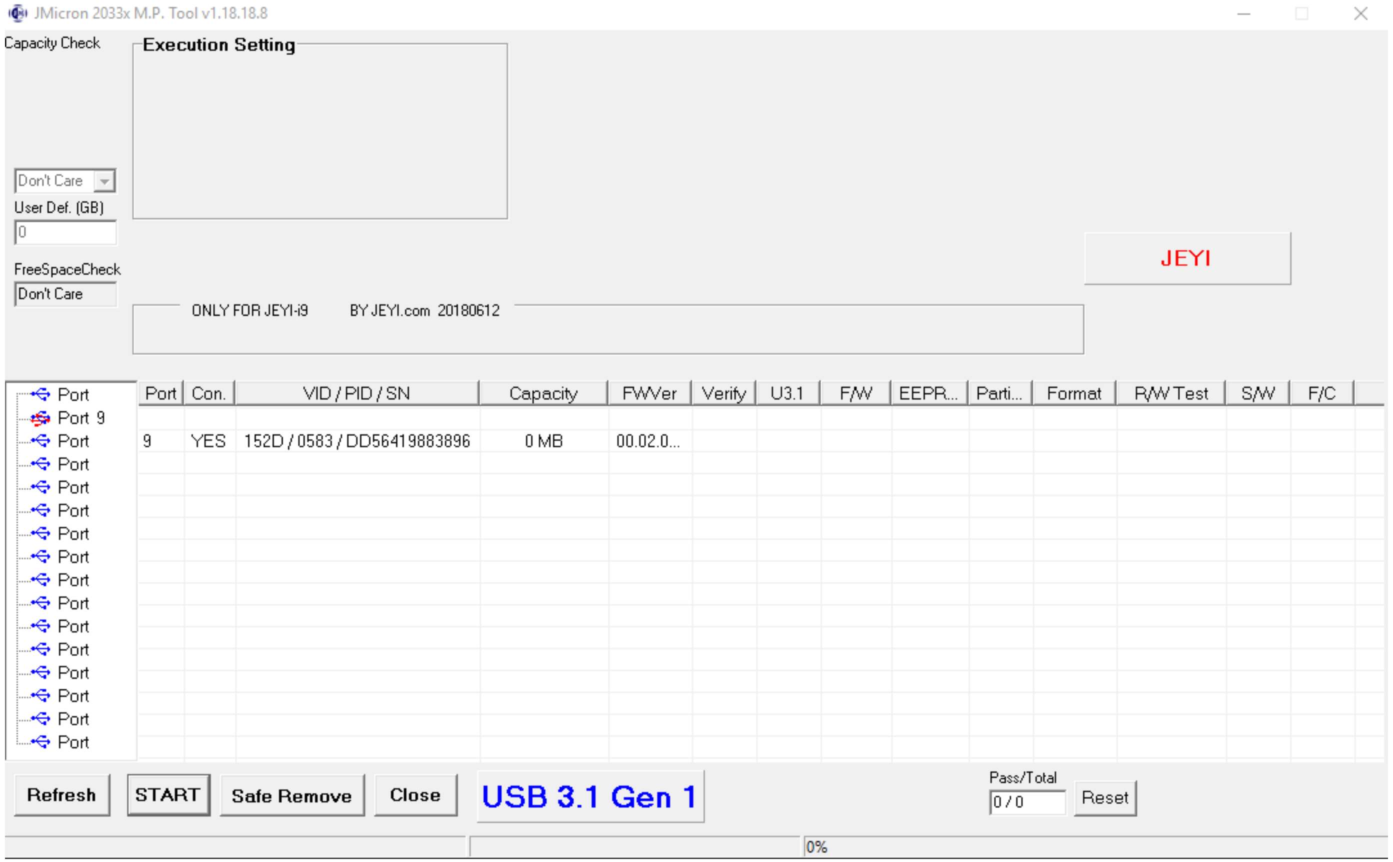


USB 3.1 Gen2 – PCIe Gen3 x2
Поддержка TCG passthru
Поддержка TRIM to the SSD
Скорость до 10 Гбит/с
Соответствует требованиям USB Mass Storage Class BOT и UASP
Поддержка внешнего SPI NVRAM для Vendor VID/PID



- GET FIRMWARE

OSINT



JMicron 2033x M.P. Tool v1.18.18.8

Capacity Check: Don't Care

User Def. (GB): 0

FreeSpaceCheck: Don't Care

Execution Setting

JEYI

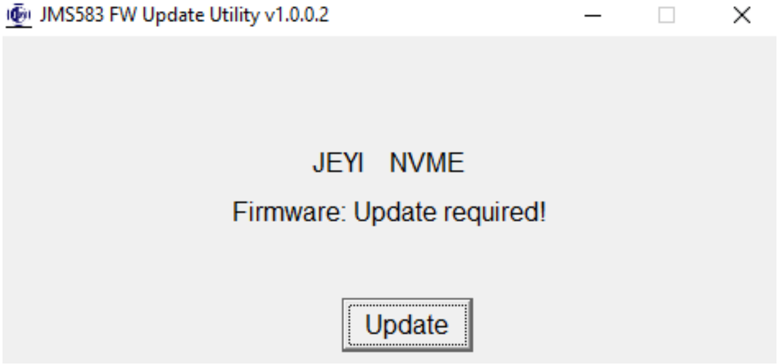
ONLY FOR JEYI-i9 BY JEYI.com 20180612

Port
Port 9
Port
Port
Port
Port
Port
Port
Port
Port
Port
Port
Port
Port
Port
Port

Port	Con.	VID / PID / SN	Capacity	FWVer	Verify	U3.1	F/W	EEPR...	Parti...	Format	R/W Test	S/W	F/C
9	YES	152D / 0583 / DD56419883896	0 MB	00.02.0...									

USB 3.1 Gen 1

Pass/Total: 0 / 0



JMS583 FW Update Utility v1.0.0.2

JEYI NVME

Firmware: Update required!

JMS FIRMWARE

```
00000000  01 00 15 2d 05 83 03 03  05 05 4a 4d 69 63 72 6f  |...-.....JMicro|
00000010  6e 20 4a 4d 53 35 38 33  01 00 00 00 09 00 00 00  |n JMS583........|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 3c 15 b7 7a  |............<..z|
00000200  5a c3 69 e1 00 00 00 00  00 00 00 09 d9 af bc 35  |Z.i............5|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000003f0  00 00 00 00 00 00 00 00  00 00 00 00 b8 3a 1c ad  |.............:..|
00000400  02 b7 6d 02 52 28 7f 01  02 db dc 02 a9 5f 75 2c  |..m.R(........_u,|
00000410  ff 22 31 02 de 21 c3 22  d3 22 32 02 00 1a 32 32  |."1..!."."2...22|
00000420  32 00 00 02 00 1f 22 22  22 00 00 02 00 1e 90 50  |2.....""......P|
00000430  e0 e0 60 06 90 50 23 74  80 f0 22 02 00 20 bb 01  |..`..P#t.."..  ..|
00000440  06 89 82 8a 83 e0 22 50  02 e7 22 bb fe 02 e3 22  |......"P.."....."|
00000450  89 82 8a 83 e4 93 22 bb  01 0c e5 82 29 f5 82 e5  |......".....)...|
00000460  83 3a f5 83 e0 22 50 06  e9 25 82 f8 e6 22 bb fe  |.:..."P..%..."..|
00000470  06 e9 25 82 f8 e2 22 e5  82 29 f5 82 e5 83 3a f5  |..%..."..)....:.|
00000480  83 e4 93 22 bc 00 0b be  00 29 ef 8d f0 84 ff ad  |..."....."......|
00000490  f0 22 e4 cc f8 75 f0 08  ef 2f ff ee 33 fe ec 33  |."...u.../..3..3|
000004a0  fc ee 9d ec 98 40 05 fc  ee 9d fe 0f d5 f0 e9 e4  |.....@..........|
000004b0  ce fd 22 ed f8 f5 f0 ee  84 20 d2 1c fe ad f0 75  |.."..... .....u|
000004c0  f0 08 ef 2f ff ed 33 fd  40 07 98 50 06 d5 f0 f2  |.../..3.@..P....|
000004d0  22 c3 98 fd 0f d5 f0 ea  22 e8 8f f0 a4 cc 8b f0  |"......."......|
000004e0  a4 2c fc e9 8e f0 a4 2c  fc 8a f0 ed a4 2c fc ea  |.,.....,.....,..|
000004f0  8e f0 a4 cd a8 f0 8b f0  a4 2d cc 38 25 f0 fd e9  |.........-.8%...|
00000500  8f f0 a4 2c cd 35 f0 fc  eb 8e f0 a4 fe a9 f0 eb  |...,.5..........|
00000510  8f f0 a4 cf c5 f0 2e cd  39 fe e4 3c fc ea a4 2d  |........9..<...-|
00000520  ce 35 f0 fd e4 3c fc 22  eb 9f f5 f0 ea 9e 42 f0  |.5...<."......B.|
00000530  e9 9d 42 f0 ec 64 80 c8  64 80 98 45 f0 22 eb 9f  |..B..d..d..E."..|
00000540  f5 f0 ea 9e 42 f0 e9 9d  42 f0 e8 9c 45 f0 22 e8  |....B...B...E.".|
00000550  60 0f ec c3 13 fc ed 13  fd ee 13 fe ef 13 ff d8  |`...............|
00000560  f1 22 e8 60 0f ef c3 33  ff ee 33 fe ed 33 fd ec  |."..`...3..3..3..|
00000570  33 fc d8 f1 22 ec f0 a3  ed f0 a3 ee f0 a3 ef f0  |3..."...........|
00000580  22 a8 82 85 83 f0 d0 83  d0 82 12 01 98 12 01 98  |"...............|
00000590  12 01 98 12 01 98 e4 73  e4 93 a3 c5 83 c5 f0 c5  |.......s........|
000005a0  83 c8 c5 82 c8 f0 a3 c5  83 c5 f0 c5 83 c8 c5 82  |................|
000005b0  c8 22 a4 25 82 f5 82 e5  f0 35 83 f5 83 22 d0 83  |."..%.....5..."..|
000005c0  d0 82 f8 e4 93 70 12 74  01 93 70 0d a3 a3 93 f8  |.....p.t..p.....|
000005d0  74 01 93 f5 82 88 83 e4  73 74 02 93 68 60 ef a3  |t.......st..h`..|
000005e0  a3 a3 80 df 8a 83 89 82  e4 73 e5 5e c3 94 10 50  |.........s.^...P|
000005f0  08 74 3a 25 5e f8 c6 ef  c6 ef 60 03 75 5a 02 22  |.t:%^.....`.uZ."|
00000600  90 48 07 ef f0 e4 90 48  09 f0 ef 24 fe 60 43 14  |.H.....H...$.`C.|
00000610  60 4c 24 02 60 03 02 02  cd e4 f5 5e 90 48 09 04  |`L$.`......^.H..|
```

Header 1

Header 2

Code (arch== 8051)

USBC/USBS

```
ROM:8886
ROM:8886          Fill_Usbs:                                  ; CODE XREF: SCSI_Responce↓p
ROM:8886 90 40 00                 mov     DPTR, #TxBuffUsb ; "USBS"
ROM:8889 74 55                    mov     A, #0x55 ; 'U'   ; Move (Op1 <- Op2)
ROM:888B F0                       movx    @DPTR, A         ; Move from/to external RAM
ROM:888C A3                       inc     DPTR             ; Increment Operand
ROM:888D 74 53                    mov     A, #0x53 ; 'S'   ; Move (Op1 <- Op2)
ROM:888F F0                       movx    @DPTR, A         ; Move from/to external RAM
ROM:8890 A3                       inc     DPTR             ; Increment Operand
ROM:8891 74 42                    mov     A, #0x42 ; 'B'   ; Move (Op1 <- Op2)
ROM:8893 F0                       movx    @DPTR, A         ; Move from/to external RAM
ROM:8894 A3                       inc     DPTR             ; Increment Operand
ROM:8895 74 53                    mov     A, #0x53 ; 'S'   ; Move (Op1 <- Op2)
ROM:8897 F0                       movx    @DPTR, A         ; Move from/to external RAM
ROM:8898 22                       ret                      ; Return from subroutine
ROM:8898          ; End of function Fill_Usbs
ROM:8898
```

SCSI COMMANDS

```
ROM:05F3                ROM_5F3:                                ; CODE XREF: vend_cmd_+F↑j
ROM:05F3 90 79 8F                       mov     DPTR, #scsi_cmd ; CSUM err == 0x3b
ROM:05F6 E0                             movx    A, @DPTR        ; Move from/to external RAM
ROM:05F7 12 01 BE                       lcall   u_Switch        ; Long Subroutine Call
ROM:05F7                ; End of function vend_cmd_
ROM:05F7
ROM:05F7                ; ---------------------------------------------------------------------
ROM:05FA 06 73 00+                      uSwitch_st <SCSI_00_TEST_UNIT_READY, 0>; 0
ROM:05FA 06 A9 01+                      uSwitch_st <SCSI_01_REWIND, 1>; 1
ROM:05FA 06 70 03+                      uSwitch_st <SCSI_03_REQUEST_SENCE, 3>; 2
ROM:05FA 06 76 04+                      uSwitch_st <SCSI_04_FORMAT, 4>; 3
ROM:05FA 06 8E 08+                      uSwitch_st <SCSI_08_READ, 8>; 4
ROM:05FA 06 9A 0A+                      uSwitch_st <SCSI_0A_write, 0xA>; 5
ROM:05FA 06 6A 12+                      uSwitch_st <SCSI_12_INQUIRY, 0x12>; 6
ROM:05FA 06 D0 15+                      uSwitch_st <SCSI_15_55_mode_select, 0x15>; 7
ROM:05FA 06 CD 1A+                      uSwitch_st <SCSI_1A_5A_mode_sense, 0x1A>; 8
ROM:05FA 06 BE 1B+                      uSwitch_st <SCSI_1B_START_STOP_UNIT, 0x1B>; 9
ROM:05FA 06 D3 1D+                      uSwitch_st <SCSI_1D_SEND_DIAGNOSTIC, 0x1D>; 0xA
ROM:05FA 06 8B 25+                      uSwitch_st <SCSI_25_READ_CAPACITY, 0x25>; 0xB
ROM:05FA 06 93 28+                      uSwitch_st <SCSI_28_READ, 0x28>; 0xC
ROM:05FA 06 A1 2A+                      uSwitch_st <SCSI_2A_write, 0x2A>; 0xD
ROM:05FA 06 AC 2B+                      uSwitch_st <SCSI_2B_LOCATE, 0x2B>; 0xE
ROM:05FA 06 C7 2F+                      uSwitch_st <SCSI_2F_VERIFY, 0x2F>; 0xF
```

VENDOR COMMANDS

```
ROM:5B35
ROM:5B35          DF__ven_cmd_hndr:                          ; CODE XREF: DF_ven_cmd_getInfo+5↑p
ROM:5B35                                                     ; DF_ven_cmd_getInfo:ROM_6EA↑j
ROM:5B35 90 79 92                mov     DPTR, #scsi_cmd_byte3 ; CSUM err == 0x01 - addr_Hbyte
ROM:5B38 E0                      movx    A, @DPTR           ; Move from/to external RAM
ROM:5B39 FE                      mov     R6, A              ; Move (Op1 <- Op2)
ROM:5B3A A3                      inc     DPTR               ; Increment Operand
ROM:5B3B E0                      movx    A, @DPTR           ; Move from/to external RAM
ROM:5B3C 7C 00                   mov     R4, #0             ; Move (Op1 <- Op2)
ROM:5B3E 24 00                   add     A, #0              ; Add Second Operand to Acc
ROM:5B40 F5 5C                   mov     data_tr_size+1, A ; Move (Op1 <- Op2)
ROM:5B42 EC                      mov     A, R4              ; Move (Op1 <- Op2)
ROM:5B43 3E                      addc    A, R6              ; Add Second Operand to Acc with carry
ROM:5B44 F5 5B                   mov     data_tr_size, A ; Move (Op1 <- Op2)
ROM:5B46 90 79 9A                mov     DPTR, #scsi_cmd_byteB ; Move (Op1 <- Op2)
ROM:5B49 E0                      movx    A, @DPTR           ; Move from/to external RAM
ROM:5B4A 12 01 BE                lcall   u_Switch           ; Long Subroutine Call
ROM:5B4A          ; ---------------------------------------------------------------------------
ROM:5B4D 5C 4A E0+DF_switch_info_cmd:uSwitch_st <vend_cmd_df_e0_or_e1, 0xE0>; 0 ; 0xe0 - ?
ROM:5B4D 5C 4A E1+               uSwitch_st <vend_cmd_df_e0_or_e1, 0xE1>; 1 ; 0xe1 - ?
ROM:5B4D 5C 4F E2+               uSwitch_st <vend_cmd_df_e2, 0xE2>; 2 ; 0xe2 - ?
ROM:5B4D 5C 5B E3+               uSwitch_st <vend_cmd_df_e3, 0xE3>; 3 ; 0xe3 - ?
ROM:5B4D 5B 6C FA+               uSwitch_st <vend_cmd_df_fa_rNvram, 0xFA>; 4 ; 0xfa - read NVRAM
ROM:5B4D 5B 79 FB+               uSwitch_st <vend_cmd_df_fb_wNvram, 0xFB>; 5 ; 0xfb - write NVRAM
ROM:5B4D 5B D1 FD+               uSwitch_st <vend_cmd_df_fd_rxRam, 0xFD>; 6 ; 0xfd - read xRam
ROM:5B4D 5C 22 FE+               uSwitch_st <vend_cmd_df_fe_wxRam, 0xFE>; 7 ; 0xfe - write xRam
ROM:5B4D 5C 32 FF                uSwitch_st <vend_cmd_df_ff_xRam, 0xFF>; 8 ; 0xff - read xRam:3500
ROM:5B4D          ; End of function DF__ven_cmd_hndr
```

READ RAM

```python
def ReadRam(addr, size):
    connect()
    drv.lock()
    cmd = b"\xDF\x10\x00" + struct.pack(">H", size) + b"\x00" + struct.pack(">H", addr) + b"\x00\x00\x00\xFD"
    cmd += b"\x00" * (0x10 - len(cmd))
    response = drv.read(cmd, size)
    drv.close()
    return response
```

```python
def DumpdRam(addr,size):
    connect()
    b=b''
    for bsize in [0x80,0x40,0x20,0x10,0x4]:
        while size>=bsize:
            print(f'{addr:0>4x}')
            response = drv.read(b'\xDF\x10\x00\x00'+struct.pack(">B",bsize)+\
                b'\x00'+struct.pack(">H",addr)+b'\x00\x00\x00\xFD', bsize)
            if type(response)!=bytes or len(response)!=bsize:
                print("error")
                drv.close()
                return b
            b+=response
            size-=bsize
            addr+=bsize
    drv.close()
    return b
```

```
>>> hexdump(DumpdRam(0x7900,0x100))
True
7900
7980
00000000: 55 53 42 43 B0 84 7D 41  80 00 00 00 80 00 0C DF  USBC..}A........
00000010: 10 00 00 80 00 79 00 00  00 00 FD 00 00 00 00 01  .....y..........
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 55 53 42 43 80 B7 01 3D  80 00 00 00 80 00 0C DF  USBC...=........
00000090: 10 00 00 80 00 79 80 00  00 00 FD 00 00 00 00 01  .....y..........
000000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0: 01 00 00 00 20 FF 4E 00  00 FF 23 23 00 00 00 05  .... .N...##....
>>>
```

PCI HEADER

Start	End	Symbol	Name
00h	03h	ID	Identifiers
04h	05h	CMD	Command Register
06h	07h	STS	Device Status
08h	08h	RID	Revision ID
09h	0Bh	CC	Class Codes
0Ch	0Ch	CLS	Cache Line Size
0Dh	0Dh	MLT	Master Latency Timer
0Eh	0Eh	HTYPE	Header Type
0Fh	0Fh	BIST	Built In Self Test (Optional)
10h	13h	MLBAR (BAR0)	Memory Register Base Address, lower 32-bits <BAR0>
14h	17h	MUBAR (BAR1)	Memory Register Base Address, upper 32-bits <BAR1>
18h	1Bh	BAR2	Refer to section 2.1.12
1Ch	1Fh	BAR3	Vendor Specific
20h	23h	BAR4	Vendor Specific
24h	27h	BAR5	Vendor Specific
28h	2Bh	CCPTR	CardBus CIS Pointer
2Ch	2Fh	SS	Subsystem Identifiers
30h	33h	EROM	Expansion ROM Base Address (Optional)
34h	34h	CAP	Capabilities Pointer
35h	3Bh	R	Reserved
3Ch	3Dh	INTR	Interrupt Information
3Eh	3Eh	MGNT	Minimum Grant (Optional)
3Fh	3Fh	MLAT	Maximum Latency (Optional)

GET PCI HEADER DWORDS

```
ROM:35F6                ROM_35F6:                                       ; CODE XREF: MSW_c_9+156↓p
ROM:35F6 7F 10                          mov     R7, #0x10               ; Move (Op1 <- Op2)
ROM:35F8 7E 00                          mov     R6, #0                  ; Move (Op1 <- Op2)
ROM:35FA 12 31 90                       lcall   getPCI_BAR_DWORDS       ; Long Subroutine Call
ROM:35FD 7F 14                          mov     R7, #0x14               ; Move (Op1 <- Op2)
ROM:35FF 7E 00                          mov     R6, #0                  ; Move (Op1 <- Op2)
ROM:3601 12 31 90                       lcall   getPCI_BAR_DWORDS       ; Long Subroutine Call
ROM:3604 90 48 08                       mov     DPTR, #xRam_4808        ; Move (Op1 <- Op2)
ROM:3607 12 01 81                       lcall   wr_xRam32               ; DWORD = 0xffffffff
ROM:360A FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:360B FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:360C FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:360D FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:360E 12 32 E7                       lcall   ROM_32E7                ; Long Subroutine Call
ROM:3611 90 48 08                       mov     DPTR, #xRam_4808        ; Move (Op1 <- Op2)
ROM:3614 12 01 81                       lcall   wr_xRam32               ; DWORD = 0xffffffff
ROM:3617 FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:3618 FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:3619 FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:361A FF                             mov     R7, A                   ; Move (Op1 <- Op2)
ROM:361B 7F 14                          mov     R7, #0x14               ; Move (Op1 <- Op2)
ROM:361D 7E 00                          mov     R6, #0                  ; Move (Op1 <- Op2)
ROM:361F 12 32 EB                       lcall   ROM_32EB                ; Long Subroutine Call
ROM:3622 7F 10                          mov     R7, #0x10               ; Move (Op1 <- Op2)
ROM:3624 7E 00                          mov     R6, #0                  ; Move (Op1 <- Op2)
ROM:3626 12 31 90                       lcall   getPCI_BAR_DWORDS       ; Long Subroutine Call
ROM:3629 90 48 02                       mov     DPTR, #nvram_write_addr ; Move (Op1 <- Op2)
ROM:362C 12 01 75                       lcall   wr_32xRam_r4r7          ; Long Subroutine Call
```

0x10 - Memory Register Base Address (lower 32 bits)
0x14 - Memory Register Base Address (upper 32 bits)

READ PCI IDS

```
vend_cmd_df_ee:
mov     A, data_tr_size+1 ; Move (Op1 <- Op2)
orl     A, data_tr_size ; Logical OR (op1 |= op2)
jz      ROM_E830        ; Jump if Acc is zero
```

```
nop                     ; No operation
nop                     ; No operation
nop                     ; No operation
nop                     ; No operation
mov     R6, #0x35 ; '5' ; Move (Op1 <- Op2)
mov     R7, #0          ; Move (Op1 <- Op2)
mov     R5, #0          ; Move (Op1 <- Op2)
lcall   memclr          ; Long Subroutine Call
mov     R7, #0          ; Move (Op1 <- Op2)
mov     R6, #0          ; Move (Op1 <- Op2)
lcall   getPCI_BAR_DWORDS ; Long Subroutine Call
mov     DPTR, #0x3500   ; Move (Op1 <- Op2)
lcall   wr_32xRam_r4r7  ; Long Subroutine Call
mov     R7, #0x2c ; ',' ; Move (Op1 <- Op2)
mov     R6, #0          ; Move (Op1 <- Op2)
lcall   getPCI_BAR_DWORDS ; Long Subroutine Call
mov     DPTR, #0x3504   ; Move (Op1 <- Op2)
lcall   wr_32xRam_r4r7  ; Long Subroutine Call
mov     SCSI_CMD_processing_status___, #2 ; Move (Op1 <- Op2)
```

```
ROM_E830:
ret                     ; Return from subroutine
; End of function vend_cmd_df_ee
```

00h – Identifiers

2Ch – Subsystem Identifiers

NO FF ONE 2022

- PATCH

FWUPDATE: WIRESHARK

2883	21.963945	host	4.66.1	USB	64 URB_BULK in
2884	21.963957	4.66.1	host	USBMS	77
2885	21.964300	host	4.66.2	USBMS	95 SCSI: Write Buffer LUN: 0x00
2886	21.964314	4.66.2	host	USB	64 URB_BULK out
2887	21.964481	host	4.66.2	USB	1088 URB_BULK out
2888	21.964497	4.66.2	host	USB	64 URB_BULK out
2889	21.964650	host	4.66.1	USB	64 URB_BULK in
2890	22.152610	4.66.1	host	USBMS	77
2891	22.152896	host	4.66.2	USBMS	95 SCSI: Write Buffer LUN: 0x00
2892	22.152915	4.66.2	host	USB	64 URB_BULK out
2893	22.153025	host	4.66.2	USB	1088 URB_BULK out
2894	22.153046	4.66.2	host	USB	64 URB_BULK out
2895	22.153168	host	4.66.1	USB	64 URB_BULK in
2896	22.164888	4.66.1	host	USBMS	77
2897	22.165257	host	4.66.2	USBMS	95 SCSI: Write Buffer LUN: 0x00
2898	22.165279	4.66.2	host	USB	64 URB_BULK out
2899	22.165400	host	4.66.2	USB	1088 URB_BULK out
2900	22.165419	4.66.2	host	USB	64 URB_BULK out
2901	22.165523	host	4.66.1	USB	64 URB_BULK in
2902	22.177255	4.66.1	host	USBMS	77
2903	22.177533	host	4.66.2	USBMS	95 SCSI: Write Buffer LUN: 0x00
2904	22.177548	4.66.2	host	USB	64 URB_BULK out
2905	22.177673	host	4.66.2	USB	1088 URB_BULK out
2906	22.177689	4.66.2	host	USB	64 URB_BULK out
2907	22.177818	host	4.66.1	USB	64 URB_BULK in
2908	22.189526	4.66.1	host	USBMS	77
2909	22.189738	host	4.66.2	USBMS	95 SCSI: Write Buffer LUN: 0x00
2910	22.189754	4.66.2	host	USB	64 URB_BULK out
2911	22.189848	host	4.66.2	USB	1088 URB_BULK out
2912	22.189862	4.66.2	host	USB	64 URB_BULK out
2913	22.189918	host	4.66.1	USB	64 URB_BULK in
2914	22.201702	4.66.1	host	USBMS	77
2915	22.202015	host	4.66.2	USBMS	95 SCSI: Write Buffer LUN: 0x00

```
Data length [bytes]: 31
[Response in: 2886]
[bInterfaceClass: Unknown (0xffff)]
Unused Setup Header
Interval: 0
Start frame: 0
Copy of Transfer Flags: 0x00000000
Number of ISO descriptors: 0
USB Mass Storage
```

```
0000  00 5f 7d 21 d8 91 ff ff  53 03 02 42 04 00 2d 00   ·_}!···· S··B··-·
0010  03 9d d6 62 00 00 00 00  2b e6 09 00 8d ff ff ff   ···b···· +·******
0020  1f 00 00 00 1f 00 00 00  00 00 00 00 00 00 00 00   ******** ********
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ******** ********
0040  55 53 42 43 20 6b d1 3d  00 04 00 00 00 00 0a 3b   USBC k·= *******;
0050  0e 00 00 00 00 00 04 00  00 00 00 00 00 00 00 00   ******** ********
```

Byte \ bit	7	6	5	4	3	2	1	0
0-3	*dCBWSignature*							
4-7	*dCBWTag*							
8-11 (08h-0Bh)	*dCBWDataTransferLength*							
12 (0Ch)	*bmCBWFlags*							
13 (0Dh)	Reserved (0)				*bCBWLUN*			
14 (0Eh)	Reserved (0)			*bCBWCBLength*				
15-30 (0Fh-1Eh)	*CBWCB*							

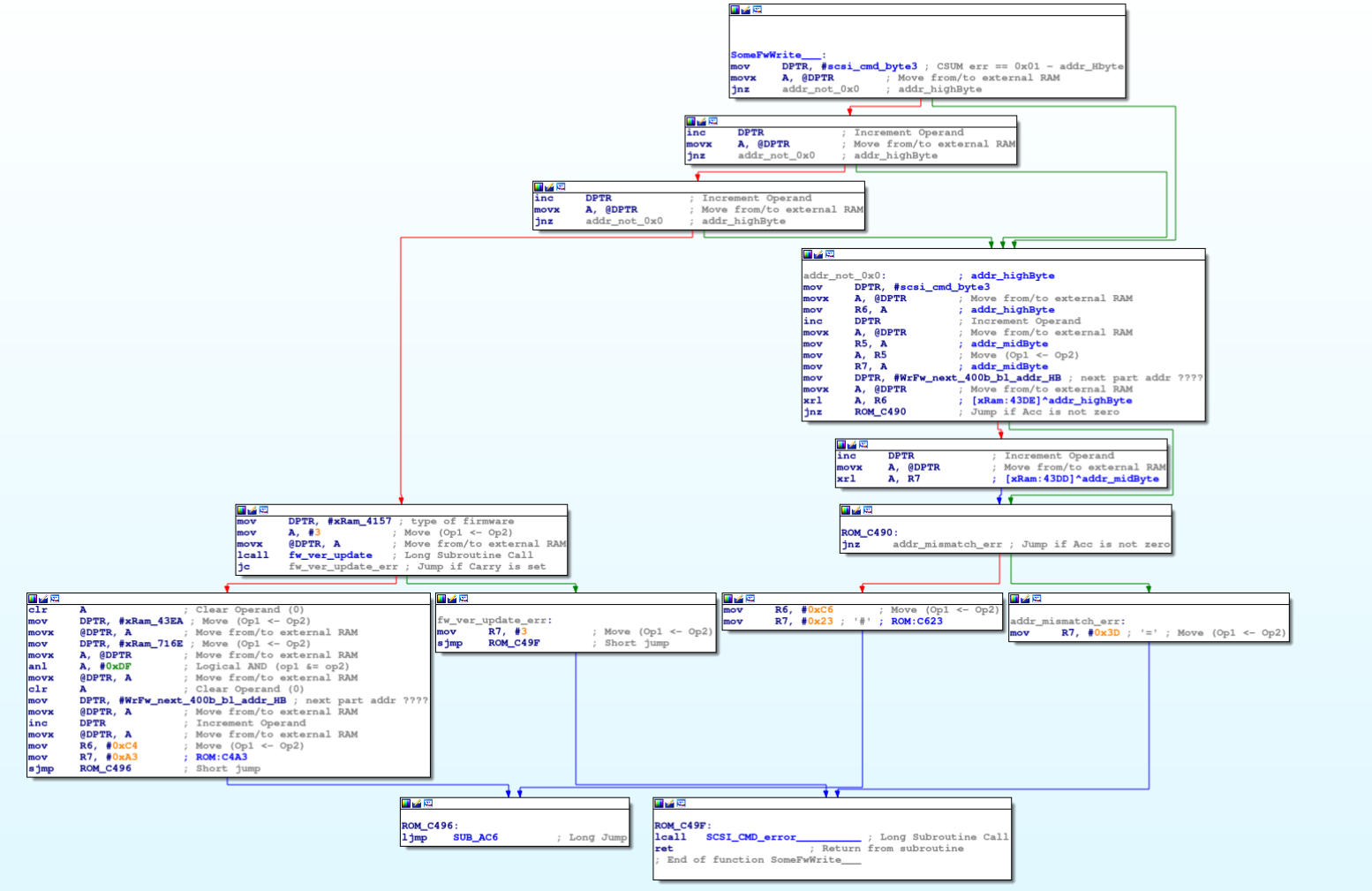
```
55 53 42 43 20 6b d1 3d 00 04 00 00 00 00 0a 3b
0e 00 00 00 00 00 04 00 00 00 00 00 00 00 00
```

3B – WRITE BUFFER

FWUPDATE: CUSTOM FLAGS

```
ROM:C3E2                ; FUNCTION CHUNK AT ROM:7E47 SIZE 0000008C BYTES
ROM:C3E2
ROM:C3E2 12 0A DF                       lcall   ROM_ADF          ; Long Subroutine Call
ROM:C3E5 40 62                          jc      ROM_C449         ; Jump if Carry is set
ROM:C3E7 90 79 90                       mov     DPTR, #scsi_flag ; CSUM err == 0x0e
ROM:C3EA E0                             movx    A, @DPTR         ; Move from/to external RAM
ROM:C3EB B4 FF 03                       cjne    A, #0xFF, ROM_C3F1 ; Compare Operands and JNE
ROM:C3EE 02 62 C6                       ljmp    SCSI_write_buffer_flag_FF ; Long Jump
ROM:C3F1                ; ---------------------------------------------------------------------------
ROM:C3F1
ROM:C3F1                ROM_C3F1:                                ; CODE XREF: SCSI_write_buffer_handl+9↑j
ROM:C3F1 90 79 90                       mov     DPTR, #scsi_flag ; CSUM err == 0x0e
ROM:C3F4 E0                             movx    A, @DPTR         ; Move from/to external RAM
ROM:C3F5 54 1F                          anl     A, #0x1F         ; Logical AND (op1 &= op2)
ROM:C3F7 24 F1                          add     A, #0xF1         ; Add Second Operand to Acc
ROM:C3F9 60 1A                          jz      SCSI_write_buffer_flag_0F ; Jump if Acc is zero
ROM:C3FB 04                             inc     A                ; Increment Operand
ROM:C3FC 70 46                          jnz     Bad_Cmd          ; Jump if Acc is not zero
ROM:C3FE                FLAG=0x0E
ROM:C3FE 90 79 91                       mov     DPTR, #scsi_cmd_byte2 ; buffer_id
ROM:C401 E0                             movx    A, @DPTR         ; Move from/to external RAM
ROM:C402 70 40                          jnz     Bad_Cmd          ; Jump if Acc is not zero
ROM:C404                FLAG=0x0E BUFF_ID=0x00
ROM:C404 7F 03                          mov     R7, #3           ; Move (Op1 <- Op2)
```

FWUPDATE


SomeFwWrite___:
mov DPTR, #scsi_cmd_byte3 ; CSUM err == 0x01 - addr_Hbyte
movx A, @DPTR ; Move from/to external RAM
jnz addr_not_0x0 ; addr_highByte
inc DPTR ; Increment Operand
movx A, @DPTR ; Move from/to external RAM
jnz addr_not_0x0 ; addr_highByte
inc DPTR ; Increment Operand
movx A, @DPTR ; Move from/to external RAM
jnz addr_not_0x0 ; addr_highByte
addr_not_0x0: ; addr_highByte
mov DPTR, #scsi_cmd_byte3
movx A, @DPTR ; Move from/to external RAM
mov R6, A ; addr_highByte
inc DPTR ; Increment Operand
movx A, @DPTR ; Move from/to external RAM
mov R5, A ; addr_midByte
mov A, R5 ; Move (Op1 <- Op2)
mov R7, A ; addr_midByte
mov DPTR, #WrFw_next_400b_bl_addr_HB ; next part addr ????
movx A, @DPTR ; Move from/to external RAM
xrl A, R6 ; [xRam:43DE]^addr_highByte
jnz ROM_C490 ; Jump if Acc is not zero
inc DPTR ; Increment Operand
movx A, @DPTR ; Move from/to external RAM
xrl A, R7 ; [xRam:43DD]^addr_midByte
ROM_C490:
jnz addr_mismatch_err ; Jump if Acc is not zero
mov DPTR, #xRam_4157 ; type of firmware
mov A, #3 ; Move (Op1 <- Op2)
movx @DPTR, A ; Move from/to external RAM
lcall fw_ver_update ; Long Subroutine Call
jc fw_ver_update_err ; Jump if Carry is set
clr A ; Clear Operand (0)
mov DPTR, #xRam_43EA ; Move (Op1 <- Op2)
movx @DPTR, A ; Move from/to external RAM
mov DPTR, #xRam_716E ; Move (Op1 <- Op2)
movx A, @DPTR ; Move from/to external RAM
anl A, #0xDF ; Logical AND (op1 &= op2)
movx @DPTR, A ; Move from/to external RAM
clr A ; Clear Operand (0)
mov DPTR, #WrFw_next_400b_bl_addr_HB ; next part addr ????
movx @DPTR, A ; Move from/to external RAM
inc DPTR ; Increment Operand
movx @DPTR, A ; Move from/to external RAM
mov R6, #0xC4 ; Move (Op1 <- Op2)
mov R7, #0xA3 ; ROM:C4A3
sjmp ROM_C496 ; Short jump
fw_ver_update_err:
mov R7, #3 ; Move (Op1 <- Op2)
sjmp ROM_C49F ; Short jump
mov R6, #0xC6 ; Move (Op1 <- Op2)
mov R7, #0x23 ; '#' ; ROM:C623
addr_mismatch_err:
mov R7, #0x3D ; '=' ; Move (Op1 <- Op2)
ROM_C496:
ljmp SUB_AC6 ; Long Jump
ROM_C49F:
lcall SCSI_CMD_error_____ ; Long Subroutine Call
ret ; Return from subroutine
; End of function SomeFwWrite___

- CRC32

```
crc := 0000... (Startwert)
für alle Bits b im Datenstrom:
  wenn das am weitesten links stehende Bit von crc 1 ist:
    crc := (crc * 2 + b) xor CRC-Polynom
  sonst:
    crc := crc * 2 + b
crc enthält das Ergebnis.
```

CHECKSUM

В прошивке проверок не нашли

В одной из утилит обнаружили
полином для CRC32 0x4c11db7, seed 0xffffffff

Перебором границ нашли 2 crc
от заголовка [0x0-0x1fc] и от кода [0x400-0x103f8]

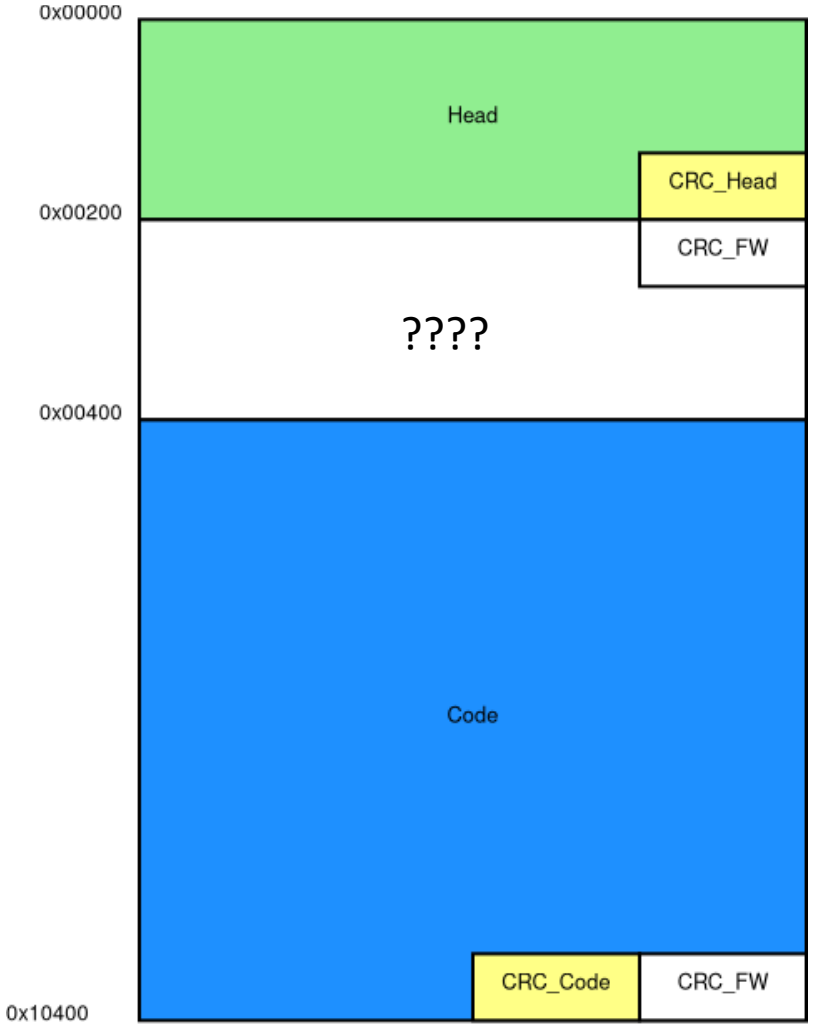
Но это не всё

```
>>> def crc32(val, seed):
...   if (seed & 0x80000000) != (val << 0x1F ):
...     crc = 0x4C11DB7
...   else:
...     crc = 0
...   return (crc ^ (seed << 1)) & 0xffff_ffff
...
...
>>> def crc_byte(val, seed):
...   for i in range(8):
...     seed = crc32((val >> i) & 1, seed)
...   return seed
...
>>> def crc_32b(buffer, seed):
...   for i in range(0, len(buffer), 4):
...       seed = crc_byte(buffer[i+3], seed)
...       seed = crc_byte(buffer[i+2], seed)
...       seed = crc_byte(buffer[i+1], seed)
...       seed = crc_byte(buffer[i], seed)
...   return seed
...
>>> def find_crc32(data):
...     seed = 0xffff_ffff
...     for i in range(0x8, len(data), 4):
...         if i == 8:
...             seed = crc_32b(data[0:i], seed)
...         else:
...             seed = crc_32b(data[i-4:i], seed)
...         if seed == int.from_bytes(data[i:i+4],"big"):
...             print("FOUND in offset: 0x%x" % i)
...
>>> f = open("C:\\Users\\                                    JMS583_FW", "rb")
>>> data = f.read()
>>>
>>> fin
finally:    find_crc32(
>>> find_crc32(data[0:0x400])
FOUND in offset: 0x1fc
FOUND in offset: 0x3fc
>>> find_crc32(data[0x400:])
FOUND in offset: 0xfff8
>>> _
```

HARDWARE CRC32

```
OK at 0x10000
00000000: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 05 02 12 00 01 00 00 00  00 01 00 00 00 02 01 01  ................
00000050: 85 60 13 00 00 00 00 00  00 00 00 00 00 00 00 00  .`..............
00000060: 00 39 00 00 00 12 00 01  00 00 00 00 00 00 0A 49  .9.............I
00000070: 95 3C 5E B3 D6 54 0B 13  F0 55 FA 21 00 00 00 00  .<^..T...U.!....
00000080: C2 68 EC DD 8B 08 91 4E  09 ED 06 8C 6D 29 17 FE  .h.....N....m)..
00000090: 37 06 1A 41 00 00 00 00  00 00 00 00 00 00 00 00  7..A............
000000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
PS C:\Users\я\Desktop\jicron>
```

FW CRC

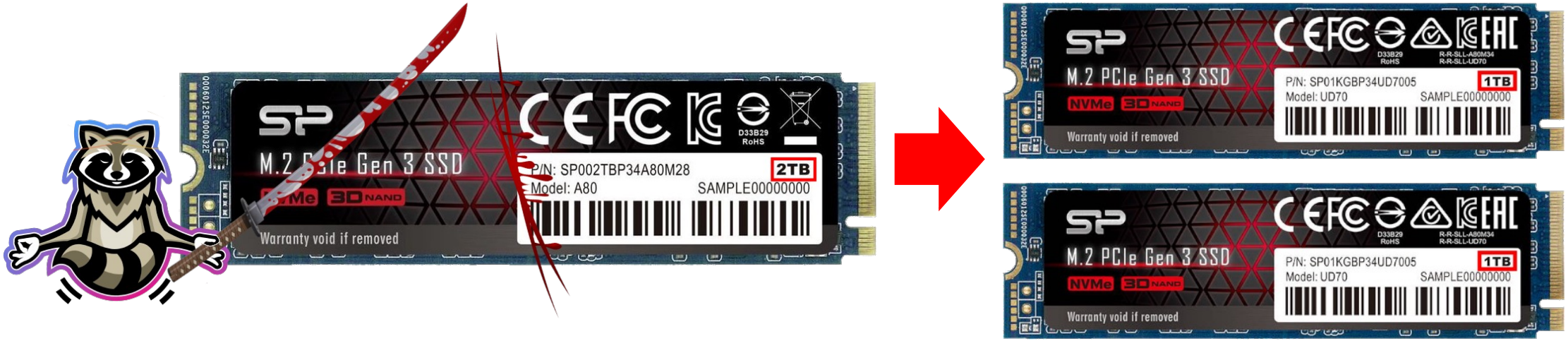



```python

def main():
    x_base=0x7100
    x_size=0x100
    fw0=open('C:\\Users\\                                    \\fw_patch','rb').read()
    seed = 0xffff_ffff
    fw1=fw0[:0x1fc]+struct.pack(">L",crc_32b(fw0[:0x1fc],seed))
    fwC=fw0[0x400:0x103f8]+struct.pack(">L",crc_32b(fw0[0x400:0x103f8],seed))
    fwcrc=struct.pack(">L",crc_32b(fw1+fwC,seed))
    fw=fw1+fw0[0x200:0x20c]+fwcrc+fw0[0x210:0x3fc]
    fw+=struct.pack(">L",0)+fwC+fwcrc
    print(drv.physicalOpen(1))
    drv.lock()
    for addr in range(0x0,0x103ff,0x400):
        a=send_blk(fw[addr:addr+0x400],addr)
        xram=DumpdRam(x_base,x_size)
        clear()
        if a:
            print(f'OK at 0x{addr:x}')
            hexdump(xram)
        else:
            print(f'fail at 0x{addr:x}')
            hexdump(xram)
            drv.unlock()
            drv.close()
            exit(0)
    drv.unlock()
    drv.close()
```

NO OFF ONE 2022

- DUALBOOT



GET CAPACITY

```
ReadCapHndr:
mov     A, RAM_60          ; Move (Op1 <- Op2)
xrl     A, #5              ; Exclusive OR (op1 ^= op2)
jz      ROM_1202           ; Jump if Acc is zero
```

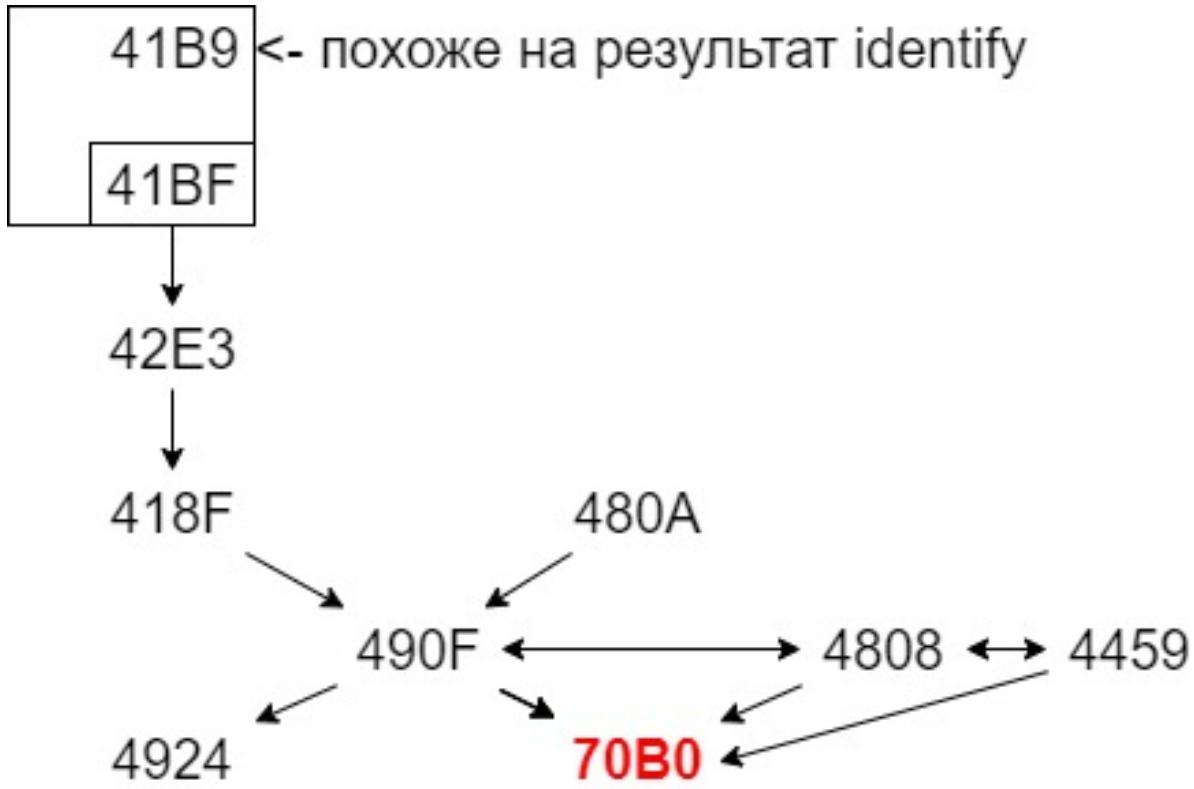
```
ROM_1202:
lcall   ROM_ADF            ; Long Subroutine Call
jnc     ROM_120A           ; Jump if Carry is clear
```

```
ROM_120A:                  ; CSUM err == 0x00 - size_Lbyte
mov     DPTR, #scsi_cmd_byte8
movx    A, @DPTR           ; Move from/to external RAM
jnb     ACC.0, ROM_1213    ; Jump if Bit is clear
```

```
ROM_1213:                  ; CSUM err == 0x00 - buff_id
mov     DPTR, #scsi_cmd_byte2
movx    A, @DPTR           ; Move from/to external RAM
mov     R4, A              ; Move (Op1 <- Op2)
inc     DPTR               ; Increment Operand
movx    A, @DPTR           ; Move from/to external RAM
mov     R5, A              ; Move (Op1 <- Op2)
inc     DPTR               ; Increment Operand
movx    A, @DPTR           ; Move from/to external RAM
mov     R6, A              ; Move (Op1 <- Op2)
inc     DPTR               ; Increment Operand
movx    A, @DPTR           ; Move from/to external RAM
mov     R7, A              ; Move (Op1 <- Op2)
mov     A, R4              ; Move (Op1 <- Op2)
orl     A, R5              ; Logical OR (op1 |= op2)
orl     A, R6              ; Logical OR (op1 |= op2)
orl     A, R7              ; Logical OR (op1 |= op2)
jz      ROM_122B           ; Jump if Acc is zero
```

```
ROM_122B:
mov     DPTR, #xRam_70B6 ; Move (Op1 <- Op2)
mov     A, #0xC0         ; Move (Op1 <- Op2)
movx    @DPTR, A         ; Move from/to external RAM
mov     R6, #0x44 ; 'D'  ; Move (Op1 <- Op2)
mov     R7, #0x59 ; 'Y'  ; Move (Op1 <- Op2)
mov     R3, #6           ; Move (Op1 <- Op2)
mov     R5, #0xB0        ; Move (Op1 <- Op2)
mov     R4, #0x70 ; 'p'  ; Move (Op1 <- Op2)
lcall   memcpy_xRam      ; r4r5-  dst
                         ; r6r7 - src
                         ; r3 - size
mov     DPTR, #xRam_41D1 ; Move (Op1 <- Op2)
movx    A, @DPTR         ; Move from/to external RAM
```

GET CAPACITY




```
ROM_E259:
mov     R6, #0x41 ; 'A' ; Move (Op1 <- Op2)
mov     R7, #0xBF       ; Move (Op1 <- Op2)
mov     R4, #0x42 ; 'B' ; Move (Op1 <- Op2)
mov     R5, #0xE3       ; Move (Op1 <- Op2)
mov     R3, #6          ; Move (Op1 <- Op2)
lcall   memcpy_xRam     ; r4r5-  dst
                        ; r6r7 - src
                        ; r3 - size
mov     R6, #0x42 ; 'B' ; Move (Op1 <- Op2)
mov     R7, #0xE3       ; Move (Op1 <- Op2)
mov     R4, #0x41 ; 'A' ; Move (Op1 <- Op2)
mov     R5, #0x8F       ; Move (Op1 <- Op2)
mov     R3, #6          ; Move (Op1 <- Op2)
ljmp    memcpy_xRam     ; r4r5-  dst
; End of function ROM_E259 ; r6r7 - src
                        ; r3 - size
```

READ/WRITE

```
ROM:068E
ROM:068E
ROM:068E
ROM:068E                SCSI_08_READ:
ROM:068E 12 AD 1C       lcall   ROM_AD1C        ; Long Subroutine Call
ROM:0691 80 03          sjmp    SCSI_88_READ    ; Short jump
```

```
ROM:0693
ROM:0693                SCSI_28_READ:
ROM:0693 12 B3 90       lcall   ROM_B390        ; Long Subroutine Call
```

```
ROM:06A1
ROM:06A1                SCSI_2A_write:
ROM:06A1 12 B3 90       lcall   ROM_B390        ; Long Subroutine Call
```

```
ROM:0696
ROM:0696                SCSI_88_READ:
ROM:0696 7F 01          mov     R7, #1          ; Move (Op1 <- Op2)
ROM:0698 80 0C          sjmp    ROM_6A6         ; Short jump
```

```
ROM:069A
ROM:069A                SCSI_0A_write:
ROM:069A 12 AD 1C       lcall   ROM_AD1C        ; Long Subroutine Call
ROM:069D E4             clr     A               ; Clear Operand (0)
ROM:069E FF             mov     R7, A           ; Move (Op1 <- Op2)
ROM:069F 80 05          sjmp    ROM_6A6         ; Short jump
```

```
ROM:06A4
ROM:06A4                SCSI_8A_write:
ROM:06A4 E4             clr     A               ; Clear Operand (0)
ROM:06A5 FF             mov     R7, A           ; Move (Op1 <- Op2)
```

```
ROM:06A6
ROM:06A6                ROM_6A6:
ROM:06A6 02 A8 97       ljmp    ROM_A897        ; Long Jump
ROM:06A6                ; End of function SCSI_08_READ
ROM:06A6
```

PATCH

```
void size_patch(void)
{
    #pragma asm
        mov     DPTR, #0x41c4
        movx    A, @DPTR
        mov     DPTR, #0xf805
        clr     C
        rrc     A
        movx    @DPTR, A

        mov     DPTR, #0x41c3
        movx    A, @DPTR
        mov     DPTR, #0xf804
        rrc     A
        movx    @DPTR, A

        mov     DPTR, #0x41c2
        movx    A, @DPTR
        mov     DPTR, #0xf803
        rrc     A
        movx    @DPTR, A

        mov     DPTR, #0x41c1
        movx    A, @DPTR
        mov     DPTR, #0xf802
        rrc     A
        movx    @DPTR, A

        mov     DPTR, #0x41c0
        movx    A, @DPTR
        mov     DPTR, #0xf801
        rrc     A
        movx    @DPTR, A

        mov     DPTR, #0x41bf
        movx    A, @DPTR
        mov     A, #0x10
        mov     DPTR, #0xf800
        rrc     A
        movx    @DPTR, A

        ljmp 0x87e5
    #pragma endasm
}
```

```
void lba_patch(void)
{
    #pragma asm
        mov DPTR, #0xfc00
        movx A,@DPTR
        inc A
        movx @DPTR, A

        mov DPTR, #0xf800
        movx A, @DPTR
        mov R3, A
        inc DPTR
        movx A, @DPTR
        mov R4, A
        inc DPTR
        movx A, @DPTR
        mov R5, A
        inc DPTR
        movx A, @DPTR
        mov R6, A



        clr C
        mov DPTR, #0x7994
        movx A, @DPTR
        addc A,R3
        mov DPTR, #0x7998
        movx @DPTR,A

        mov DPTR, #0x7993
        movx A, @DPTR
        addc A,R4
        mov DPTR, #0x7997
        movx @DPTR,A

        mov DPTR, #0x7992
        movx A, @DPTR
        addc A,R5
        mov DPTR, #0x7996
        movx @DPTR,A

        mov DPTR, #0x7991
        movx A, @DPTR
        addc A,R6
        mov DPTR, #0x7995
        movx @DPTR,A

        ret

    #pragma endasm
}
```

```
void read_size_patch(void)
{
    #pragma asm
        mov     DPTR, #0xf803
        movx    A, @DPTR
        mov     DPTR, #0x3500
        movx    @DPTR, A

        mov     DPTR, #0xf802
        movx    A, @DPTR
        mov     DPTR, #0x3501
        movx    @DPTR, A

        mov     DPTR, #0xf801
        movx    A, @DPTR
        mov     DPTR, #0x3502
        movx    @DPTR, A

        mov     DPTR, #0xf800
        movx    A, @DPTR
        mov     A, #0x00
        mov     DPTR, #0x3503
        movx    @DPTR, A

        ljmp 0x1296
    #pragma endasm
}
```

```
void read_longsize_patch(void)
{
    #pragma asm
        mov     DPTR, #0xf803
        movx    A, @DPTR
        mov     DPTR, #0x3504
        movx    @DPTR, A

        mov     DPTR, #0xf802
        movx    A, @DPTR
        mov     DPTR, #0x3505
        movx    @DPTR, A

        mov     DPTR, #0xf801
        movx    A, @DPTR
        mov     DPTR, #0x3506
        movx    @DPTR, A

        mov     DPTR, #0xf800
        movx    A, @DPTR
        mov     A, #0x00
        mov     DPTR, #0x3507
        movx    @DPTR, A

        ljmp 0x1366
    #pragma endasm
}
```

- DEMO

NO FF ONE

2022