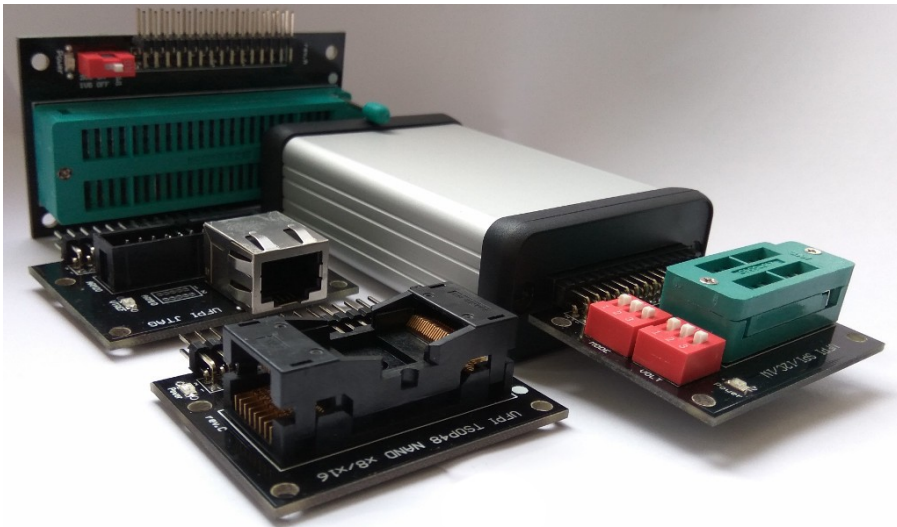


UFPI Programmer

User Manual



24.05.21 REVISION

Оглавление

INTRODUCTION.....	5
1 Download and install.....	6
1.1 Software DOWNLOADING.....	6
1.2 DRIVER INSTALLATION.....	8
2 Description of tabs and menus.....	10
2.1 Main screen (tab).....	10
2.2 TOOLBAR.....	13
2.3 MAIN MENU.....	16
2.4 LOG.....	19
3 Program SETTINGS.....	20
3.1 UFPI tab.....	21
3.2 GUI tab.....	24
3.3 MEDIA.....	25
3.4 BUTTONS.....	26
3.5 MODULES.....	28
3.5.1 NAND.....	29
3.5.2 OneNAND.....	32
3.5.3 Serial NAND.....	34
3.5.4 NOR.....	35
3.5.5 SDMMC.....	36
3.5.6 SPI.....	38
3.5.7 EEPROM.....	39
4 MODULES.....	40
4.1 LOGGER.....	41
4.2 NAND.....	43
4.2.1 Options. Select CE (chip).....	43
4.2.2 ACCESS CYCLE.....	44
4.2.3 «READ IN» и «WRITE FROM».....	44
4.2.4 MODEL.....	45
4.2.5 MODE (access).....	45
4.2.6 BAD BLOCKS (BB).....	45
4.2.7 ECC.....	46
4.2.8 Bad block list management buttons.....	47
4.2.9 Miscellaneous /Additional features. (some of them s appear after ID reading).....	47
4.2.10 Application. Bad Blocks (BB).....	50
4.2.11 Application. Bad Block Table (BBT).....	56
4.2.12 Application. ECC.....	59
4.2.13 Application. Dump analysis. Dump preparation. Creating a "clean" dump.....	61
4.2.14 Application. Dump analysis when reading ID and writing of "clean" dump with control "Use reserved area.".....	62
4.2.15 Description in the log during operation.....	64
4.3 SD/eMMC.....	65
4.3.1 Options.....	66
4.3.2 «Read IN» и «Write FROM».....	66
4.3.3 MODE.....	66
4.3.4 MODEL.....	66
4.3.5 Partition.....	66
4.3.6 4-bit mode / 1-bit mode.....	67

4.3.7	Smart report.....	67
4.3.8	Download configuration.....	67
4.3.9	Full BACKUP.....	67
4.3.10	Platform BACKUP.....	67
4.3.11	Additional features/ Miscellaneous. (some items appear after reading the ID).....	68
4.3.12	Application. CID, CSD, eCSD, OCR registers.....	72
4.3.13	Application. USER, BOOT1, BOOT2, RPMB, GPP1, GPP2, GPP3, GPP4 partitions.....	74
4.3.14	Application. ISP connect mode.....	75
4.3.15	Description of items in the log during operation.....	77
4.4	NOR.....	78
4.5	1-Wire.....	80
4.6	EEPROM I2C.....	81
4.7	EEPROM SPI.....	82
4.8	EEPROM Microwire (3-Wire).....	83
4.9	SPI Flash.....	84
4.10	UART.....	88
4.11	Serial NAND.....	90
4.12	OneNAND.....	92
4.13	BDM.....	95
4.14	JTAG.....	97
4.15	Mount File.....	98
4.16	Mount Chip.....	99
5	RW Mode.....	100
6	Power supply management.....	104
7	.UDEV devices files.....	106
8	GZIP.....	109
9	Working with buffer.....	110
10	Test FILE.....	112
11	File manager and file system (FS) mount.....	113
12	Containers.....	116
13	Programmers Self-diagnostics.....	118
14	Working with scripts.....	119
14.1	SCRIPTS.....	120
15	FAQ.....	122
15.1	Working with FORUM.....	123
15.2	General Questions.....	125
15.3	Commonly called:.....	128
15.4	Power supply during operation.....	129
15.5	NAND.....	129
15.6	EMMC.....	130
15.7	SPI.....	130
15.8	UART.....	130
16	Attachment.....	131
16.1	Programmers, sockets, matching and compatibility.....	131
16.1.1	NAND sets.....	132
16.1.2	SD/eMMC sets.....	133
16.1.3	NOR sets.....	133
16.1.4	Other kits.....	135
16.1.5	Tips and useful information for selecting your adapter kit.....	137
16.2	EXAMPLES.....	139
16.2.1	Attachement. Create a NAND Flash configuration file.....	139

16.2.2 Attachment. Compiling the SPI Flash configuration file.....	141
16.2.3 Attachment. Creating NOR Flash Configuration File.....	142
16.2.4 Attachment. File manager and FS (file system) mount.....	144
16.2.5 Attachment. Reading and writing partitions/regions.....	146
16.2.6 Attachment. Full Backup with RW Mode and .UDEV.....	147
16.2.7 Attachment. Working with Partitions. Split, replace, record.....	148
16.2.8 Attachment. Working with "Platform Backup"	150
16.2.9 Attachment. Checking hashes, keys, restoring and extracting keys and BOOT using the example of the QV14/15 platform.....	151
16.2.10 Attachment. Fix the dump and write the NAND dump using the example of Samsung D5500.	153
16.2.11 Attachment. Containers. Build and writing.....	160
16.2.12 Attachment. Using LOGGER and protocol analysis.....	162
16.3 Pinouts and diagrams.....	163
16.3.1 eMMC EASY JTAG 8-Bit Socket schematics.....	164
16.3.2 eMMC E-Mate Pro 8-Bit Socket schematics.....	165
16.3.3 eMMC E-Mate X 8-Bit Socket schematics.....	166
16.3.4 eMMC ICFRIEND 8-Bit Socket schematics.....	167
16.3.5 SD/eMMC Socket schematics.....	168
16.3.6 SPI/I2C/mWire/1Wire DIP24 Socket schematics.....	169
16.3.7 LOGGER Socket schematics.....	170
16.3.8 NAND Socket schematics.....	171
16.3.9 NOR TSOP48 Socket schematics.....	172
16.3.10 NOR TSOP56 Socket schematics.....	173
16.3.11 OneNAND Socket schematics.....	174
16.3.12 UART/GPIO Socket schematics.....	175
16.3.13 JTAG/BDM Socket schematics.....	176

INTRODUCTION.

This guide is intended to familiarize you with the basics and demonstration of the UFPI hardware and software system. For experienced users, it may also be useful to read the FAQ section of the manual.

UFPI (Universal Flash Programming Interface) - it is a USB powered device based on a modern protected 200 MHz dual-core processor with High-Speed USB Phy (480 Mbps). The main purpose of this interface and software is to program Flash or EEPROM memory and work with basic serial protocols such as JTAG, SPI, I2C, NAND, eMMC, SD, 1W, and so on. The main purposes are maximum speed, simplicity of the interface and maximum number of opportunities for the user. A script language with C-like syntax and direct access to the software, OS, and hardware functions of the UFPI unit allows the user to create (if necessary) their own full-featured script applets without using the IDE, compilers, and other programming equipment. The ability to use external configuration files for the IS allows you to use any necessary parameters, commands and frequency, as well as create any necessary configuration even for "unsupported" chips.

MAIN Features

Very fast speed. For example, works with NAND faster than the "fastest" ChipProg-481.

Windows drivers are not needed. Plug-and-Play for Win10 and Win8 (WCID device).

Power from USB with allowed USB VID and PID.

Bidirectional I/O with voltage levels from 1 V to 5 V

Plug-and-Play sockets. The software automatically selects the appropriate mode and voltage when connected.

Secure and reliable Box updates. Mirror update with AES256.

Support of user chip parameters - geometry, voltages, access time, etc.

Comfortable licensing system. You can use the modules you need.

Powerful C-like scripting language.

Multilingual user interface.

Use multi-part data containers with user-defined compression and encryption.

Functions of FS partitions mounting.

Supports common ECC partition tables and algorithms.

Built-in hexadecimal editor.

Fixed modes for connected sockets.

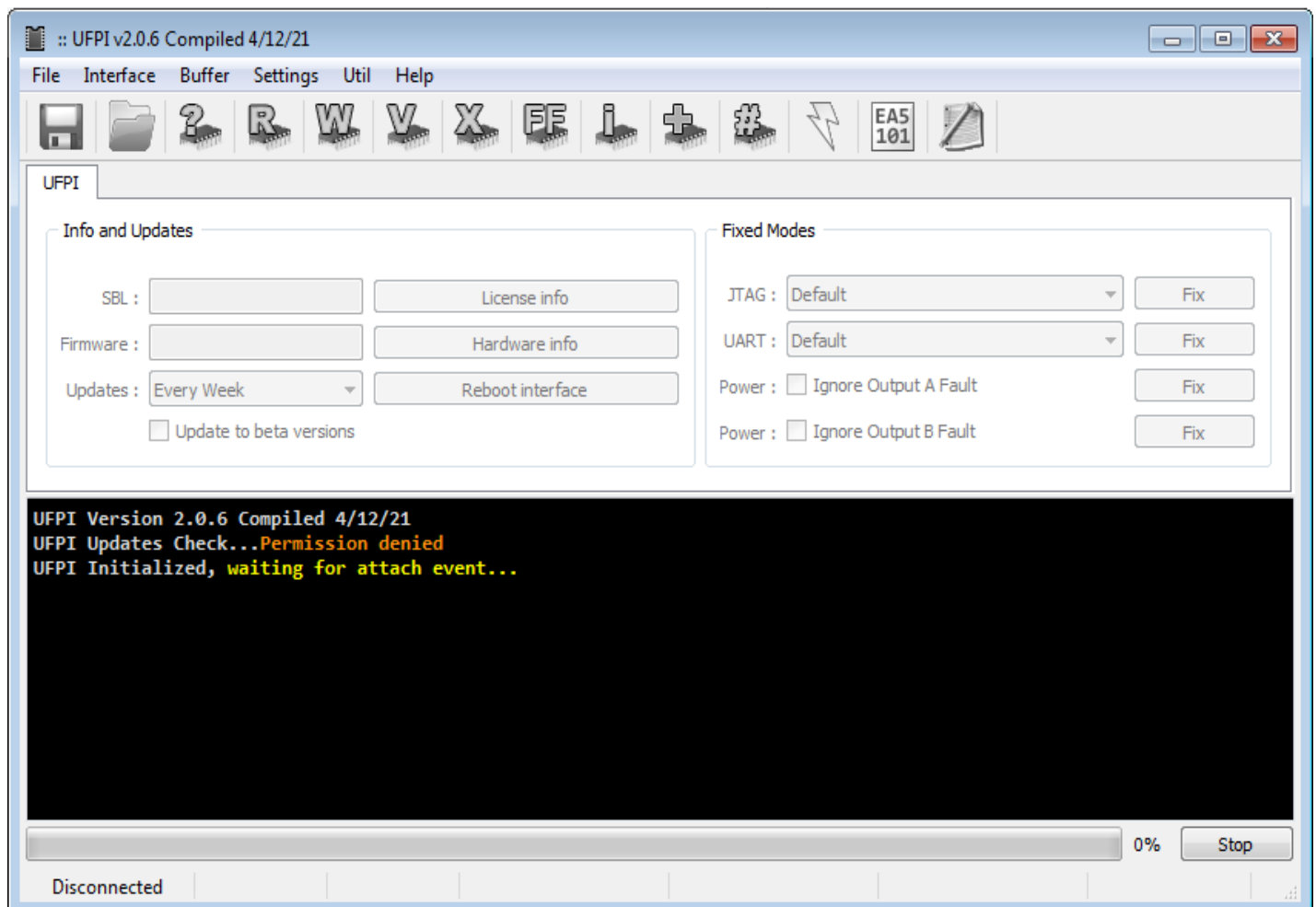
1 Download and install.

System requirements

- Any desktop or laptop with x86/x64 1 GB RAM (recommended 2 GB or more)
- Windows XP/Vista/7/8/10 operating system (and also Linux_x64 and MacOS_x64)
- USB 2.0 High Speed Port (480 Mbps)

1.1 Software DOWNLOADING.

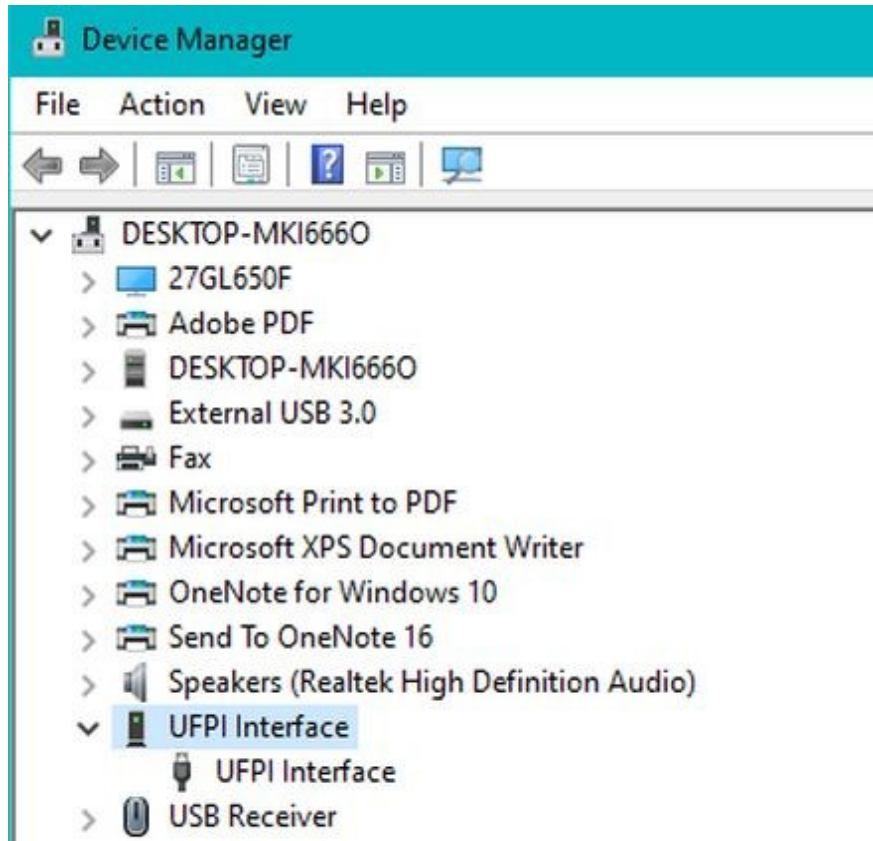
Download the UFPI software via the [link](#), unpack and run ufpi.exe. There is no need to install. The screenshot below shows the UFPI software with the interface disabled or drivers not installed.



1.2 DRIVER INSTALLATION

UFPI programmer is a WCID USB device (Windows Compatible ID).

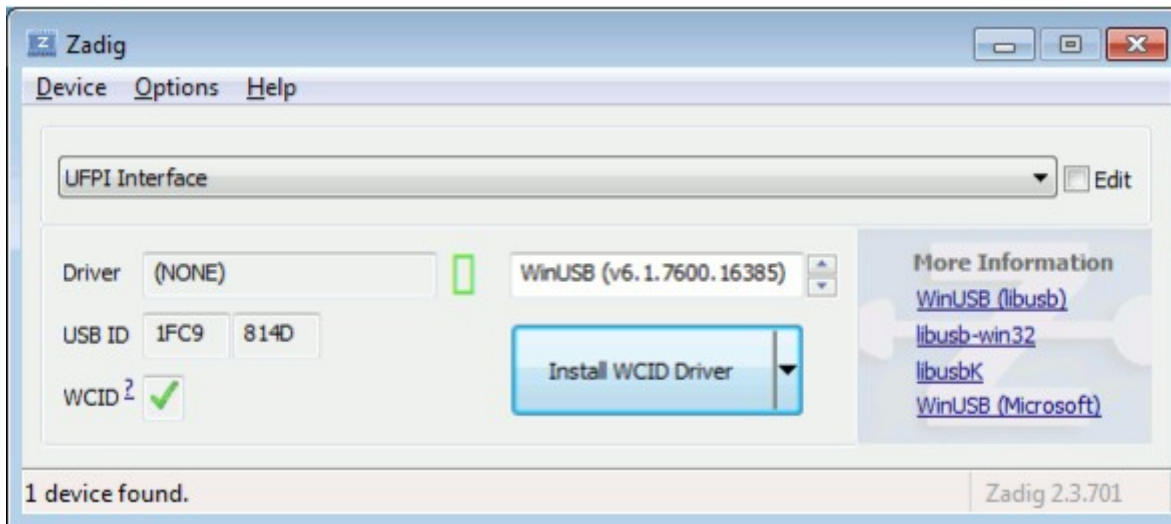
On Windows 8 or 10, the driver will be installed automatically, even without an Internet connection. With UFPI connected, a new USB device should appear in Device Manager. The screenshot below shows the device manager with the Windows 10 programmer connected.



On Windows XP/Vista/7 without WCID support installed.

- Download [Zadig](#).
- Connect the UFPI interface.
- Click Install WCID Driver.

The installation of the driver using Zadig is shown below.

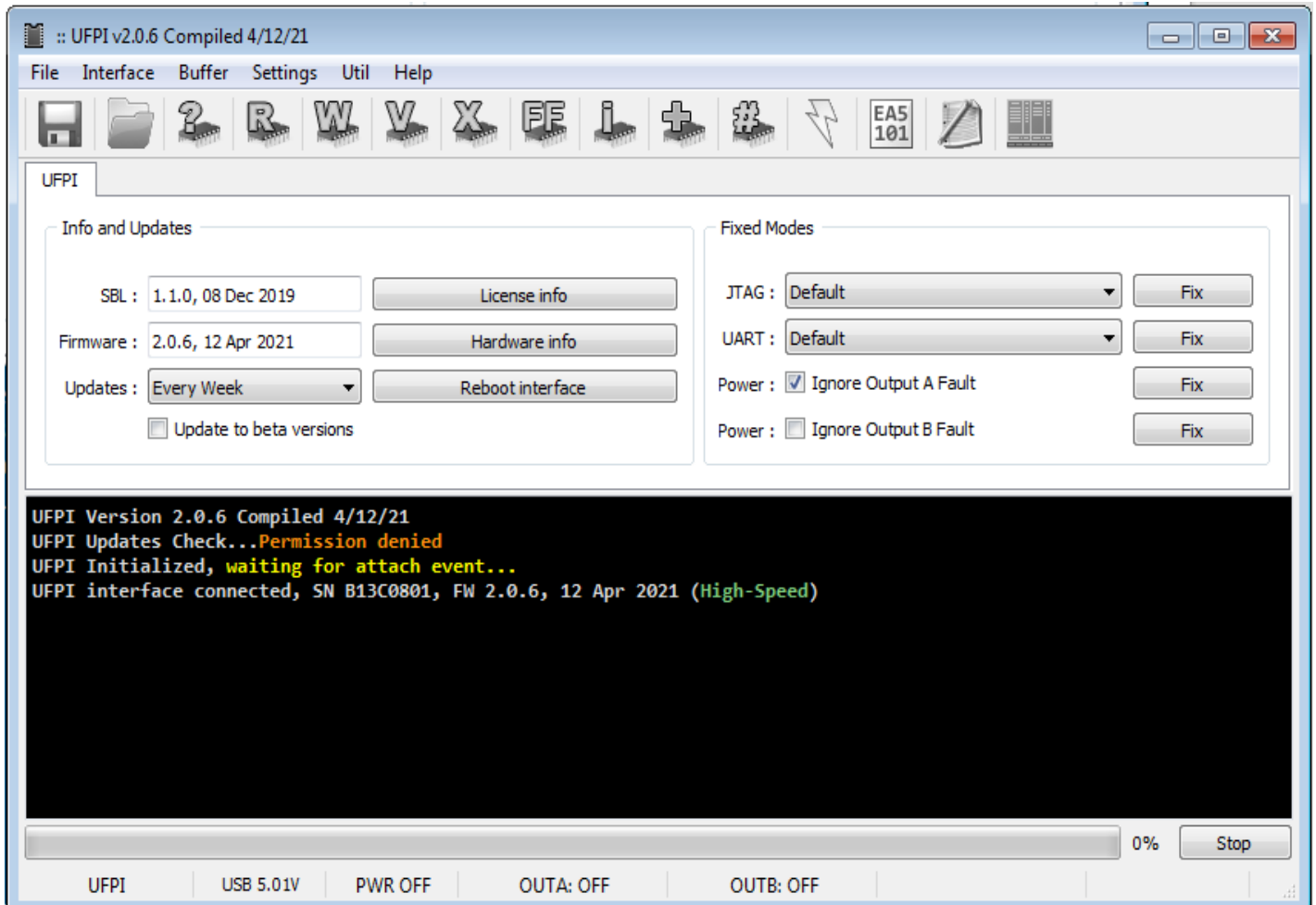


Linux.

To install on Linux, start the terminal with the command `sudo ./ufpi.run` and start the menu "Interface - Install."

2 Description of tabs and menus.

2.1 Main screen (tab).



USB voltage.

USB 5.01V

Programmers power supply

Installed voltage.

PWR OFF

Show installed voltage on socket.

PWR OFF

OUTA: OFF

Socket Power Outputs Voltages.

Voltage and current readings on the respective power supply channels of the box.

Time to end of current operation.

25 sec

Estimated time to the end of the operation.

Current operation speed.

121.3 KiB/sec

Show current operation speed

Disabling Socket Power Protection.

“Ignore Output A Fault” and “Ignore Output B Fault”

Power : Ignore Output A Fault

Power : Ignore Output B Fault

To set the mode, you need to tick the box corresponding to the channel and press the "Fix" button.

Emulation Modes.

JTAG :

UART :

It is possible to emulate CMSIS DAP. To do this, select the appropriate item in the JTAG menu and press the "Fix" button. *When a JTAG socket is inserted, a CMSIS DAP device appears in Device Manager. In this case, the program will no longer determine the programmer. To disable emulation in this case, it is enough to connect the programmer without a socket.*

Shedule update.

Updates :

Choose updates schedule.

Update mode.

Update to beta versions

Highly discouraged for untrained users. Anyone who needs to work, not experiment, is not encouraged to tick the box. This box is empty by default.

BOX and licences information.

List of activated licenses and other settings.

Additional information includes:

UFPI Reading Hardware Info...OK

UFPI PCB Revision A (Pro)

UFPI USB Power Switch SP2526A-2E

UFPI Voltage Translators NVT2008

UFPI 3V3 Calibration Value 3313 mV

2.2 TOOLBAR.



“Open File for reading” Button



“Open File for writing” Button



“Read ID” Button



Reading ID and chip initialization button. Usually, this button always starts working with the chip, receiving information about it, as well as [setting](#) the parameters of the program and programmer.

“Read flash” Button



Chip Memory Read Button. Read to file or buffer, depending on your configuration.

“Write flash” Button



Writes the chip from a file for writing or from a buffer, depending on the configuration.

“Verify flash” Button



Compares the chip contents with the write file or buffer, depending on the configuration.

“Erase flash” Button



“Blank check” button



«IC info” button.



«Add/Edit IC» button.

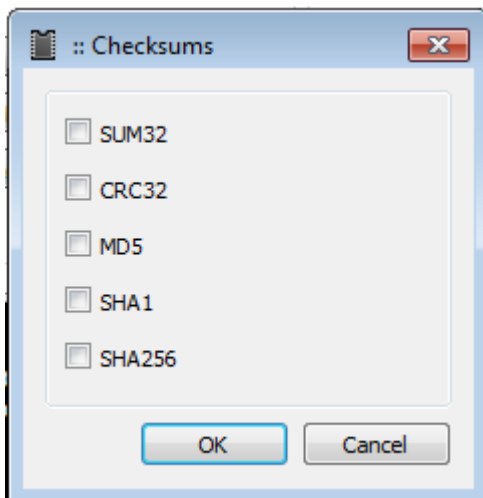


Chip Parameter Configuration. Described in [Appendix](#).

«Calk Checksum» button.



Checksum counting options during reading/recording/verification. The checksum is logged.



«Switch Socket Power Mode» button.



Enable/disable power supply of socket and power settings. Description in the relevant [chapter](#).

«Edit Buffer» button



Built-in HEX buffer editor. Description in the relevant [chapter](#).

«Execute Script» button



Start/stop script. Description in the relevant [chapter](#).

«File Manager» button



File Manager. Mount partitions, file systems, and work with their contents. Description in the relevant [chapter](#).

2.3 MAIN MENU

File Interface Buffer Settings Util Help

IN ORDER:

File.

Open File for Reading	Ctrl+S
Open File for Writing	Ctrl+O
Edit File for Reading	Alt+R
Edit File for Writing	Alt+W
Open Device	Ctrl+D
Close Device	Alt+D
Open Package	Ctrl+K
Close Package	Alt+K
Exit	Ctrl+Q

Open File for Read/Write - Duplicates buttons on the toolbar.

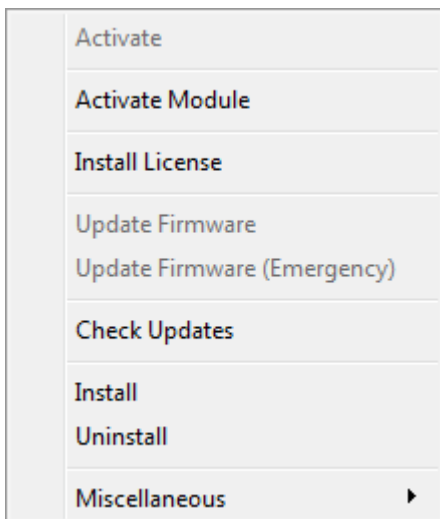
Edit Read/Write File - Opens the selected file in the embedded HEX editor without loading it into the clipboard.

Open/Close Device - Work with .UDEV files. Description in the relevant [chapter](#).

Open/Close Package - Works with Package files. Description in the relevant [chapter](#).

Exit – close the program.

Interface.



Install License - When you purchase a module license, you must select the file you received by mail to activate it from this menu.

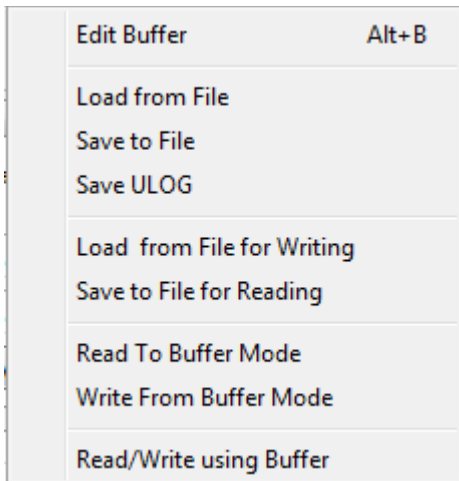
Firmware update (Emergency) - Very frequent question, [procedure description](#).

Install/Uninstall - Hint or install/uninstall the software on the OS.

Miscellaneous - Contains a [UFPI Self-Test](#).



Buffer.



Edit Buffer - Duplicates the button on the toolbar. Description in the relevant [chapter](#).

Load/Save to File and from File - Accordingly opens the file selection dialog and loads from file to buffer, or saves the contents of the buffer to file.

Save ULOG - Saves the result of writing [LOGGER](#) to a buffer in ULOG format.

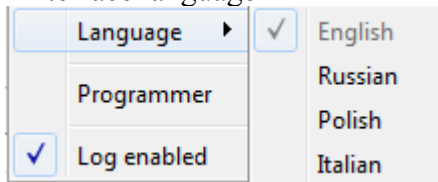
Download/Save To/From Read/Write File - Matches the name of the item in the application of files specified in the Read To and Write From fields on the module tab.

Read/write to/from buffer mode - Read/write flash to buffer mode. At the same time, the fields for the corresponding item "Read to" and "Write from" become inactive.

Read/Write to Buffer - Activates/deactivates both previous items at once.

Settings.

Interface language



[Program Settings](#) - Describes the settings.

[Log](#) enabled - tick the box to save logs as text in the Logs folder in the program folder.

Util.



Dump Analysis - [NAND](#) dump preparation tool for recording. Contains adjustment settings. Contains [ECC](#)

adjustment settings, [BB tables](#), sections, and more. Example in [attachement](#).

Package Editor - Open the Package Editor form. Description in the relevant [chapter](#).

Conversion:

S-Records - format used in MCU when working through BDM

ULOG - (internal LOGGER UFPI format) in Sigrok .SR (format PulseView) converts the logger-recorded stream into a format for viewing in Pulseview, respectively, from the file or buffer where it was captured.

Encryption - Script encryption options. Description in the relevant [chapter](#).

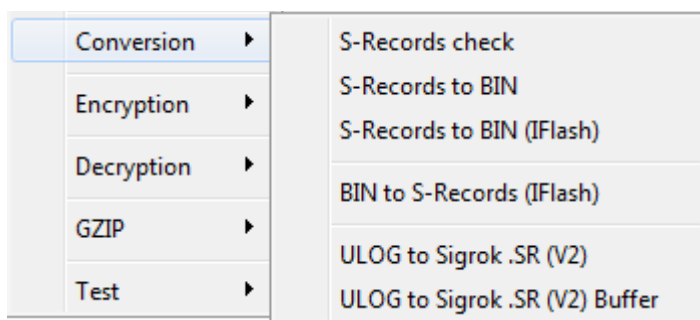


Decryption - Decryption of .nttc, .userc dumps.



[GZIP](#) - Packing/unpacking of the gzip file.

Test - Generates a test file individually for the connected chip.

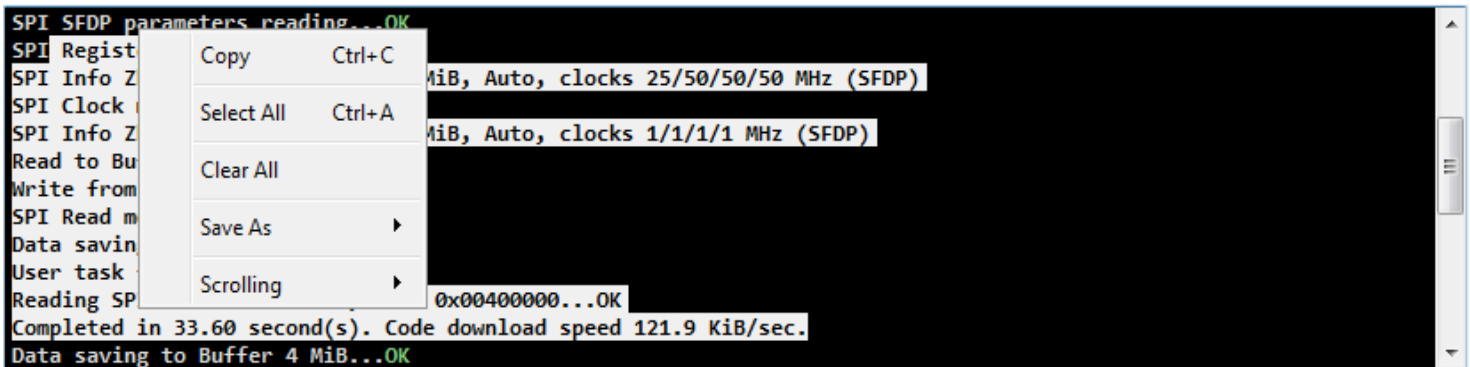


Help.

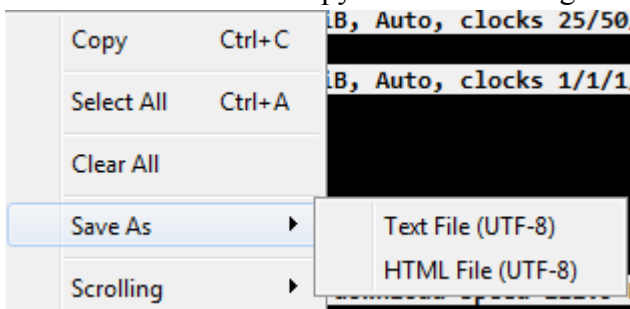
Program information and help file.

2.4 LOG.

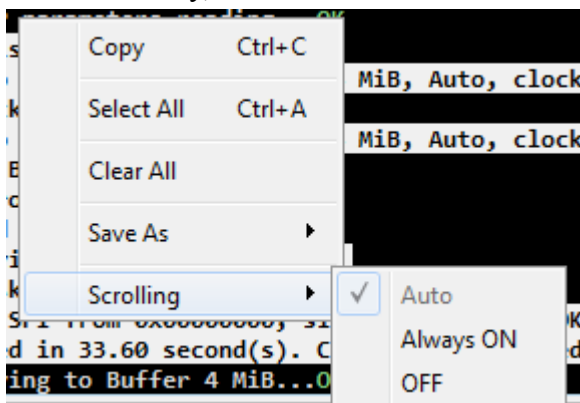
The log displays information about all transactions that occur. You can select and copy text from the log to the clipboard.



You can select and copy text from the log to the clipboard.



Alternatively, select Auto-Scroll mode



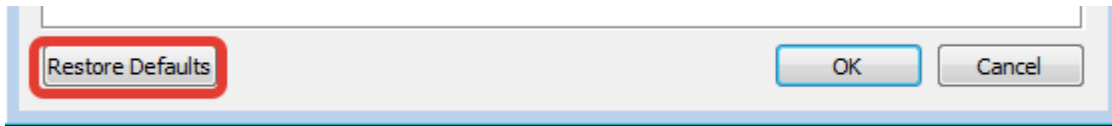
The default is Auto Select. Auto-scroll works in this mode if the scroll bar is lowered to the lowest position and there are no other selected fragments in the log.

The “Always On” and “Off” modes match the names.

3 Program SETTINGS.

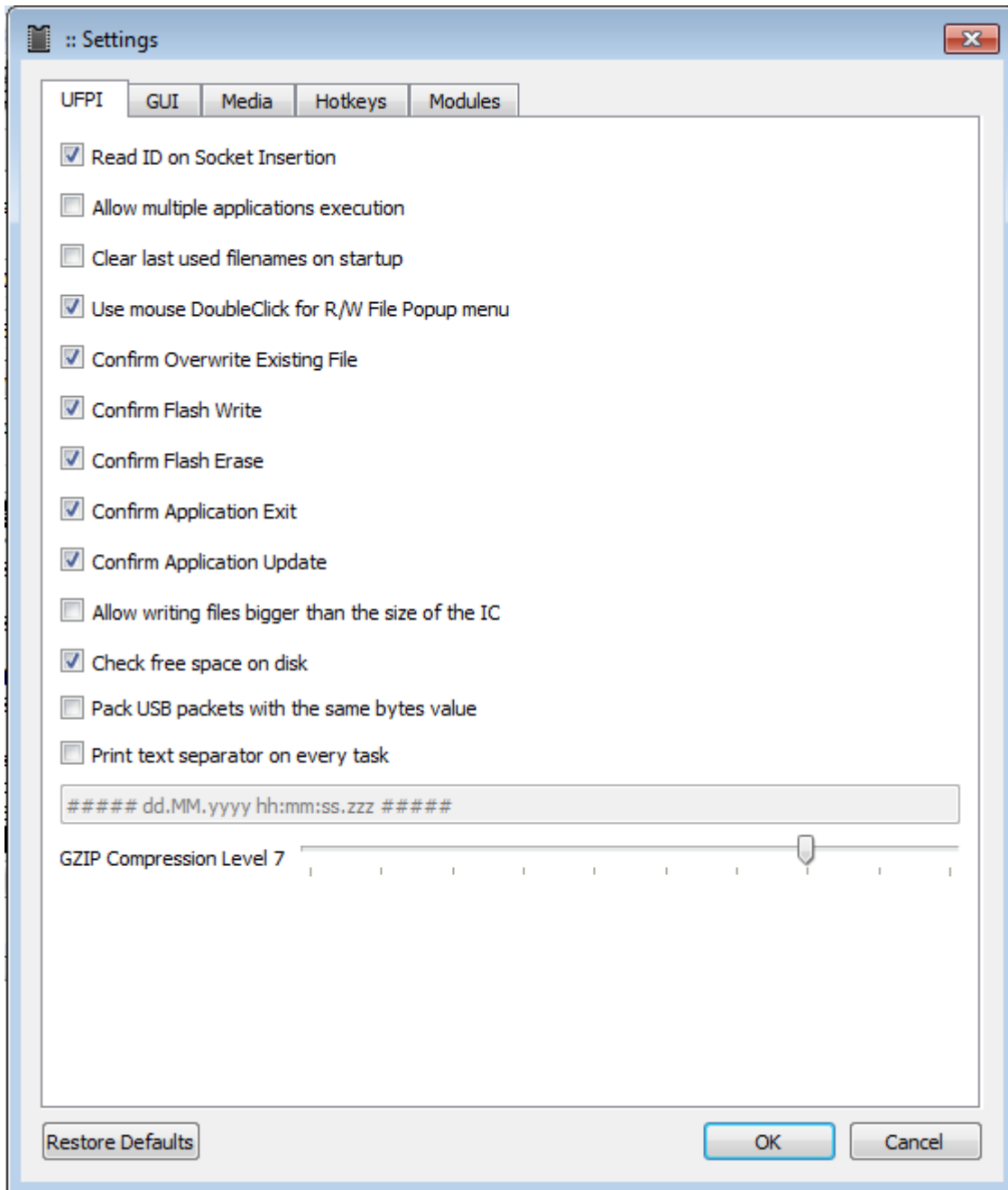
All settings are saved to the folder next to the program in the ufpi.ini file

To reset the settings, you can either delete the file and then automatically create it with the default settings the next time you start, or there is a “Restore Defaults” button at the bottom left of the Settings window that will set the default flags on the current tab.



3.1 UFPI tab.

The [Settings](#) in Programmer menu opens the main settings window on the UFPI tab.



By items:

Read ID on Socket Insertion.

When the tick box is set ON, immediately when the socket or socket box is connected, the chip ID reading command is executed. If not installed, only the "Read ID" button will read the ID. (Default is set ON)

Allow multiple application execution.

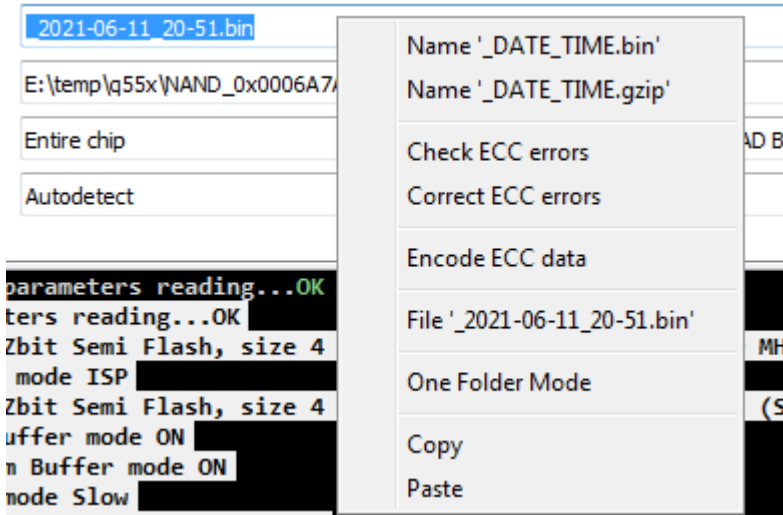
It can be useful if you need to run a script in parallel that does not require a box connection, and the program is already running. But this does not allow programmer to work with the second copy of the running program, even if two programmers are physically connected. (not set by default)

Clear last used filenames on startup.

When you start the program, clears the «Read To» and «Write From» fields. Then, when you select a read/write file, the UFPI program location folder opens. (not set by default)

Use mouse **DoubleClick** for R/W File Popup menu.

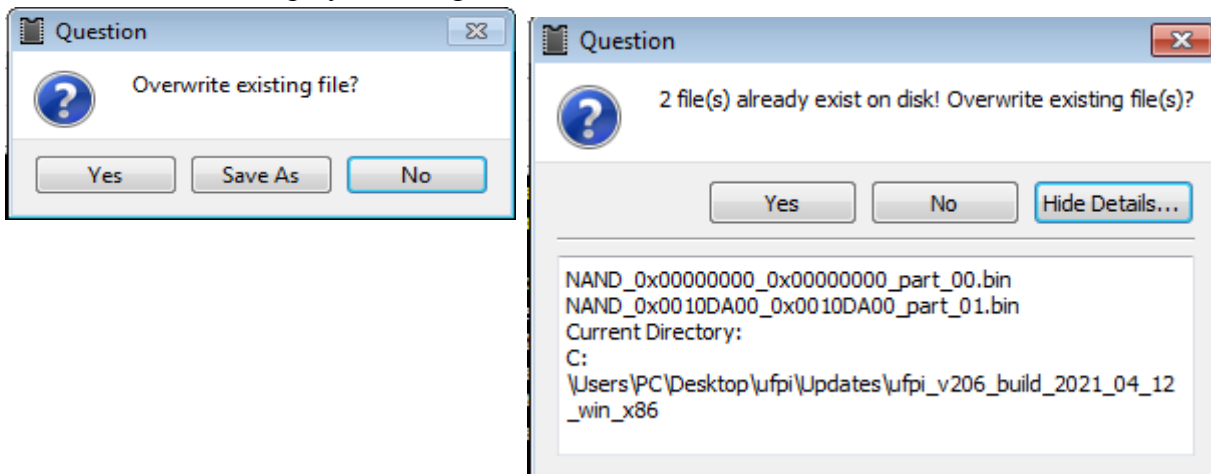
By double clicking in the field "Read to" and "Write from" menu allows you to set autaname, check and correct ECC, select files from history and other... (is set ON by default)



Confirm Overwrite Existing File.

Clearing the check box avoids prompting you to overwrite an existing file, such as a read-to-file operation. (Default is set ON)

When activated, displays a dialog.



Confirm Flash Write.

If the check box is cleared, the program does not ask for confirmation to write to the flash. (Default is set ON)

Confirm Flash erase.

If the check box is cleared, the program does not ask for confirmation to erase the flash. (Default is set ON)

Confirm Application Exit.

If the check box is cleared, the program does not ask for confirmation to exit the application. (Default is set ON)

Confirm Application Update.

If the check box is cleared, the program will be updated without confirmation. (Default is set ON)

Allow write files, bigger than the size of the IC.

With the check box cleared when trying to write a dump larger than the flash size, the operation will be aborted with a description of the cause in the log. If this check box is selected, dump will be recorded, but the size of the recorded size will be highlighted in the [warning color](#) in the log. (not set by default)

Check free space on disk.

If the check box is selected, the free space on the target disk will be evaluated before the read operation and if it is not enough, a warning will be issued. (not set by default)

Print text separator on every task.

When the tick box is set ON, prints the divider between operations specified below in the window. Example with date and time shown (not set by default)

Print text separator on every task

```
##### dd.MM.yyyy hh:mm:ss.zzz #####
```

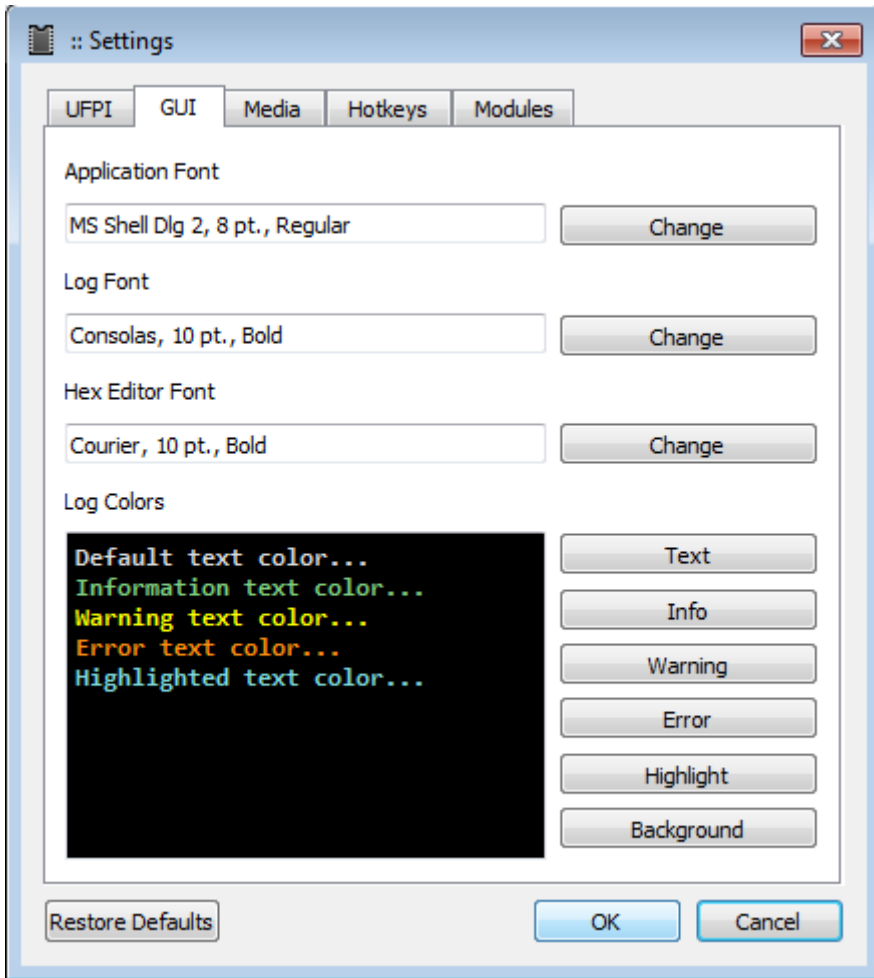
GZIP Compression Level.

Set the compression level for the archive to be created during the gzip read operation. (Default is set 7)

3.2 GUI tab.

The settings allow you select interface [colors and fonts](#).

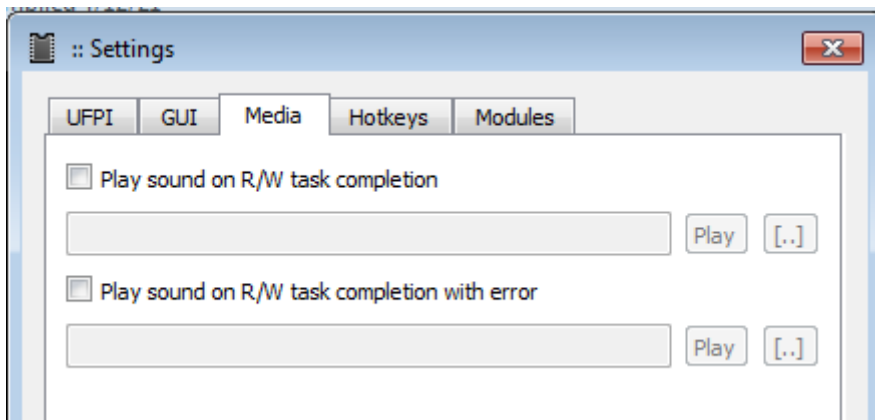
Opposite each item there is a corresponding button that activates the font or color selection dialog. The picture shows a window with default settings.



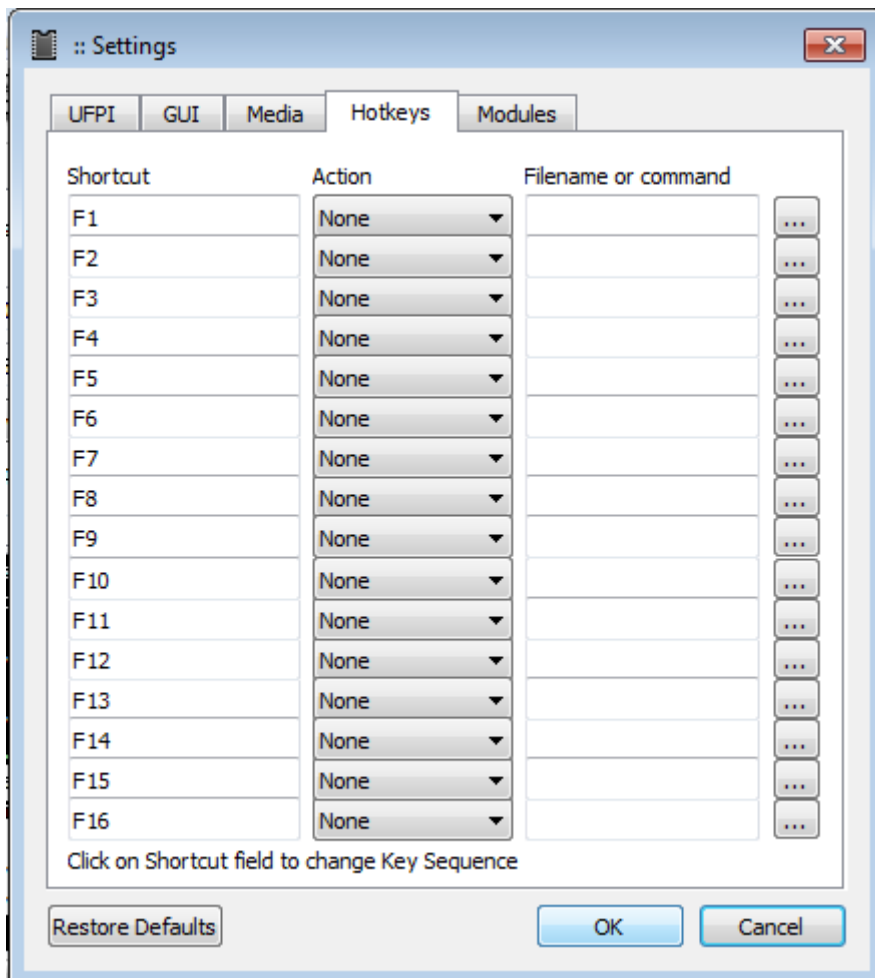
3.3 MEDIA.

Use this tab to set and check the selected sound effects for “task is complete”.

For example, for Windows 7, you can select C :\Windows\Media\Savanna\Windows Ding.wav.



3.4 BUTTONS.

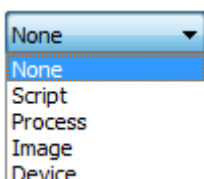


This tab allows you to set up to 16 hot keys.

Click the box in the «Shortcut» column.

You can select any button combination for an action.

In the «Action» column.



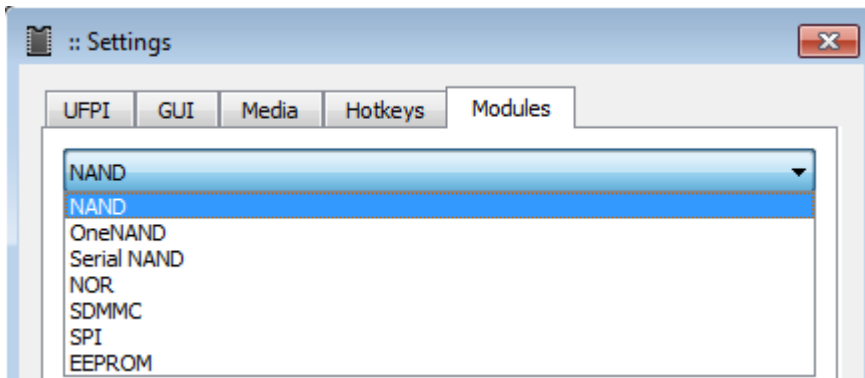
Select the type of action: script, process or image (opens in the built-in viewer, .jpg, .png, .gif, .bmp formats are allowed).

In the «FileName or Command» column.

You select a startup object (script, program, or image), or you can fit a command with parameters.

3.5 MODULES.

The tab has settings for each module, which are moved through the combo box.



3.5.1 NAND.

NAND

Show first page dump on ID reading. Du bytes (0=page size)

Show full IC info on ID reading

Show errors for blocks

Show corrected ECC errors

Do not stop verification on errors

Do not use RB signal. Read data access delay us

Show first page dump on ID reading.

If the checkbox is set ON, and the chips ID is successfully read, the contents of the first page will be displayed. Example in the picture below. (Active by default)

```

NAND Socket x8/x16, 4xCE, 3.30V
NAND ID ECD584725042 (ECD584725042ECD5), CE1 (*)
NAND Info Samsung K9GAG08U0E, 2.03 GiB, page 8192+436, x8, MLC ECC-24/1024, 3.30V, 30ns, 1xCE
NAND ECC correction OFF
NAND BAD Blocks Table OFF (Not used) (Block #0)
BAD blocks detection OFF (BAD Blocks Ignored)
BAD blocks management OFF
NAND Dump, 256 byte(s)
000000 40 55 52 5F 55 54 4C 00 01 01 02 01 FF FF FF | PSK_STL.....yyyy
000010 FE FF FF FF 00 08 00 12 00 00 01 00 01 00 02 00 | pyyy.....
000020 01 00 10 00 00 00 04 00 01 00 00 00 01 00 00 00 | .....
000030 02 00 D4 00 04 00 01 01 C9 00 0A 00 07 00 C9 00 | ..ô.....É.....É.
000040 00 48 06 00 80 64 00 00 01 00 00 02 09 00 00 00 | .H...d.....
000050 02 00 03 00 04 00 05 00 FF FF FF FF FF FF FF FF | .....yyyyyyyy
000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
000070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
000080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
000090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
0000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
0000B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
0000C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
0000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
0000E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | yyyyyyyyyyyyyyyy
0000F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....

```

SIZE.

Set the size of the first page when reading ID. The default value is 0x0 and the page size is based on the actual flash page size. NAND pages are large enough and it is enough 0x100 - 0x200 size for evaluation.

Show full chip information on ID reading.

3.5.2 OneNAND.

OneNAND

Show first page dump on ID reading

Show full IC info on ID reading

Show errors for blocks

Do not stop verification on errors

Show first page dump on ID reading.

If this check box is selected ON, the contents of the first page will be displayed after ID is successfully read. An example in the picture below. (not set by default)

```

OneNAND ID 00EC0035 (1xCE) OTP
OneNAND Info Samsung KFG1G16U2M, 128 MiB, page 2048+64
BAD blocks detection OFF (BAD Blocks Ignored)
BAD blocks management OFF

OneNAND Dump, 256 byte(s)
00000000 00 68 80 40 00 B8 08 40 FF FD 01 3C FF FF 21 34 | .h.@...@ÿÿ.<ÿÿ!4
00000010 24 40 01 01 00 B8 88 40 00 48 80 40 40 10 08 3C | $@...@.H.@@.<
00000020 00 60 88 40 00 BF 08 3C 00 00 09 24 10 60 09 AD | .`.@.ç.<...$.`.-
00000030 00 BF 08 3C 00 00 09 24 14 60 09 AD 00 00 00 00 | .ç.<...$.`.....
00000040 17 00 11 04 00 00 00 00 63 00 F0 0F 00 00 00 00 | .....c.ð.....
00000050 A8 01 F0 0F 00 00 00 00 60 83 13 3C 00 00 73 26 | ".ð.....`..<...s&
00000060 02 00 04 24 00 0C 05 24 A8 00 F0 0F 00 00 00 00 | ...$....$".ð.....
00000070 00 06 04 24 21 28 60 02 00 00 06 24 00 08 07 24 | ...$((`.....$....$
00000080 6D 00 F0 0F 00 00 00 00 00 82 08 3C 00 00 1D 25 | m.ð.....<...%
00000090 60 83 04 3C 00 00 84 24 08 00 80 00 00 00 00 00 | `..<...$......
000000A0 02 00 08 24 00 80 88 40 00 80 08 40 00 80 09 3C | ...$....@...@...<
000000B0 FF FF 29 35 24 40 09 01 04 36 09 3C 25 40 09 01 | ÿÿ)5$@...6.<%@..
000000C0 00 80 88 40 00 60 08 40 FC FF 09 3C FF FF 29 35 | ...@.`.@üÿ.<ÿÿ)5
000000D0 24 40 09 01 00 00 09 24 25 40 09 01 00 60 88 40 | $@...$.%@...`.@
000000E0 08 00 E0 03 00 00 00 00 00 00 00 00 00 00 00 00 | ..à.....
000000F0 22 BF 0B 3C 18 2C 6B 35 00 00 69 85 01 00 29 31 | "ç.<.,k5..i...)1

```

Show full IC info on ID reading.

Show in the Log more detailed information. The example in the picture. Compared with the previous one. (Not set by default).

```

OneNAND ID 00EC0035 (1xCE) OTP
Manufacturer Samsung
Model name KFG1G16U2M (Single)
Bus and voltage x16 (3.30V)
RAM Test OK (0xAAAA5555)
Ctrl. Status 0x0020 (OTP)
Page size 0x800 (2048)
Spare size 0x40 (64)
Pages in Block 0x40 (64)
Block data size 0x20000 (131072) 128 KiB
Block raw size 0x21000 (135168) 132 KiB
Blocks count 0x400 (1024)
Bits per Cell 1 (SLC)
ECC Requirement 1/512 (Bits/CW)
Dies per Chip 1
Chip (CE) count 1
Total data size 0x8000000 (128 MiB)
Total raw size 0x8400000 (132 MiB)
BAD blocks detection OFF (BAD Blocks Ignored)

```

Show errors for blocks.

If the check box is cleared, the log will show the number of errors (verification or [ECC](#) correction errors), if any, on each page.

Do not stop verification on errors.

When This check box is set ON, it allows you to complete verify to the end if there are errors and calculate the number of errors.

Do not use RB signal. Read data access delay.

This flag allows you not to control the NAND readiness by RB signal and set the delay manually. This option is not needed for day-to-day work with normal flush in pads. It can be useful for wiring when the location of this signal is unknown.

3.5.3 Serial NAND.

Serial NAND

- Show first page dump on ID reading
- Show full IC info on ID reading
- Show errors for blocks
- Do not stop verification on errors

Show first page dump on ID reading.

If this check box is set ON, the contents of the first page will be displayed if the flash detect is successful.
(not set by default)

Show full chip information on ID reading.

Show in Log more detailed information.

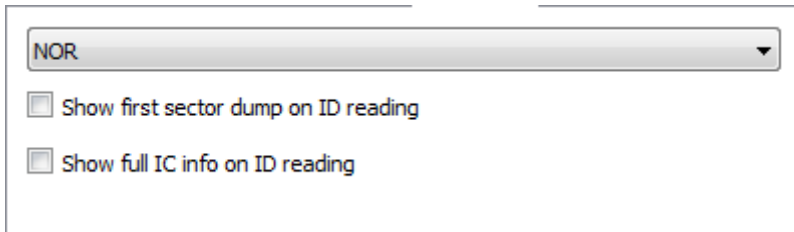
Show errors for blocks.

If the check box is cleared, the log will show the number of errors (verification or [ECC](#) correction errors), if any, on each page.

Do not stop verification on errors.

When This check box is selected- it allows you to complete verify to the end if there are errors and calculate the number of errors.

3.5.4 NOR.



NOR

Show first sector dump on ID reading

Show full IC info on ID reading

Show first sector dump on ID reading.

When the check box is selected ON, the contents of the first sector are displayed when the flash detect is successful. (not set by default)

Show full IC info on ID reading.

Show in Log more detailed information. (not set by default)

3.5.5 SDMMC.

SDMMC

- Show first page dump on ID reading
- Show full IC info on ID reading
- Set access to USER partition on ID reading
- Do not stop verification on errors
- Exclude RPMB from Full Backup

Read retry on error

Show first page dump on ID reading.

If the check box is selected ON, and the ID is read successfully, the contents of the first page will be displayed. (not set by default)

Show full IC info on ID reading.

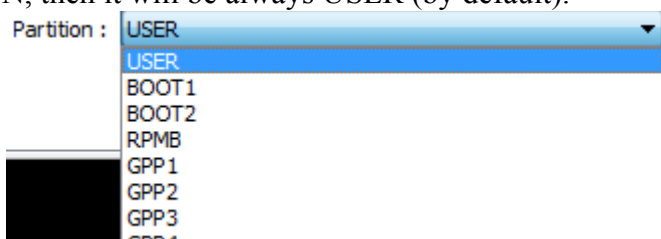
Logging detailed information about the flash. (Not installed by default)

```

eMMC ID 110100303034474530001CA7F8DD3361, DATA-DAT7
eMMC Info Toshiba '004GE0', size 3.69 GiB, SN 1CA7F8DD, Mar 2016
eMMC Mode 8-Bit, Transfer state, TI 1, Drv 0, Clock 52MHz
eMMC detailed information:
OCR 0xC0FF8080 (3V3/1V8)
CID 110100303034474530001CA7F8DD3361 (Toshiba 004GE0, SN 0x1CA7F8DD (480770269), Mar 2016)
CSD D05E00320F5903FFFFFFFFE7924000E3
Device Revision [192] 0x07 (MMC v5.0, v5.01)
Device Type [196] 0x57 (SDR HS52, HS200, HS400)
Command Classes 0xF5 (Class 0, 2, 4, 5, 6, 7)
Transfer Speed 0x32 (26MHz)
Device Pre-EOL Info 0x01 (Normal)
Device Life Time Type A 0x01 (0-10% used)
Device Life Time Type B 0x00 (Not defined)
Reset Function [162] 0x01 (Permanently enabled)
Boot Config [179] 0x78 (Boot from USER, access to USER, ACK)
Boot Bus [177] 0x00 (1-Bit SDR, Reset to 1-Bit)
Partitioning support [160] 0x07 (Yes, ENH, EXT)
Partitioning completed [155] 0x00 (NO)
USER Partition, size 0x00EC000000 (3.69 GiB)
BOOT1 Partition, size 0x0000200000 (2 MiB)
BOOT2 Partition, size 0x0000200000 (2 MiB)
RPMB Partition, size 0x0000080000 (512 KiB) Authentication Key not programmed
  
```

Set access to USER partition on ID reading.

If this check box is cleared, reading the ID in the "Partition": previously selected partition will be saved (USER/BOOT1/BOOT2/GPP *). Also, when the check box is selected ON, in "Show first page dump on ID reading," the dump of the first page of the selected partition will be displayed in the log (if the check box is set ON, then it will be always USER (by default)).



Do not stop verification on errors.

If the check box is cleared, in case of errors, the verification process will be completed the end and displays the number of errors in the log. IN normal conditions EMMC should not be read with errors.

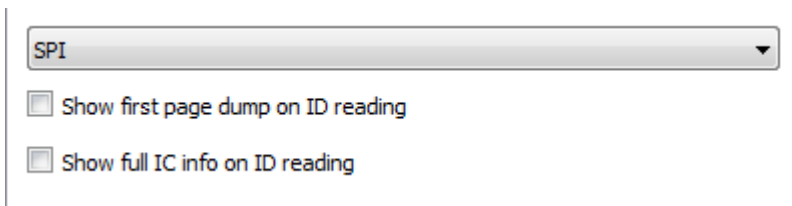
Read retry on error.

Number of read retries when eMMC fails. May be required when [reading faulty chips](#). *To do this, you need to disable in "Miscellaneous" "Asynchronous" option and set in Read Mode "Ignore Read Errors" and "Read Mode :Auto-Select"*

Exclude RPMB from the Full Backup.

Excludes the RPMB section from the «Full Backup» operation. *This partition usually does not contain cloning-free data and its presence in the full backup can be misleading, and an attempt to record can lead to chip spoilage .*

3.5.6 SPI.



The screenshot shows a software interface for SPI settings. At the top, there is a dropdown menu with 'SPI' selected. Below the dropdown are two checkboxes, both of which are currently unchecked. The first checkbox is labeled 'Show first page dump on ID reading' and the second is labeled 'Show full IC info on ID reading'.

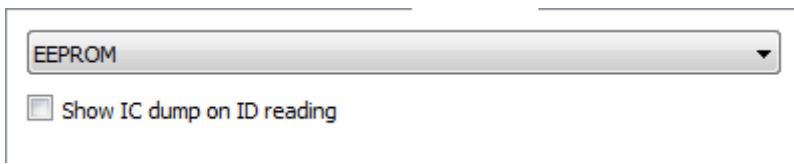
Show first page dump on ID reading.

If the check box is selected ON, and ID is read successfully, the contents of the first page will be displayed. (not set by default)

Show full IC info on ID reading.

Show in Log more detailed information. (not set by default)

3.5.7 EEPROM.



Show IC dump on ID reading.

If the check box is selected ON, and ID is read successfully, the contents of the first page will be displayed.
(not set by default)

4 MODULES.

MODULES list:

[1-Wire](#)

[EEPROM I2C](#)

[EEPROM SPI](#)

[EEPROM Microwire \(3-Wire\)](#)

[SPI Flash](#)

[UART](#)

[SD/eMMC](#)

[NOR](#)

[NAND](#)

[Serial NAND](#)

[OneNAND](#)

[BDM](#)

[JTAG](#)

[LOGGER](#)

[Mount File](#)

[Mount Chip](#)

4.1 LOGGER.

SOCKET.

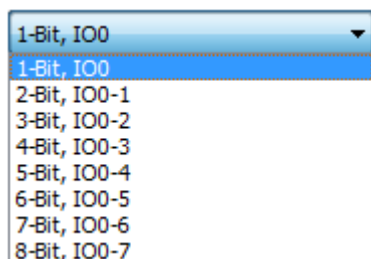
Works with socket [2045](#).

Features.

The socket has power supply and recording LEDs. It may also have a Vref/VCC logic level voltage switch.

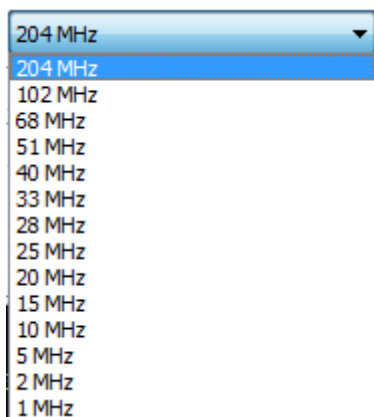
Logger records the status of connected lines (up to 8 lines) in RAM, or a file (the volume is limited only to free space on the medium where the save path is indicated) for analysis in the program ([PulseView by Sigrok](#)). You can connect to the i2c exchange bus and learn the exchange protocol.

Number of lines.



Select the number of lines to write. The number of lines affects the write speed and volume of the resulting file.

Sampling rate.

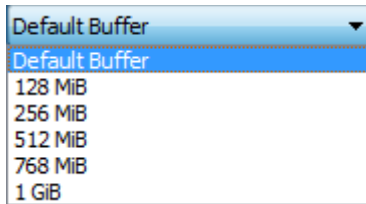


Frequency is set at least at $n * F * 5$, where n is the number of channels exposed.

F is the frequency of the block, or correspondingly the duration of the smallest pulses in the studied signal.

For example, for I2C (SDA + SCL lines) with a 200 kHz block frequency, you will need to set at least 2MHz.

Buffer size.



Select buffer limit. After filling the allotted volume, the recording stops automatically. When writing to disk, the option is ignored and does not restrict the file.

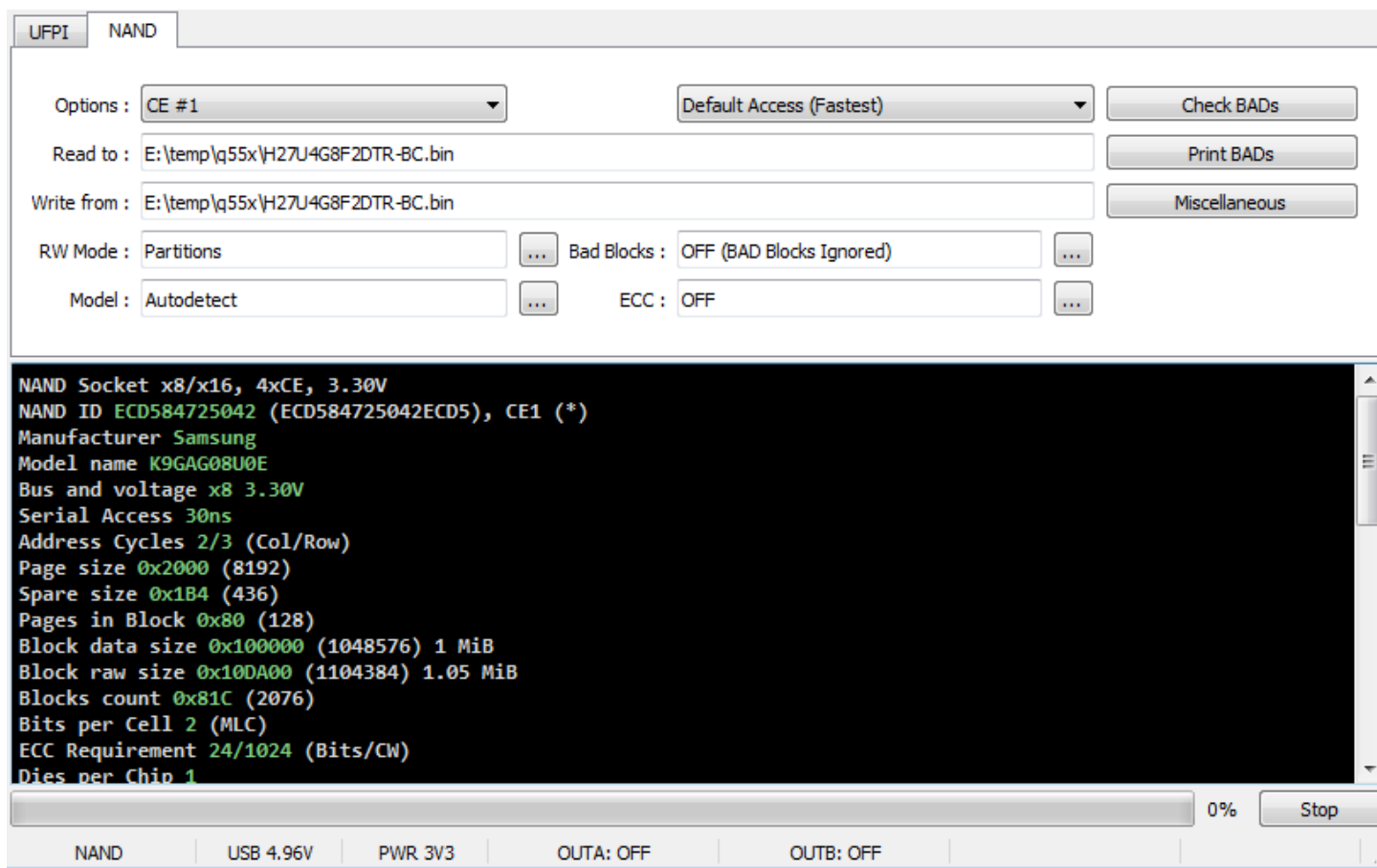
The most stable (when operating at high frequencies) capture in the computer RAM. This avoids possible operating system and hard drive delays.

Trigger.

Trigger (on any activity)

Sets the trigger (start of recording) to activity on any of the recorded channels.

4.2 NAND.



Sockets.

Works with sockets of various combinations. More specifically described in [chapter](#).

Features.

The NAND module allows you to work with a x8/x16 flash to 4xCE with a voltage from 1 to 5 volts. UFPI allows you to work with [BB](#), [ECC](#), ONFI, JESD, Partition table, etc.

Ability to work with scripts.

[Able to add manual flash configurations.](#)

By Items on the Main Window:

4.2.1 Options. Select CE (chip).



Select a chip if there are several of them in the flash. *In general, you do not need to touch it.*

Software automatically determines how many chips on ID reading.

```

NAND socket x8/x16, 4xCE, 3.30В
NAND ID ECD3552558 (ECD3552558ECD355), CE1 (*)
NAND ID ECD3552558 (ECD3552558ECD355), CE2
NAND ID ECD3552558 (ECD3552558ECD355), CE3
NAND ID ECD3552558 (ECD3552558ECD355), CE4
Производитель Samsung
Название модели K9MBG08U5M
Шина и напряжение x8 3.30В
Время доступа 50нс
Адресные циклы 2/3 (Col/Row)
Размер страницы 0x800 (2048)
Размер spare 0x40 (64)
Страниц в блоке 0x80 (128)
Размер данных блока 0x40000 (262144) 256 КиБ
Размер блока со spare 0x42000 (270336) 264 КиБ
Количество блоков 0x1000 (4096)
Бит в ячейке 2 (MLC)
Требования к ECC 4/512 (Bits/CW)
Кристаллов в чипе 1
Количество чипов (CE) 4
Общий размер данных 0x100000000 4 ГиБ
Общий размер со spare 0x108000000 4.13 ГиБ

```

An asterisk (*) opposite the CE1 indicates the selected chip.

As can be seen from LOG : four-chip flash has been determined. Each chip is 1Gb. Total data size as indicated is 4GB. So it will read in one file.

It should be noted that if the flash was determined automatically, and it is in the programmer base and in the log it is displayed in total size (in the log it is written "Number of chips (CE) 2"), then it will be read by one file. If the chip is unknown and the configuration is added manually, then each CE will need to be read individually by selecting the desired CE from the menu. So CE count cannot be set in configuration!

4.2.2 Access Cycle.

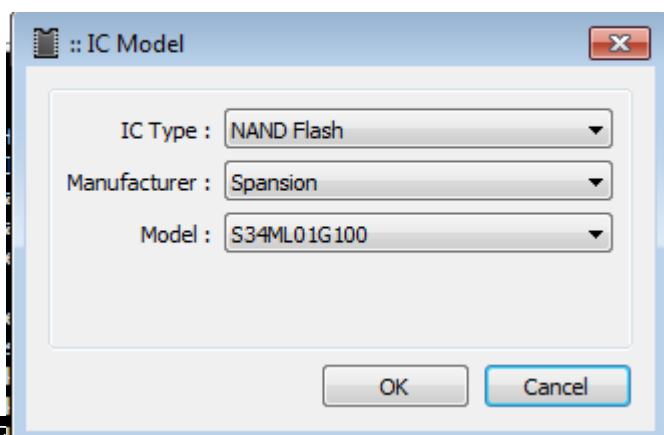
Allows you to set the duration of the RE signal, if you want to increase it when using non-standard connections, by wires. By default, it is used according to the internal database compiled from the IC documentation.

4.2.3 «Read to» и «Write from».

Standard procedures for almost every module. [DoubleClick displays AutoNames menu](#), [ECC](#) operations, file history, and more.

4.2.4 Model.

You do not need manually select the NAND model in UFPI, because the flash ID is automatically defined or

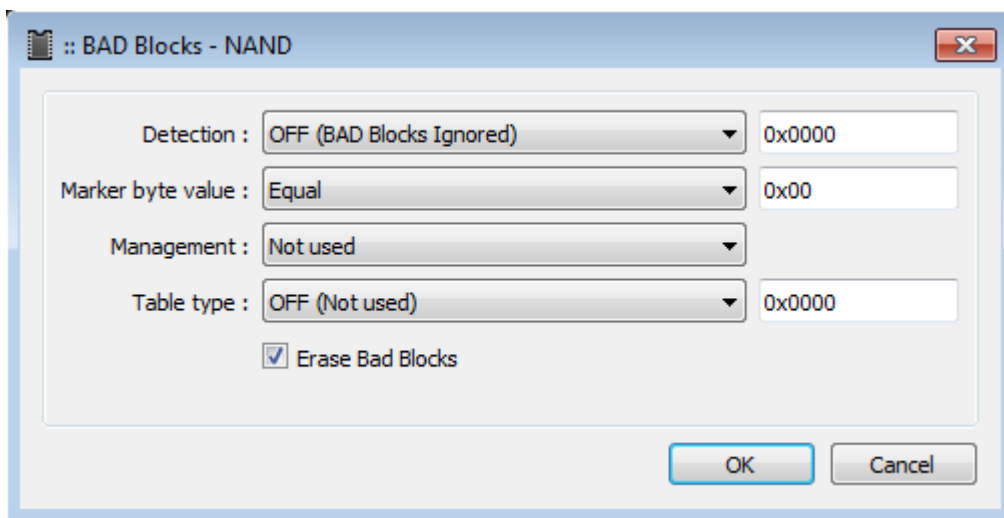


parameters from the database. But you can also choose manually.

4.2.5 RW Mode (access).

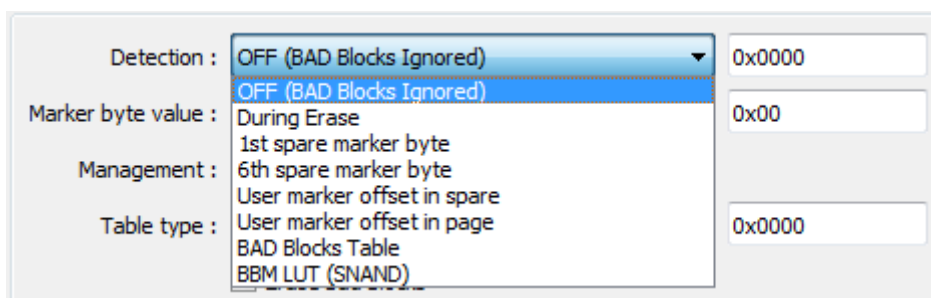
Described in the corresponding general [chapter](#). need to remember the addressing features for working with [NAND flash](#).

4.2.6 Bad Blocks (BB).



A menu that specifies the definition parameters and the BB bypass algorithm. You can load these options with a [UDEVI](#) file or set them manually. *After changing the settings, need to read the chip ID again.*

Detection



Parameter by which BB are detected.

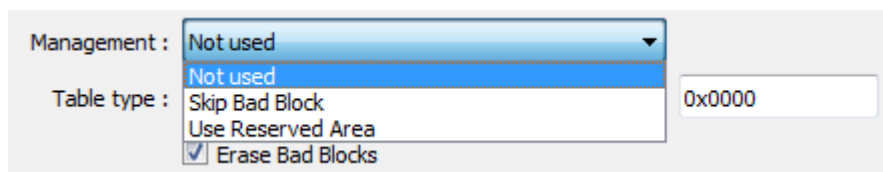
You can either turn off detection, or define them by a marker that can be specified arbitrarily (*Usually the first or sixth byte*), or by entries in the [bad block table](#) (some Samsung and LG are currently supported), or by internal BBM LUT, which uses for example Winbond SNAND.

Marker byte value.

Marker value. *Usually, such a marker is considered 0x00, but this can be any non-equal 0xFF.*

Management.

BB processing method selection.



Either not used, either skipping with continuing writing to the next (table type is not important here), either

in the most correct way, using the reserved area. *It is possible to use the reserve area if the [bad block table](#) was correctly determined during dump analysis. For a normal skip or write, "as it is" must be set to "Not in Use."*

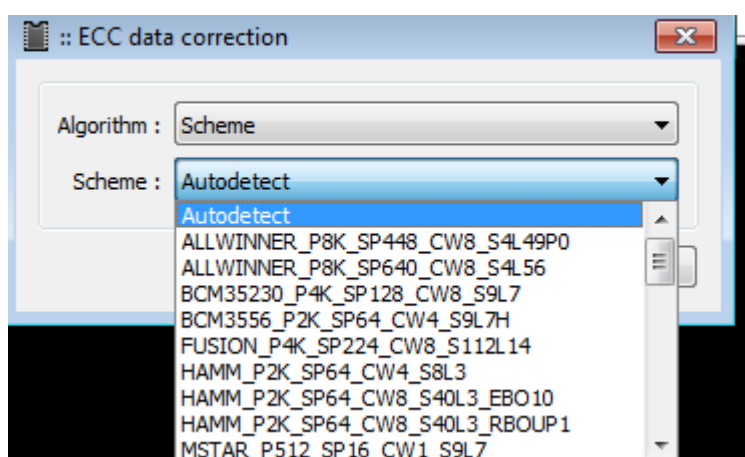
Type table.



Select the bad block table type.

For normal skipping or recording "as it is" must be turned off. The table type is specified when Using the reserved area. If we work with Samsung or LG dumps, then we put up "Auto-Determination" or the corresponding item.

4.2.7 ECC.



Select the ECC to use.

Selecting a scheme (used in a dump) or turning on the internal correction algorithm of the chip (called HWECC, this is a correction whose code is generated and processed by the chip itself).

There are a lot of different schemes, so auto-determination is added. ECC adjusted correction required for reading, writing when creating a new BBT, verification, analysis, and dump preparation.

4.2.8 **Bad block list management buttons.**

Check BADs (bad blocks).

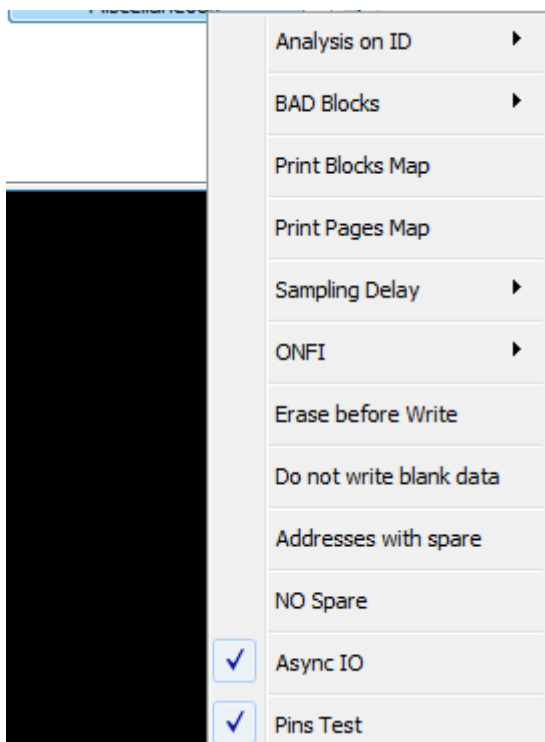
Scans chip for BB, depending on the Detection option selected.

Print BADs.

Displays information of the BB log in the current session.

4.2.9 **Miscellaneous /Additional features. (some of them s appear after ID reading)**

Additional features

**Analysis on ID reading.**

Select an analysis object. You can select dump analysis (file selected for recording) and chip analysis (chip content). After the analysis is complete, the program will check the geometry of the dump, ECC, the presence of BB, BBT, partition table and configure the programmer to record the corresponding parameters. [More](#).

BAD Blocks

Working with the [BB list](#). You can save (to the .ubad file), restore (from the .ubad file), or mark bad blocks by adding the necessary blocks, after a comma.

Print Blocks Map.

Shows in log a list of blocks with their working addressing. *Show the address including spare.*

Example:

```
Block 0x0000 (0) addr 0x00000000 (0x00000000)
Block 0x0001 (1) addr 0x00100000 (0x0010DA00)
Block 0x0002 (2) addr 0x00200000 (0x0021B400)
Block 0x0003 (3) addr 0x00300000 (0x00328E00)
Block 0x0004 (4) addr 0x00400000 (0x00436800)
Block 0x0005 (5) addr 0x00500000 (0x00544200)...
```

Print Pages MAP

Show in log list of pages with their working addressing. The address are including spare.

Example:

```
Block 0x0777 (1911) addr 0x77700000 (0x7DCB5600) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD BlockPage 0x0000 (0), block 0x0000 (0) at 0x00000000 (0x00000000)
Page 0x0001 (1), block 0x0000 (0) at 0x00002000 (0x000021B4)
Page 0x0002 (2), block 0x0000 (0) at 0x00004000 (0x00004368)
Page 0x0003 (3), block 0x0000 (0) at 0x00006000 (0x0000651C)
```

Page 0x0004 (4), block 0x0000 (0) at 0x00008000 (0x000086D0)
 Page 0x0005 (5), block 0x0000 (0) at 0x0000A000 (0x0000A884)...

Sampling delay

Delay settings can be used when ISP connecting used, or a DIY-made adapter to compensate for long lines.

ONFI.

Enabled/Disabled (enabled by default). [Some NANDs have a special ONFI memory region](#) in which its configuration is stored. UFPI can analyze these parameters and even if the chip is not in the programmer database, can apply, or supplement the parameters from these factory configurations.

CRC check (Enabled by default). Some chips contain an ONFI filled out of the standard, or have errors in it, or even an incorrect CRC for various reasons. In such cases, you can disable CRC checking or disable ONFI access.

Erase before Write

When the check box is selected ON, the erase command is automatically started before recording.

Do not write blank data.

When the check box is selected ON, blocks filled with 0xFF are skipped during recording. In this case, you can significantly save time. However, you must know that when working with NAND having HWECC, the record may be incorrect.

Adresses with Spare (disabled by default).

Show in log addresses with Spare. In this case, in the log, addresses with Spare will be highlighted in blue colour. And the task sign will have SADDR.

NO Spare. (disabled by default)

Check box sets ON to work with a dump that does not have Spare (Spare size = 0 bytes). *Such dumps have a specific purpose, for example, for working with jtag. Such dumps are not suitable for working with programmers.*

Async IO (enabled by default).

This programmer mode allows you to increase read/write speed under normal conditions due to the control of algorithms by the programmer processor itself.

Pin Test (enabled by default).

When reading chips ID, the correct installation and "shorts" of the lines between each other or on the VCC/GND are pre-checked. When defining a fault in the log, a message is displayed indicating the specific lines:

```
NAND ID Pins Test Failed!  
Pins short : D1 D2
```

Build configuration files.

Described in [appendix](#)

4.2.10 Application. Bad Blocks (BB).

Log of BB.

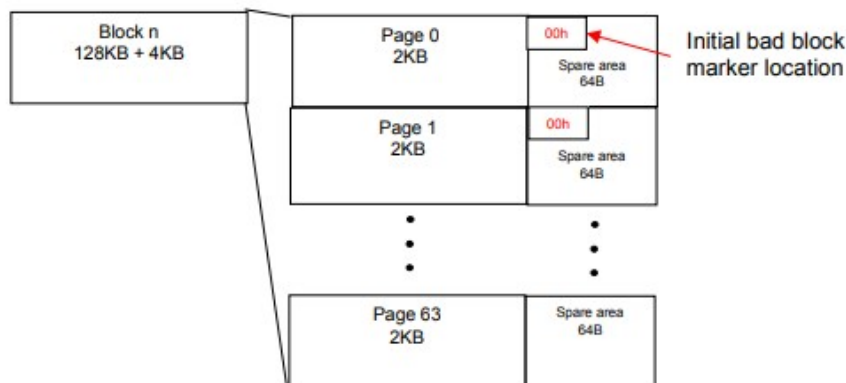
Work begins by reading the ID and displaying the NAND BB list

```
NAND socket x8/x16, 4xCE, 3.30B
NAND ID ECD584725042 (ECD584725042ECD5), CE1 (*)
Manufacturer Samsung
Model name K9GAG08U0E
Bus and voltage x8 3.30V
Serial Access 30ns
Address Cycles 2/3 (Col/Row)
Page size 0x2000 (8192)
Spare size 0x1B4 (436)
Pages in Block 0x80 (128)
Block data size 0x100000 (1048576) 1 MiB
Block raw size 0x10DA00 (1104384) 1.05 MiB
Blocks count 0x81C (2076)
Bits per Cell 2 (MLC)
ECC Requirement 24/1024 (Bits/CW)
Dies per Chip 1
Chip (CE) count 1
Total data size 0x81C00000 2.03 GiB
Total raw size 0x88A7D800 2.13 GiB
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42 (no ECC errors in Page #0)
NAND BAD Blocks Table Samsung RFS (Block #0)
BAD blocks detection (Spare bytes #0 == 0x00) //detection method "First byte marker in spare" is 0x00
BAD blocks management Use Reserved Area
BAD blocks request...OK
BAD blocks count: 9 //the following blocks are listed by this criterion
Block 0x0001 (1) addr 0x00100000 (0x0010DA00) BAD Block
Block 0x0003 (3) addr 0x00300000 (0x00328E00) BAD Block
Block 0x0005 (5) addr 0x00500000 (0x00544200) BAD Block
Block 0x0007 (7) addr 0x00700000 (0x0075F600) BAD Block
Block 0x0009 (9) addr 0x00900000 (0x0097AA00) BAD Block
Block 0x000B (11) addr 0x00B00000 (0x00B95E00) BAD Block
Block 0x025E (606) addr 0x25E00000 (0x27E40C00) BAD Block
Block 0x07E4 (2020) addr 0x7E400000 (0x84F82800) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
```

When reading the ID, a list of only those BBs that have been determined according to the put [options and markers](#) is described in LOG. There are no universal ways to correctly define a marker, although standards exist, but manufacturers do not always adhere to them. Most often, this marker corresponds to the first byte in spare, the first page of each block and is 0x00.

Have to know that block 0 is usually guaranteed by the chip manufacturer as reliable and cannot have a BB marker. If, for some reason, block 0 is marked as "Bad Block" in the log, then the marker is not selected

Figure 2-1: Macronix NAND flash Memory Array Structure



correctly.

If the number of detected BBs in the log reaches 256 pieces, then the marker is not selected correctly.

If the number of detected BB in the log exceeds 20-30 pieces, then such a flash is most likely not suitable for reliable operation. Since the spare area for replacing the BB is limited, although it all depends on the chip and for example in K9GAG08U0E Samsung D5500 has:

NAND RBA 0x07FB-0x0785 (2043-1925), 0x77 (119) Blocks

Physical BB.

In the production of NAND in the factories, blocks are blocked for recording that cannot provide the necessary reliability. Special-purpose blocks are also blocked, such as in SAMSUNG K9GAG08U0E 2073 the block is blocked on all chips. Typically, a blocked block is simply filled with 00 and write is protected, but also other filling may occur.

Logical BB.

Errors in NAND are almost always present and accumulate during work. Correction code is used to correct them.

When the number of errors in the page approaches the ECC limit, the contents of the block are transferred depending on the algorithm to a new place, an writes in the BB table, and a marker about its state is writes in the block with errors. Such blocks can be erased. recorded (if a check mark is set ON in the options), but need carefully checking their status before using.

Often, new chips (for example, K9GAG08U0E) comes from Aliexpress, having only 1-3 BB, after careful inspection, have dozens of worn-out blocks, which should be labeled as "Bad blocks." These chips are not new, just carefully erased by the seller with BB markers. Here is an example of verifying such chip after a week after recording:

NAND socket x8/x16, 4xCE, 3.30B

NAND ID ECD584725042 (ECD584725042ECD5), CE1 (*)

Manufacturer Samsung

Model name K9GAG08U0E

Bus and voltage x8 3.30V

Serial Access 30ns

Address Cycles 2/3 (Col/Row)

Page size 0x2000 (8192)

Spare size 0x1B4 (436)

Pages in Block 0x80 (128)

```

Block data size 0x100000 (1048576) 1 MiB
Block raw size 0x10DA00 (1104384) 1.05 MiB
Blocks count 0x81C (2076)
Bits per Cell 2 (MLC)
ECC Requirement 24/1024 (Bits/CW)
Dies per Chip 1
Chip (CE) count 1
Total data size 0x81C00000 2.03 GiB
Total raw size 0x88A7D800 2.13 GiB
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42 (no ECC errors in Page #0)
NAND BAD Blocks Table Samsung RFS (Block #0)
BAD blocks detection (Spare bytes #0 == 0x00)
BAD blocks management Use Reserved Area
BAD blocks request...OK
BAD blocks count: 2
Block 0x0714 (1812) addr 0x71400000 (0x77470800) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
NAND Dump, 256 bytes
00000000 46 53 52 5F 53 54 4C 00 01 01 02 01 FF FF FF FF | FSR_STL....ÿÿÿÿ
00000010 FE FF FF FF 00 08 00 12 00 00 01 00 01 00 02 00 | þÿÿÿ.....
00000020 01 00 10 00 00 00 04 00 01 00 00 00 01 00 00 00 | .....
00000030 02 00 D4 00 04 00 01 01 C9 00 0A 00 07 00 C9 00 | ..ô.....É.....É.
00000040 00 48 06 00 80 64 00 00 01 00 00 02 09 00 00 00 | .H...d.....
00000050 02 00 03 00 04 00 05 00 FF FF FF FF FF FF FF FF | .....ÿÿÿÿÿÿÿÿÿÿ
00000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
NAND Dump 'E:\temp\damp\Новая папка\SAMSUNG\D55xx\dados flash tv samsung un32d5500 k9gag08u0e - ic1302\
CLEAR\dados flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin' Analysis, size 0x88A7D800...
NAND Page 8192+436 (0x21B4), PIB 128, Block 0x10DA00 (1104384), 0x081C (2076) Blocks, IC geometry
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42
NAND BAD Blocks Table Samsung RFS, Block 0x07FD (2045)
NAND Partition Info Samsung RFS, Block 0x07FE (2046)
NAND LPCH Block 0x07FE (2046) at 0x86AE4C00
NAND LPCH TPCB 0x07FC (2044), Ages 1-1
NAND UPCH Block 0x07FD (2045) at 0x869D7200, 1 record(s) (Active)
NAND UPCH TPCB 0x07FC (2044), Ages 1-2, инфo 0x0000
NAND RBA 0x07FB-0x0785 (2043-1925), 0x77 (119) Blocks
NAND Partition #1, 'part-00-ID1B', Block 0x0 (0), count 0xD6 (214)
NAND Partition #1, addr 0x00000000 (0x00000000), size 0x0D600000 (0x0E163C00)
NAND Partition #2, 'part-01-ID1C', Block 0xD6 (214), count 0x100 (256)
NAND Partition #2, addr 0x0D600000 (0x0E163C00), size 0x10000000 (0x10DA0000)
NAND Partition #3, 'part-02-ID1D', Block 0x1D6 (470), count 0xD6 (214)
NAND Partition #3, addr 0x1D600000 (0x1EF03C00), size 0x0D600000 (0x0E163C00)
NAND Partition #4, 'part-03-ID1E', Block 0x2AC (684), count 0x410 (1040)
NAND Partition #4, addr 0x2AC00000 (0x2D067800), size 0x41000000 (0x4475A000)

```

```

NAND Partition #5, 'part-04-ID1F', Block 0x6BC (1724), count 0xC8 (200)
NAND Partition #5, addr 0x6BC00000 (0x717C1800), size 0x0C800000 (0x0D2A5000)
NAND Partition #6, 'UPCH', Block 0x7FD (2045), count 0x1 (1)
NAND Partition #6, addr 0x7FD00000 (0x869D7200), size 0x00100000 (0x0010DA00)
NAND Partition #7, 'LPCH', Block 0x7FE (2046), count 0x1 (1)
NAND Partition #7, addr 0x7FE00000 (0x86AE4C00), size 0x00100000 (0x0010DA00)
Loading IC BAD Blocks list...2 BAD Block(s)
Writing new BAD Blocks list...
NAND BAD Block 0x0714 (1812) replaced with 0x07FB (2043)
NAND BAD Block 0x0819 (2073) ignored
Encoding ECC in the patched pages...MSTAR_P8K_SP436_CW8_S12L42
BAD blocks management Use Reserved Area
*****2021.05.16 22:11:34:1134*****
Data source File 'E:\temp\damp\Новая папка\SAMSUNG\D55xx\dados flash tv samsung un32d5500 k9gag08u0e -
ic1302\CLEAR\dados flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin'...Normal
User task flags Async IO, ECC
Verifying NAND from 0x00000000, size 0x88A7D800...
Block 0x0002 (2) addr 0x00200000 (0x0021B400) 2 Bit error(s) in 1 page(s)!
Block 0x0003 (3) addr 0x00300000 (0x00328E00) 2 Bit error(s) in 1 page(s)!
Block 0x0004 (4) addr 0x00400000 (0x00436800) 3 Bit error(s) in 1 page(s)!
Block 0x0005 (5) addr 0x00500000 (0x00544200) 5 Bit error(s) in 4 page(s)!
Block 0x0006 (6) addr 0x00600000 (0x00651C00) ECC, 5 bit(s) corrected
Block 0x0007 (7) addr 0x00700000 (0x0075F600) 1 Bit error(s) in 1 page(s)!
Page 0x6C02 (27650), Block 0x00D8 (216) addr 0x0D804000 (0x0E383368) Uncorrectable ECC error!//unreliable
Block
Block 0x00D8 (216) addr 0x0D800000 (0x0E37F000) 4 Bit error(s) in 2 page(s)!
Block 0x00D9 (217) addr 0x0D900000 (0x0E48CA00) 4 Bit error(s) in 1 page(s)!
Block 0x00DA (218) addr 0x0DA00000 (0x0E59A400) 2 Bit error(s) in 1 page(s)!
Block 0x00DB (219) addr 0x0DB00000 (0x0E6A7E00) ECC, 1 bit(s) corrected
Block 0x00DB (219) addr 0x0DB00000 (0x0E6A7E00) 8 Bit error(s) in 5 page(s)!
Block 0x00DC (220) addr 0x0DC00000 (0x0E7B5800) 1 Bit error(s) in 1 page(s)!
Block 0x00DD (221) addr 0x0DD00000 (0x0E8C3200) 3 Bit error(s) in 1 page(s)!
Block 0x00DE (222) addr 0x0DE00000 (0x0E9D0C00) ECC, 3 bit(s) corrected
Page 0x6F90 (28560), Block 0x00DF (223) addr 0x0DF20000 (0x0EB00140) Uncorrectable ECC error!//unreliable
Block
Block 0x00DF (223) addr 0x0DF00000 (0x0EAD6000) 1 Bit error(s) in 1 page(s)!
Block 0x00E0 (224) addr 0x0E000000 (0x0EBEC000) 1 Bit error(s) in 1 page(s)!
Block 0x01D6 (470) addr 0x1D600000 (0x1EF03C00) 1 Bit error(s) in 1 page(s)!
Block 0x01D8 (472) addr 0x1D800000 (0x1F11F000) 16 Bit error(s) in 8 page(s)!
Block 0x01D9 (473) addr 0x1D900000 (0x1F22CA00) 10 Bit error(s) in 7 page(s)!
Page 0xED0C (60684), Block 0x01DA (474) addr 0x1DA18000 (0x1F353870) Uncorrectable ECC error!//unreliable
Block
Block 0x01DA (474) addr 0x1DA00000 (0x1F33A400) 16 Bit error(s) in 11 page(s)!
Page 0xED92 (60818), Block 0x01DB (475) addr 0x1DB24000 (0x1F46DCA8) Uncorrectable ECC error!//unreliable
Block
Page 0xEDAC (60844), Block 0x01DB (475) addr 0x1DB58000 (0x1F4A48F0) Uncorrectable ECC error!//unreliable
Block
Block 0x01DB (475) addr 0x1DB00000 (0x1F447E00) ECC, 1 bit(s) corrected
...
Block 0x0214 (532) addr 0x21400000 (0x23050800) ECC, 4 bit(s) corrected
Block 0x0215 (533) addr 0x21500000 (0x2315E200) ECC, 1 bit(s) corrected
Block 0x0215 (533) addr 0x21500000 (0x2315E200) 3 Bit error(s) in 1 page(s)!
Block 0x0216 (534) addr 0x21600000 (0x2326BC00) ECC, 2 bit(s) corrected
Block 0x0216 (534) addr 0x21600000 (0x2326BC00) 2 Bit error(s) in 1 page(s)!

```


Block 0x0217 (535) addr 0x21700000 (0x23379600) ECC, 4 bit(s) corrected
 Block 0x0218 (536) addr 0x21800000 (0x23487000) ECC, 9 bit(s) corrected
 Block 0x02AE (686) addr 0x2AE00000 (0x2D282C00) 3 Bit error(s) in 2 page(s)!
 Block 0x02AF (687) addr 0x2AF00000 (0x2D390600) 2 Bit error(s) in 1 page(s)!
 Block 0x02B0 (688) addr 0x2B000000 (0x2D49E000) 3 Bit error(s) in 1 page(s)!
 Page 0x15884 (88196), Block 0x02B1 (689) addr 0x2B108000 (0x2D5B40D0) Uncorrectable ECC error!//unreliable Block
 Page 0x1588E (88206), Block 0x02B1 (689) addr 0x2B11C000 (0x2D5C91D8) Uncorrectable ECC error!//unreliable Block
 Block 0x02B1 (689) addr 0x2B100000 (0x2D5ABA00) 11 Bit error(s) in 7 page(s)!
 Block 0x02B2 (690) addr 0x2B200000 (0x2D6B9400) 4 Bit error(s) in 2 page(s)!
 Block 0x02B4 (692) addr 0x2B400000 (0x2D8D4800) 3 Bit error(s) in 1 page(s)!
 Block 0x02B5 (693) addr 0x2B500000 (0x2D9E2200) 2 Bit error(s) in 1 page(s)!
 Block 0x02B6 (694) addr 0x2B600000 (0x2DAEFC00) 2 Bit error(s) in 1 page(s)!
 Page 0x15B88 (88968), Block 0x02B7 (695) addr 0x2B710000 (0x2DC0E3A0) Uncorrectable ECC error!//unreliable Block
 Page 0x15B94 (88980), Block 0x02B7 (695) addr 0x2B728000 (0x2DC27810) Uncorrectable ECC error!//unreliable Block
 Page 0x15BB0 (89008), Block 0x02B7 (695) addr 0x2B760000 (0x2DC627C0) Uncorrectable ECC error!//unreliable Block
 Page 0x15BF4 (89076), Block 0x02B7 (695) addr 0x2B7E8000 (0x2DCF1B90) Uncorrectable ECC error!//unreliable Block
 Block 0x02B7 (695) addr 0x2B700000 (0x2DBFD600) 37 Bit error(s) in 30 page(s)!
 Block 0x02B8 (696) addr 0x2B800000 (0x2DD0B000) 4 Bit error(s) in 1 page(s)!
 Block 0x02B9 (697) addr 0x2B900000 (0x2DE18A00) 5 Bit error(s) in 1 page(s)!
 Page 0x15D0C (89356), Block 0x02BA (698) addr 0x2BA18000 (0x2DF3F870) Uncorrectable ECC error!//unreliable Block
 Page 0x15D4C (89420), Block 0x02BA (698) addr 0x2BA98000 (0x2DFC6570) Uncorrectable ECC error!//unreliable Block
 Block 0x02BA (698) addr 0x2BA00000 (0x2DF26400) ECC, 1 bit(s) corrected
 Block 0x02BA (698) addr 0x2BA00000 (0x2DF26400) 43 Bit error(s) in 29 page(s)!
 Block 0x02BB (699) addr 0x2BB00000 (0x2E033E00) 2 Bit error(s) in 1 page(s)!
 Page 0x15E14 (89620), Block 0x02BC (700) addr 0x2BC28000 (0x2E16BA10) Uncorrectable ECC error!//unreliable Block
 Page 0x15E26 (89638), Block 0x02BC (700) addr 0x2BC4C000 (0x2E1918B8) Uncorrectable ECC error!//unreliable Block
 Page 0x15E2E (89646), Block 0x02BC (700) addr 0x2BC5C000 (0x2E1A2658) Uncorrectable ECC error!//unreliable Block
 Page 0x15E70 (89712), Block 0x02BC (700) addr 0x2BCE0000 (0x2E22D6C0) Uncorrectable ECC error!//unreliable Block
 Block 0x02BC (700) addr 0x2BC00000 (0x2E141800) 42 Bit error(s) in 32 page(s)!
 Page 0x15EE2 (89826), Block 0x02BD (701) addr 0x2BDC4000 (0x2E31D8E8) Uncorrectable ECC error!//unreliable Block
 Block 0x02BD (701) addr 0x2BD00000 (0x2E24F200) ECC, 1 bit(s) corrected
 Block 0x02BD (701) addr 0x2BD00000 (0x2E24F200) 38 Bit error(s) in 29 page(s)!
 Block 0x02BE (702) addr 0x2BE00000 (0x2E35CC00) 1 Bit error(s) in 1 page(s)!
 Block 0x02BF (703) addr 0x2BF00000 (0x2E46A600) 50 Bit error(s) in 33 page(s)!
 Block 0x02C0 (704) addr 0x2C000000 (0x2E578000) 4 Bit error(s) in 3 page(s)!
 Page 0x16086 (90246), Block 0x02C1 (705) addr 0x2C10C000 (0x2E692438) Uncorrectable ECC error!//unreliable Block
 Block 0x02C1 (705) addr 0x2C100000 (0x2E685A00) ECC, 2 bit(s) corrected
 Block 0x02C1 (705) addr 0x2C100000 (0x2E685A00) 26 Bit error(s) in 17 page(s)!
 ...
 Block 0x038F (911) addr 0x38F00000 (0x3BF7C600) ECC, 19 bit(s) corrected
 Block 0x0390 (912) addr 0x39000000 (0x3C08A000) ECC, 24 bit(s) corrected
 Block 0x0391 (913) addr 0x39100000 (0x3C197A00) 4 Bit error(s) in 2 page(s)!

```

Block 0x0392 (914) addr 0x39200000 (0x3C2A5400) ECC, 12 bit(s) corrected
Block 0x0393 (915) addr 0x39300000 (0x3C3B2E00) ECC, 17 bit(s) corrected
Page 0x35F06 (220934), Block 0x06BE (1726) addr 0x6BE0C000 (0x719E9638) Uncorrectable ECC error!//unreliable
Block
Block 0x06BE (1726) addr 0x6BE00000 (0x719DCC00) 6 Bit error(s) in 2 page(s)!
Block 0x06BF (1727) addr 0x6BF00000 (0x71AEA600) 1 Bit error(s) in 1 page(s)!
Page 0x36074 (221300), Block 0x06C0 (1728) addr 0x6C0E8000 (0x71CEC590) Uncorrectable ECC error!//unreliable
Block
Block 0x06C0 (1728) addr 0x6C000000 (0x71BF8000) 13 Bit error(s) in 12 page(s)!
Page 0x36084 (221316), Block 0x06C1 (1729) addr 0x6C108000 (0x71D0E0D0) Uncorrectable ECC error!//unreliable
Block
Block 0x06C1 (1729) addr 0x6C100000 (0x71D05A00) 5 Bit error(s) in 2 page(s)!
...
Block 0x06EE (1774) addr 0x6EE00000 (0x74C6AC00) ECC, 2 bit(s) corrected
Block 0x06EE (1774) addr 0x6EE00000 (0x74C6AC00) 2 Bit error(s) in 1 page(s)!
Block 0x07FB (2043) addr 0x7FB00000 (0x867BBE00) BAD Block
Block 0x07FD (2045) addr 0x7FD00000 (0x869D7200) ECC, 1 bit(s) corrected
Block 0x07FD (2045) addr 0x7FD00000 (0x869D7200) 19 Bit error(s) in 2 page(s)!
Block 0x07FE (2046) addr 0x7FE00000 (0x86AE4C00) 1 Bit error(s) in 1 page(s)!
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
ECC checked in 36793 page(s), 228357 blank page(s) skipped, 174 blank codeword(s), 2607 codeword(s) with
blank ECC
Corrected 4349 ECC Bit error(s), 269 in ECC data
Detected 27 uncorrectable ECC error(s) in 27 page(s) //main indicator
Total Bit errors count during verification: 567
Completed in 3 min. 46 second(s) Code download speed 9.64 MiB/sec
As a result, at least blocks 216, 223, 474, 475, 689, 695, 698, 700, 701, 705, 829, 875, 881, 889, 893, 1726,
1728, 1729, 1812 need to be marked and seems not everything is so beautiful, although verification after
recording showed a minimum number of corrected errors and everything looked good.
You can notice the accuracy of bad blocks, they correspond to the most active parts in the chip. It can be
seen that in a short period of time we got absolutely non-working chip, with which the device will not be able
to work correctly. This is especially true for people who buy or sell finished, recorded NAND flash for repair.
Only by recording and checking , you can evaluate that chip is good for use.
After adding the BB, before the following operations, apply the new BB list by re-reading chips ID.
After adding and re-checking after a week, you can see in the log uncorrectable errors in blocks that have
already been marked. This means that the blocks to which the BB is reassigned is also worn out and requires
inclusion in the BB list. To do this, based on the list, you need to mark the block to which it refers:
NAND BAD Block 0x0714 (1812) replaced with 0x07FB (2043)

```

4.2.11 Application. Bad Block Table (BBT).

Purpose.

You cannot erase and work with faulty blocks. It takes a long time to determine the state by scanning the first page of each block when accessing, so a bad block table is used. BB table is the dump place where the list of bad blocks, their new locations and other service data is recorded. There are no universal ways to work with BB tables. Although certain specifications exist, manufacturers are not required to comply with them. Therefore, equipment manufacturers usually have their own view of the table and UFPI knows how to identify and work with some of them.

BBT location.

The bad block table can be located at the beginning, at the end of the chip or at other fixed offsets. You can specify the exact location of the selected table in the Bad Blocks definition settings.

BBT changes

Changing the BBT like any other data in the NAND requires further recalculation of the ECC of this page. Using bypass, skip, transfer of bad blocks without entering into the BBT will not be able to work correctly in the device. Based on the purpose and content of the Samsung D5500 table, it can be seen that each bad block has a reassignment. Programmer's log:

```
flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin' Analysis, size 0x88A7D800...
NAND Page 8192+436 (0x21B4), PIB 128, Block 0x10DA00 (1104384), 0x081C (2076) Blocks, IC geometry
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42
NAND BAD Blocks Table Samsung RFS, Block 0x07FD (2045)
NAND Partition Info Samsung RFS, Block 0x07FE (2046)
NAND LPCH Block 0x07FE (2046) at 0x86AE4C00
NAND LPCH TPCB 0x07FC (2044), Ages 1-1
NAND UPCH Block 0x07FD (2045) at 0x869D7200, 1 record(s) (Active)
NAND UPCH TPCB 0x07FC (2044), Ages 1-2, инфo 0x0000
NAND RBA 0x07FB-0x0785 (2043-1925), 0x77 (119) Blocks
NAND Partition #1, 'part-00-ID1B', Block 0x0 (0), count 0xD6 (214)
NAND Partition #1, addr 0x00000000 (0x00000000), size 0x0D600000 (0x0E163C00)
NAND Partition #2, 'part-01-ID1C', Block 0xD6 (214), count 0x100 (256)
NAND Partition #2, addr 0x0D600000 (0x0E163C00), size 0x10000000 (0x10DA0000)
NAND Partition #3, 'part-02-ID1D', Block 0x1D6 (470), count 0xD6 (214)
NAND Partition #3, addr 0x1D600000 (0x1EF03C00), size 0x0D600000 (0x0E163C00)
NAND Partition #4, 'part-03-ID1E', Block 0x2AC (684), count 0x410 (1040)
NAND Partition #4, addr 0x2AC00000 (0x2D067800), size 0x41000000 (0x4475A000)
NAND Partition #5, 'part-04-ID1F', Block 0x6BC (1724), count 0xC8 (200)
NAND Partition #5, addr 0x6BC00000 (0x717C1800), size 0x0C800000 (0x0D2A5000)
NAND Partition #6, 'UPCH', Block 0x7FD (2045), count 0x1 (1)
NAND Partition #6, addr 0x7FD00000 (0x869D7200), size 0x00100000 (0x0010DA00)
NAND Partition #7, 'LPCH', Block 0x7FE (2046), count 0x1 (1)
NAND Partition #7, addr 0x7FE00000 (0x86AE4C00), size 0x00100000 (0x0010DA00)
Loading IC BAD Blocks list...20 BAD Block(s)
Writing new BAD Blocks list...
NAND BAD Block 0x00D8 (216) replaced with 0x07FB (2043)
NAND BAD Block 0x00DF (223) replaced with 0x07FA (2042)
NAND BAD Block 0x01DA (474) replaced with 0x07F9 (2041)
NAND BAD Block 0x01DB (475) replaced with 0x07F8 (2040)
NAND BAD Block 0x02B1 (689) replaced with 0x07F7 (2039)
```


4.2.12 Application. ECC.

Purpose.

ECC - (error-correcting code) Without going into the physical subtleties of error formation, this is just for understanding that the number of errors increases over time, and the more erasure cycles the flash block, the faster these errors can accumulate in it. *NAND flash initially involves the use of a correction code. To maintain the operation of NAND, Hamming, BCH, Reed-Solomon correction codes with various options and derived from them using encryption and other standard and non-standard techniques are used. The full NAND page size consists of page and spare. Usually page contains data, and spare code correction.*

No changes, patches, BBT recalculations can be made without recalculating the ECC code.

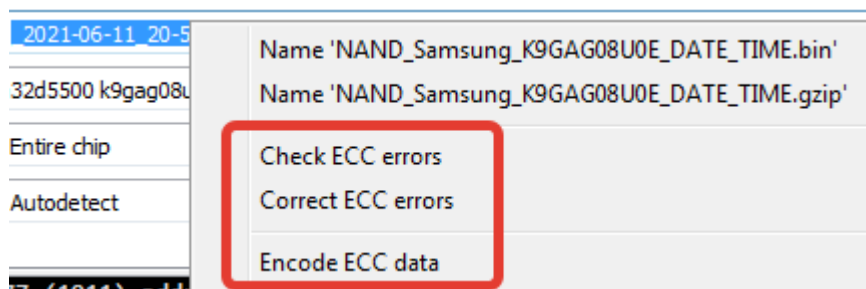
No guarantees for the dump and its entry in the NAND (even if there is not a single BB in it) can be given without ECC verification.

ECC Selection.

It is difficult or even impossible for a regular user to define ECC parameters and algorithm, so UFPI simplified the task by entering an Autodetect of the algorithm. To do this, it is enough to insert chip (having the desired dump) and read the ID when the Autodetect scheme is set. The name of the algorithm will be shown in the log:

NAND ECC correction Autodetect MSTAR_P8K_SP436_CW8_S12L42 (no ECC errors in Page #0)

Alternatively, if you select a dump with a double click on the "Read To" or "Write From" field, select Check or Correct ECC from the context menu..



So, you can check the file for correctness even before programming and installation chip into device.

NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42 (Autodetect)

Проверка ECC в 'E:\temp\delete\111\NAND_Samsung_K9GAG08U0E_2021-05-16_23-20.bin'...

ECC checked in 36767 page(s), 228940 blank page(s) skipped, 7 blank codeword(s), 168 codeword(s) with blank ECC

Detected 1 ECC Bit error(s), 1 in ECC data

Very often files from files uploads, marked "verified" or shared in forums, have uncorrectable errors. Files with uncorrectable errors cannot guarantee proper operation.

Designation

To facilitate and visually perceive ECC schemes, names have been introduced that can be guided without delving into the complexity of the algorithm. For example, a dump from Samsung D5500.

When reading ID, the algorithm is determined automatically:

NAND ECC correction Autodetect MSTAR_P8K_SP436_CW8_S12L42 (no ECC errors in Page #0)

MSTAR — manufacturer of the Mstar processor. You can find by the inscription on the processor or description on the Internet.

P8K — page size 8192 bytes. You can see from the NAND or Log options when reading the ID in the programmer.

SP436 is the size of spare 436 bytes. You can see from the NAND or log parameters when reading the ID in

the programmer.

S12 — is the number of bytes from the start of spare to the start of the correction code.

```

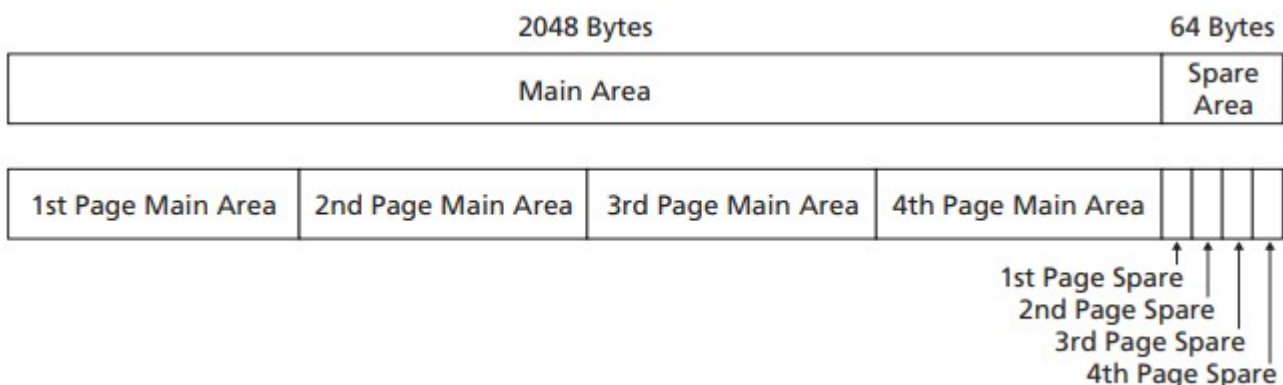
00001ff0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff | .....
00002000 ff ff ff ff 01 00 fe ff ff ff ff ff ff | .....j...
00002010 e6 c2 f4 af 7a ea 54 a4 13 2e 8f f3 56 e1 84 c0 | ....z.T.....V...
00002020 e2 b2 fc 2e 59 fb 69 fc fc 82 b6 d6 16 7d 5f 39 | ....Y.i.....} 9
00002030 89 a2 35 7a f1 ce 40 01 bf fe ff ff ff ff ff ff | ..5z..@.....
00002040 ff ff bc ea 73 1f e0 77 12 c2 cc 63 cb 40 17 cc | ....s..w...c.@..
00002050 61 b8 5b 05 5b 49 7a 26 8d e3 e9 c8 5f 4e f8 af | a.[.[Iz&.... N..
00002060 c9 5f c5 a3 29 11 62 77 e4 be f7 26 00 00 31 09 | . ..) .bw...&..1.
    
```

L42 — the number of bytes of the correction code.

```

00001ff0 ff ff ff ff ff ff ff ff ff ff ff ff ff ff | .....
00002000 ff ff ff ff 01 00 fe ff ff ff ff ff ff | .....j...
00002010 e6 c2 f4 af 7a ea 54 a4 13 2e 8f f3 56 e1 84 c0 | ....z.T.....V...
00002020 e2 b2 fc 2e 59 fb 69 fc fc 82 b6 d6 16 7d 5f 39 | ....Y.i.....} 9
00002030 89 a2 35 7a f1 ce 40 01 bf fe ff ff ff ff ff | ..5z..@.....
00002040 ff ff bc ea 73 1f e0 77 12 c2 cc 63 cb 40 17 cc | ....s..w...c.@..
00002050 61 b8 5b 05 5b 49 7a 26 8d e3 e9 c8 5f 4e f8 af | a.[.[Iz&.... N..
    
```

CW8 — The number of parts the page is divided into. The approximate illustration in the Micron documentation looks like this:



Only in the dump from D5500 page 8192 bytes, spare 436 and divided into 8 parts. This can be clearly seen by counting how many parts of the correction code for the full page in spare, separated from each other, with bytes 0xFF.

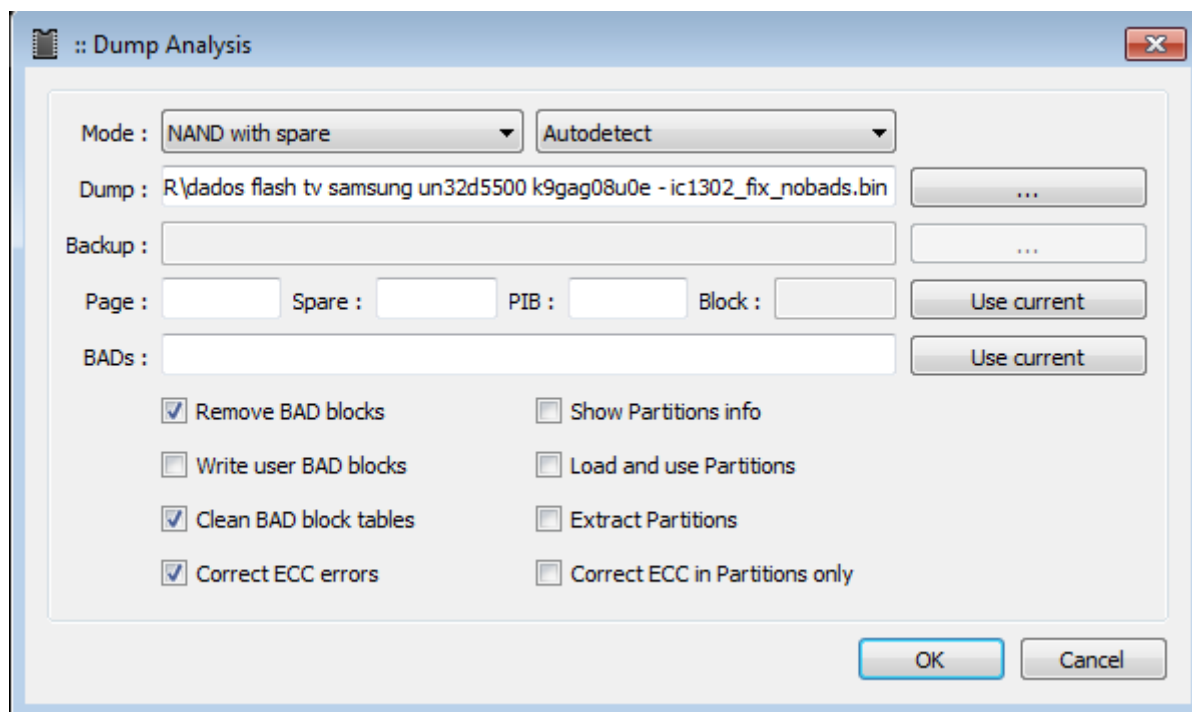
The remaining additions to the name of the algorithm for the user do not carry useful information.

4.2.13 Application. Dump Analysis. Dump preparation. Creating a "clean" dump.

Connect the programmer with the chip.

In [ECC](#) settings select [algorithm](#). Usually, if the algorithm is known, then auto-determination works.

From the «Util» menu, select «Dump Analysis».



In the «Mode» field, select «NAND with spare» and «Autodetect». In some cases, when the analysis cannot automatically recognize, you must explicitly specify dimensions by selecting «User Geometry» or by setting the desired flash, read the ID, and select «IC Geometry».

In the Dump field, select the dump to correct.

Fields "Page", "Spare", "PIB" need to fill, if as mentioned earlier, auto-determination did not recognize the NAND geometry by dump. Here you need to fit the page size, spare and the number of pages in the block. The Block field automatically counts the size of the block. If the required chip is installed in the programmer and the ID is previously read, you can click the "Use current" button and the fields will be filled with the current (from this chip) page size, spare and their number.

The Bad Blocks field is used to generate user [BB](#) in the dump to be prepared and is used when the "Write user BAD blocks" check box is set ON. It can be useful in preparing an individual dump for a specific chips instance for recording without the functions of bypassing the BB. The "Use current" button is automatically filled in [current BB list that was received while reading ID](#). These parameters are not needed to create a clean dump for further recording with BB control in UFPI,

"Show Partition info" and "Load and Use Partitions" check boxes uses to analyze the layout of partitions and put them in the [«Partitions»](#) RW Mode.

If you select the "Extract partitions" check box, the analysis will create separate dump section files in the dump folder.

The "Correct ECC in Partitions only" check box is necessary when using different ECC schemes in different partitions of the same dump. This check box automatically determines ECC on each partition individually.

A demonstration of the application can be found in the supplementary [appendix](#).

4.2.14 Application. Dump analysis when reading ID and writing of "clean" dump with control "Use reserved area."

Additional options.

Enable "«[A Dump Analysis when Reading ID](#)» in "Add. Options."

For normal conditions and chips, set "Asynchronous" and "Do not write empty data." ON.

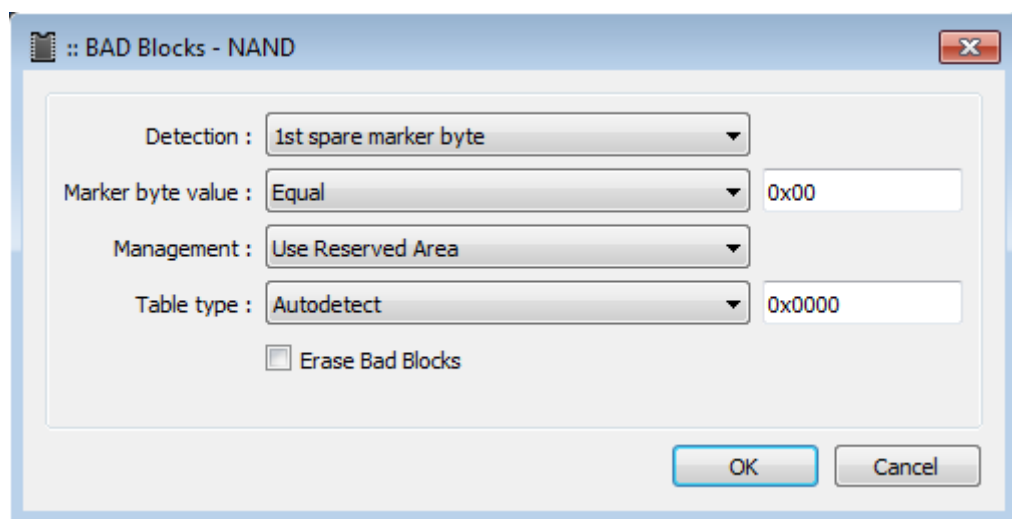
Bad Block Settings:

Enable BB detection by marker. This is usually the first byte with a value of 0x00.

For the supported BB tables, set the management to "Use reserved area."

The table type is usually determined automatically in the case of using an analysis when reading an ID. Therefore, select AutoDetect.

It is not recommended to erase bad blocks unnecessarily. If only the BB token does not match and too many BBs were defined on it when reading the ID.



ECC setup.

For normal NAND, you must select a scheme with Autodetect.

Ffile.

Select [«clean»](#) dump. *If you select a dump containing a BB in the log, the analysis will receive a corresponding error.*

Configuring the programmer for recording.

Read the ID, dump analysis will configure the programmer for further work with the chip and dump. Processing of replaced blocks will be performed when reading/writing/erasing automatically by firmware of the programmer.

Erase Chip.

If necessary, enter a list of bad blocks via "Miscellaneous - BAD Blocks - Mark BADs. [Consider blocks that already have been redirected.](#) Usually, candidates for the bad blocks appear after a short time on old flash. Therefore, you can overwrite several times, marking new bad blocks.

Read the ID again, dump analysis will configure the programmer for further work with the chip and dump. Operation of replaced blocks will be performed when reading/writing/erasing automatically by firmware of the programmer.

Write, Verify.

If uncorrectable errors are found during the verification process, repeat the software configuration process with additional bad blocks and record, verify again.

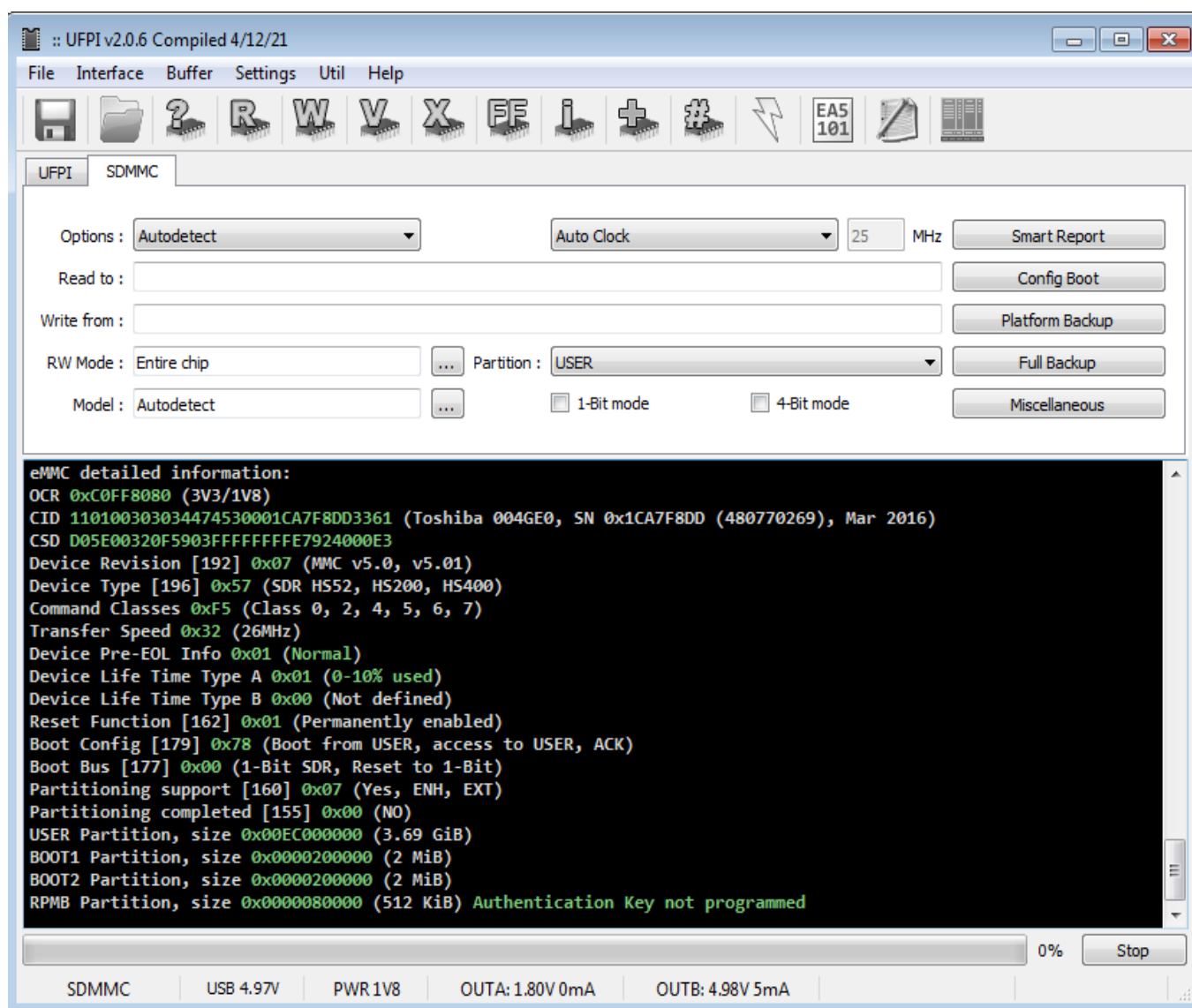
Verification can be carried out after an unlimited time. To do this, perform the items, except for erasing and recording chip.

A demonstration of the application can be found in the supplementary [appendix](#).

4.2.15 Description in the log during operation.

Under construction

4.3 SD/eMMC.



Sockets.

A large quantity of sockets for eMMC is listed in the application. [application](#).

Features.

- Fastest among famous competitors. Speed is limited only by the properties of the chip type, the connection and the USB port of the computer.
- It allows in most cases to work successfully in ISP mode with fast speed.
- Does not require a chip model. The model and parameters are automatically defined.
- Advanced options with partitions and boot options, as well as the ability to use configurations for backups of platforms and partitions and much more. It is advisable to familiarize yourself with the main provisions in the [JEDEC](#) specification.
- Expansion of functionality using scripts.
- Operation of 1-4-8 bit mode, as well as synchronously, with blocks with integrity control and repeated re-reading in case of errors, allows you to subtract problem chips.
- EMMC need not to be [ERASED](#).

4.3.1 Options.

Options : MHz

Read to :

Write from :

Auto/MMC/SDC - Sets the device type.

Automatic Frequency Selection/User Frequency. Usually, you can leave an "auto-selection" to work with a healthy chip in the socket. It may be necessary to lower the frequency during unstable operation, but there is no point in lowering below 5-10MHz. It is possible to increase for experimental purposes even to 100 MHz, however, at such frequencies eMMC even in sockets cannot work stably. In practice, it makes no sense to increase at HS52 higher than 52MHz, and when HS200 more than 80-85 MHz, because unstable work often begins. After entering in the frequency field, press "Enter" to apply the parameter and then read the ID again.

4.3.2 «Read to» и «Write from».

Standard for almost every module. [Double click calls MENU](#) autonames, history and other.

4.3.3 RW Mode.

Described in another [chapter](#) with [samples](#).

4.3.4 Model.

NEED NOT.

4.3.5 Partition.

Selection partition: [USER/RPMB/BOOT1/BOOT1/GPP1-4](#).

4.3.6 4-bit mode / 1-bit mode.

The programmer automatically determines in which mode the flash is connected. If 8 (8-bit mode is available only in [Pro](#) version of the programmer) lines are connected in the correct order. If 4, then the configuration for working in 4-bit mode will automatically apply.

4.3.7 Smart report.

Display Smart report for eMMC revisions usually up to v5.0. It describes the state of "health" of the chip. It is worth paying special attention to the following points:

"Error mode" - the Norm must be 0xD2D2D2D2.

"number of spare units remaining" - units must remain > 10 pieces, or enough to continue operation.

"ECC Uncorrectable Error Count" [ECC](#) these errors should not have a good chip.

It is important to know that in the revision eMMC above v5.0, «[Health life time](#)» is already present, it is automatically displayed in the log if it is available and displays the current level of wear and tear:

Device Pre-EOL Info 0x01 (Normal)

Device Life Time Type A 0x01 (0-10% used)

Device Life Time Type B 0x01 (0-10% used)

4.3.8 Download configuration.

This window contains the basic parameters that should be set when replacing the flash. You can view the settings you want to set when reading your native flash ID. In fact, there is a calculator of three bytes (162,

177, 179) from the [eCSD](#) register At the top, select options from the drop-down menus, and at the bottom based on the bit configuration, a byte is immediately compiled. It also works vice versa, by inscribing a byte and clicking "Apply" you can see the decryption by items from above. Click OK to write the values to the appropriate registers.

Separately issued on *RST_N [162] bytes*. As a fuse, the check box is cleared and a warning is signed that the parameter "OTP" (that is, this usually means write once without the ability to change it back).

All these bytes can be written to the menu individually [«Add.features»](#)

4.3.9 Full Backup.

Save the full dump of USER, BOOT1/2, RPMB partitions (unless the check box is set ON in the "Exclude RPMB from full backup" module settings) and eCSD with full log to one folder.

4.3.10 Platform Backup.

Simplifies routine tasks to migrate regions and partitions that have bindings and keys in the dump. For an example, see the [appendix](#).

PBI is a normal INI file, encoding UTF-8.

```
;List of manufacturers.
[groups]
group1=LG
group2=Philips
group3=Samsung
;List of models.
[LG]
model1=LC_LD_LE42B_42G

[LC_LD_LE42B_42G]
Name=LC_LD_LE42B_42G
act1=0x3F00000:0x80000:USER:sestore
act2=0x3F80000:0x80000:USER:sedata

[Philips]
model1=QV14
model2=QV15
model3=TPM14
model4=TPM15
model5=QM16

[QV14]
Name=QV14
...
```

Each module will use .pbi with specific names of type tv.mmc.pbi, sat.nand.pbi, etc.

There can be multiple .pbi files with the same extension for one module of type tv.mmc.pbi, sat.mmc.pbi.

Имя	Дата изменения	Тип	Размер
sat.mmc.pbi	09.11.2020 21:41	Файл "PBI"	0 КБ
sat.nand.pbi	09.11.2020 21:42	Файл "PBI"	0 КБ
tv.mmc.pbi	24.11.2020 19:18	Файл "PBI"	2 КБ

After pressing the "?" button and reading the ID, the "Platform Backup" menu button becomes available.

.mmc.pbi files are used for eMMC.

For SD cards, .sdc.pbi files are used.

PBI files can be compiled, exchanged, or searched in a forum. [example1](#) , [example2](#).

4.3.11 Additional features/ Miscellaneous. (some items appear after reading the ID)

Analysis on ID	▶
Extended CSD	▶
Samsung Internals	▶
Hynix Internals	▶
Read mode	▶
Write mode	▶
Erase mode	▶
Timing mode	▶
Partitioning	▶
RPMB	▶
Password	▶
Power Cycle	
<input checked="" type="checkbox"/> Async IO	
<input checked="" type="checkbox"/> Pins Test	

Analysis on ID.

Selects an object to search for the partition table. You can assign "eMMC" or "Dump" separately, or a combination by selecting the priority "Dump, eMMC." If a .UDEV description file is opened, the list of spartitions from UDEV is loaded. Otherwise, the program will try to analyze markup with its own algorithms.

Extended CSD.

Write a single byte to the register, or the entire [eCSD](#) file.

When reading the ID in the log, some bytes of eCSD are described and signed. The number of the byte in the register is specified in square brackets. Next, its meaning and decryption. When writing a byte, it is sufficient to specify its number and value. Example:

Boot Config [179] 0x00 (No Boot, access to USER)

Each time you read an ID, files are created in the ufpi\Logs\eMMC folder, including the full eCSD. They are signed with chips number and date/time of creation. When writing a full file, you can select it, edit it in hex editor if necessary.

It is important to remember that not all registers are eCSD writable and not all registers are recoverable. Different manufacturers have different reserved registers, which cannot be changed without consequences. Please read the JEDEC specifications carefully.

Basic bytes can be configured in the calculator [CONFIG CALCULATOR](#).

Samsung features.

A great choice of features are available primarily for Samsung eMMC revisions up to v5.0. Collected in one

place.

«Write [CID](#)» and «Write [CID #2](#)» — writes a custom CID value, even from another eMMC manufacturer..

«Write [CSD](#)» — writes a custom CSD value.

"Remove Write Protection" - Remove write protection that is set to CSD and eCSD and is displayed in the log when reading ID:

Permanent Write Protect

or

Boot Write Protection 0x0A (BOOT1_PERM BOOT2_PERM)

or

User Write Protection 0x10 (US_PERM_WP_DIS)

"Formatting" - Clears the contents of partitions and registers to factory status.

"Firmware Read" and "Firmware Write" are special commands for reading and writing internal FW eMMC. The "Read To" and "Write From" fields must be filled with the appropriate read/write files. The firmware file read by extraneous programmers may be incompatible, or need to be corrected. The firmware operation of the internal FW in the eMMC does not guarantee the correct further operation of the chip.

Hynix features.

"Clear Lifetimer (v5.1)" - Experimental function to reset the operating time counter «[LifeTimer](#)». Made just to extend the life of devices for which it is critical. Works on eMMC Hynix revision v5.1. Besides dumping of LifeTimer, writes down "partitions Are Complete" value in byte [155] of eCSD.

Read mode.

"Auto" - Optimal mode for reading good flashes in the socket at high speed. It's the default.

"Single blocks" - reading occurs sequentially on one block with check of integrity of the received block CMD17 (READ_SINGLE_BLOCK). The default block size is specified in the CSD register. Suitable for working with faulty chips, where monitoring and re-reading of problem blocks is required. The speed of this mode is the lowest.

"Multiple Blocks (Open)" - read multiple blocks using the CMD18 (READ_MULTIPLE_BLOCK) command and is interrupted by the stop command.

"Multiple Blocks (Count)" - read multiple blocks using the CMD18 (READ_MULTIPLE_BLOCK) command and stops. The counter is set by the CMD23 (SET_BLOCK_COUNT) command.

"Multiple Blocks (One request)" - read with CMD18 (READ_MULTIPLE_BLOCK) without stopping until the end of the specified region.

"Ignore Read Errors" - Do not stop with read errors, read from the next block. This option can be useful when reading faulty chips.

Write mode.

"Auto" - Optimal mode for recording good chips in the block at high speed. It's the default.

"Single blocks" - similar to reading.

"Multiple Blocks (Open)" - similar to reading.

"Multiple Blocks (Count)" - similar to reading.

"Multiple Blocks (Single Query)" - similar to reading.

"4-bit mode at VCCQ = 1V8" - during recording, sets the mode to 4 bits if VCCQ 1,8V. It is recommended that you use this option for a stable and predictable result.

Erase mode.

«Auto» select is the default. [eMMC erasure](#).

"Single Blocks" - Erases commands in one block.
 Multiple Blocks - Erases multiple block commands.
 «Write Zeros» — fill 0x00.

Timing mode.

"Auto" Select - Optimal mode. Set by default.

"Legacy 26 MHz" - compatibility mode with obsolete MMC cards.

"High speed 26 MHz" - high-speed mode (HS52), with a set frequency of 26 MHz.

"High speed 52 MHz" - high-speed mode (HS52) with a set frequency of 52MHz. This mode is set by default as the most stable and suitable for most cases.

"High speed 200MHz" - High-speed mode (HS200) up to 200MHz. In practice, above 90MHz, stable work in common BGA sockets is difficult to achieve.

"Sampling Delay" and "Drive Delay" - parameters can be useful for matching lines when operating at a high frequency. To use, it is enough to set the maximum value, if the errors are missing, to lower the stable value.

Partitioning.

BOOT/ RPMB / GPP. - some eMMC, in particular Samsung before revision 5.0 support the partition markup function BOOT, RPMB, GPP. To modify, enter a dimension (kB/MB) and an integer value.

RPMB.

"Authentication" - enter the key for further operation and recording in RPMB.

Write Authentication Key - Set the authentication key only if it has not previously been written. This is according to specification 32 a byte key in HEX form (i.e. 0x...). *Before using, you should understand if this action is necessary. The operation for most chips is irreversible.*

Disabled - Disable RPMB status and polling when reading ID. According to users, there were faulty Toshiba eMMC, which, after polling RPMB status, ceased to read until the next reboot.

Password.

Unlock - Requires a password (in ASCII format) to start a locked chip session. For example, in the log, such a chip will be:

```
eMMC ID 1501003030585056300304051227C0CF, DAT0 Locked!
```

"Lock" - End the session with the blocked chips.

"Clear" - Allows you to clear a known password.

"Set" - Set the password to block the chip.

"Erase" - Erases the password and clears the contents of the chip. This operation does not require a password.

"Brutforce" - automatic password selection. You can specify length, symbols, and stop and continue to search.

Power Cycle

Full power-off cycle, delay, and reinitialization and read ID.

Async IO.

[similar to other modules](#). *It is important to remember that it is not possible to read and ignore errors in asynchronous mode.*

Pin Test

[Similar to other modules.](#)

4.3.12 Application. CID, CSD, eCSD, OCR registers.

CID (Card identification data) .

MMC register, which contains data from which a memory card can be identified (serial number, manufacturer ID, date of manufacture, chip name, etc.), length 16 bytes. There are devices where the name of the chip contains the loader, when replacing the chip with a not similar one, sometimes you have write the CID from the same chip or from your native one, if it can be read. Such operations are applicable only on Samsung revision eMMC up to v5.0. The following is a table in the documentation that describes the value of this register.

CSD (Card-specific data).

CID Fields Name	Field	Width	CID slice	Value
Manufacturer ID	MID	8	[127:120]	70h
Reserved		6	[119:114]	0h
Device/BGA	CBX	2	[113:112]	1h
OEM/Application ID	OID	8	[111:104]	0h
Product name	PNM	48	[103:56]	(4D3532353136h"M52516")
Product revision	PRV	8	[55:48]	51h*
Product serial number	PSN	32	[47:16]	Random by Production
Manufacturing date	MDT	8	[15:8]	month, year
CRC7 checksum	CRC	7	[7:1]	-(Note 1)
not used, always "1"	-	1	[0]	1h

Contains various information about the memory card (from the size of the sector of the memory card to consumption in read/write mode). 16 bytes long.

Important areas in CSD are 12 and 13 bits: the value of this register.

PERM_WRITE_PROTECT -Permanent write/erase protection.

TMP_WRITE_PROTECT - Write/erase protection prior to card reboot.

If the permanent protection check box of the working device is set ON, it will turn on and work, but all changes will be reset after the next reboot. Some such chips can cure formatting, or firmware internal FW, but their reliability will still remain in doubt.

extCSD (Extended Card-specific data).

Contains all kinds of additional information and memory card modes. It is 512 bytes long. A rather long register - but the main part of it ReadOnly - never change and are registered at the factory. The main part of the register is described by JEDEC specifications, but individual bytes are reserved and have an individual purpose for each manufacturer, the values of which are not in the public documentation.

OCR (Operation conditions register).

Contains data on the power supply of the memory card, the type of power supply of the memory card, the status of the card initialization process.

OCR Register Definitions OCR bit	VDD voltage window	High Voltage MultimediaCard	Dual voltage MultimediaCard and eMMC™
[6:0]	Reserved	00 00000b	00 00000b
[7]	1.70 - 1.95V	0b	1b
[14:8]	2.0-2.6V	000 0000b	000 0000b
[23:15]	2.7-3.6V	1 1111 1111b	1 1111 1111b
[28:24]	Reserved	0 0000b	0 0000b
[30:29]	Access Mode	00b (byte mode) 10b (sector mode)	00b (byte mode) 10b (sector mode)
[31]	(Device power up status bit (busy)) ₁		
Note1 : This bit is set to LOW if the Device has not finished the power up routine.			

So the normal response of the current running eMMC is OCR 0xC0FF8080 (3V3/1V8), that is, the last [31] bit is set to 1 if the card activation procedure is completed successfully.

4.3.13 Application. USER, BOOT1, BOOT2, RPMB, GPP1, GPP2, GPP3, GPP4 partitions.

RPMB (Replay Protected Memory Block) .

A secure partition that supports a private key verification mechanism. It can be used by the manufacturer of equipment to protect against cloning and ignore of protected software functions. UFPI supports this section in its entirety as described in the specification. Allows you to write, read data, set a secret key and size (if provided by the manufacturer). This partition is always created in the factory (usually size 4MB) and is present on the chip, even if it is not used and empty.

BOOT1, BOOT2.

These sections are also always present on the chip, even if they are not used and empty. Typically, the size is 4MB. Designed usually to accommodate the boot area. UFPI supports these sections and they are described in the specification. Allows you to write, read data and set the size (if provided by the manufacturer).

GPP1 – GPP4.

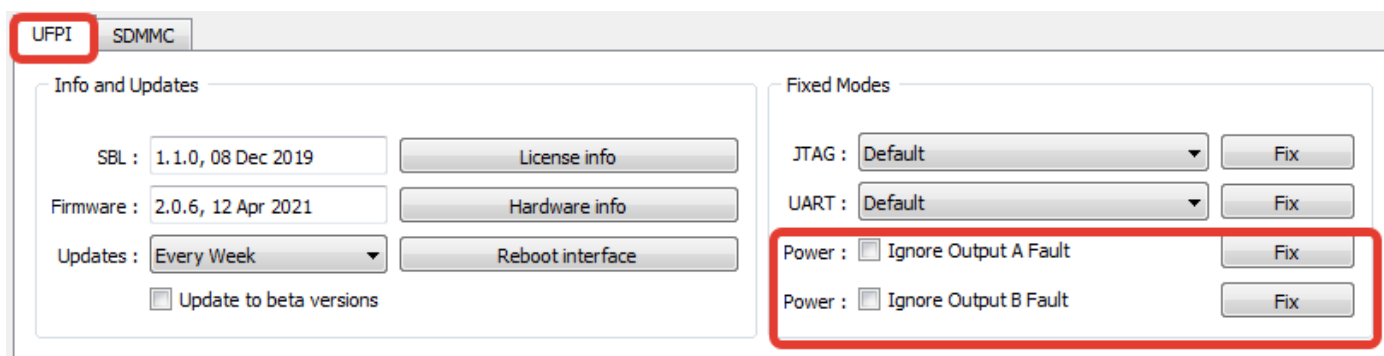
General purpose partitions. By default, they are not enabled in the manufacturer, but the user can activate them. Their number and size are set by the user and can be from 1 to 4. UFPI supports these partitions and they described in the specification. Allows you to write, read data, and set quantity and size (if required by the manufacturer).

USER.

The main user-defined partition that can contain the loader, partition table, partitions, and other data. Its capacity is the total volume minus the capacity occupied by the previous partitions.

4.3.14 Application. ISP connect mode.

To operate in this mode, in some cases, it may be necessary to turn off current protection.



The following Pins are used to connect:

VSS (GND) — ground

VCC(VDDF) — core power supply, nand, microcontroller = 2.7v-3.6v.

VCCQ(VDD) — interface power supply = 1.7v-1.95v or 2.7v-3.6v.

CMD — two-way bus for transmitting and receiving commands.

CLK — clock.

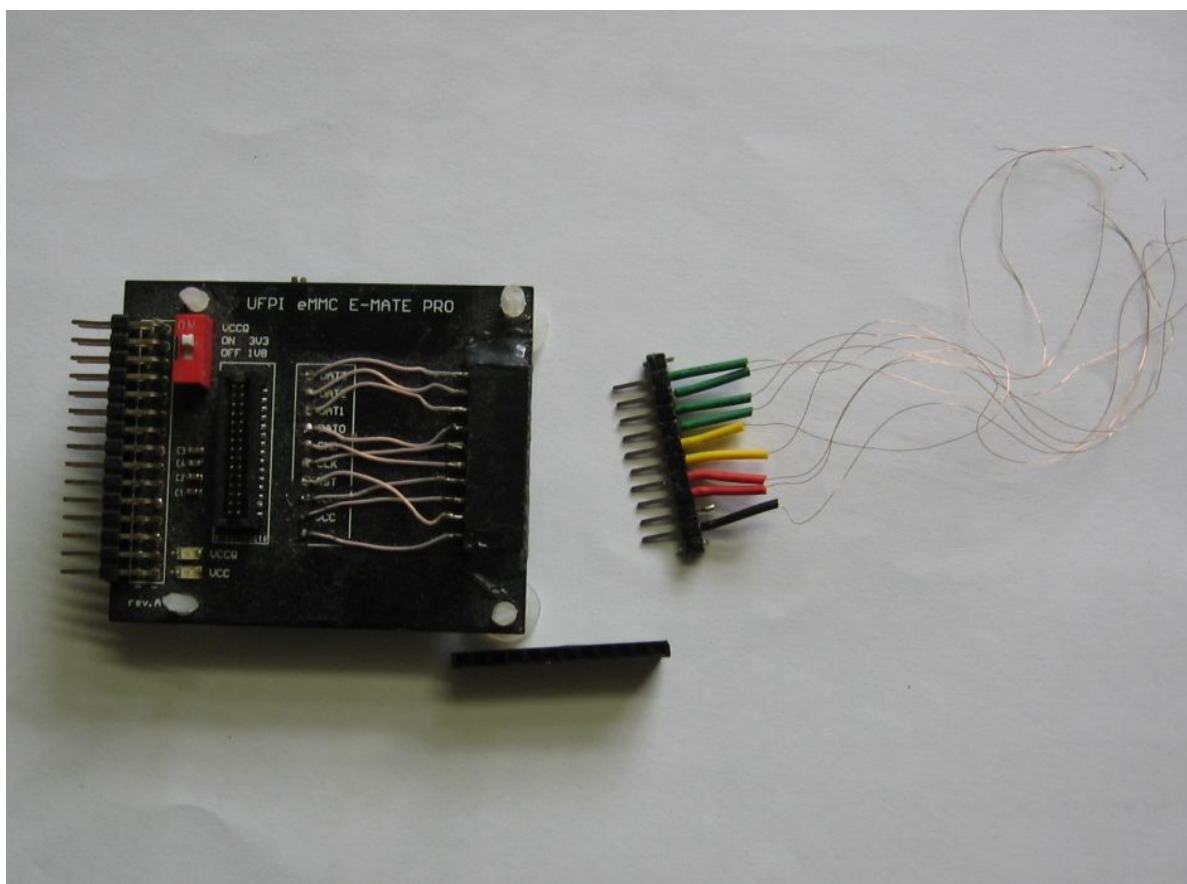
D0-D3 — data transmission and reception buses

RST — reset line. The chip can be software dependent on this bus - 162 bytes per extCSD. If 1 is prescribed there, then there must be 1 on this bus. In most cases, this bus is connected by the resistor to power, but there are sets where the processor controls this bus, in this case, to connect in isp mode, you need to supply regular power and stop the processor or put it in a third state, for example, through reset.

On some mainboards, it is enough to connect at one point. 3.3V There, either the core and interface power are combined, or the 1.8v output stabilizer is powered by the 3.3V bus to power the interface.

Some mainboards have to remove the throttle that is powered by the chip so as not to power the rest parts of the board. Sometimes you have to cut tracks or remove pass-through resistors to eliminate the influence of the processor. In these cases, you need to make sure that the resistors of the data bus tightening and the reset resistor are powered.

It is best to use a soft wire with a thickness of 0.1 mm and a length of not more than 10-15 cm. You need to be very careful when soldering, since you can damage the ends of SMD resistors and get a mine that does not turn on.



When trying to power the mainboard by the programmer, the currents are hundreds of mA, it should be borne in mind that the usb cable, and the port should be of good quality.

If the board is powered by an external source rather than a programmer, set the VCCQ power switch on the adapter to the position corresponding to the voltage applied to the mainboard. You cannot combine the VCCQ of the programmer and the mainboard at the same time.

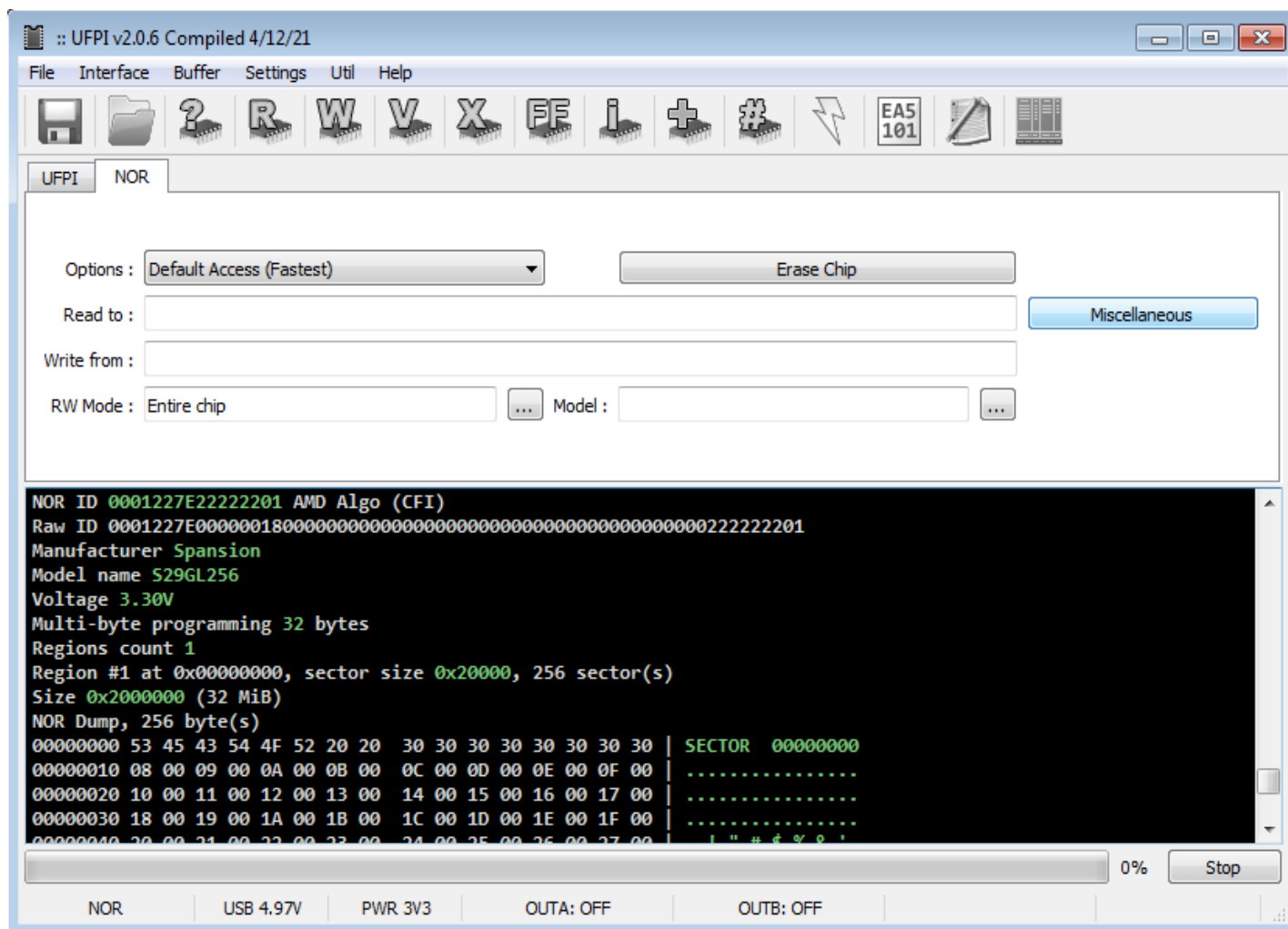
For such work, it is more convenient to disable the «Read ID» after socket connection" settings and control power supply through the. [power button](#).

If you have not described socket or when using the layout board, you need to familiarize yourself with the [diagram](#) and note that the VCC of the eMMC programmer is connected to the VCCQ.

4.3.15 Description of items in the log during operation.

Under construction...

4.4 NOR.



Sockets.

Works with sockets [2046](#) or [2060](#).

Features.

- • The chip parameters are determined automatically, provided that they are properly connected. Chip selection in this module is not active.
- • For the most correct operation, UFPI represents the address space of regions, sectors with individual dimensions.
- Supports [CFI](#) and [manual settings](#).
- Ability to work with scripts.

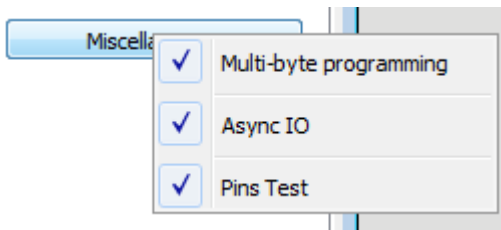
Erasure.

Either the < X > button or the Chip Erase button. In the first case, it is erased by sectors, it is necessary when working with parts/sectors/sections. In the second, the entire chip is erased with one command.

RW Mode.

The same as with other chips [working with chips regions](#).

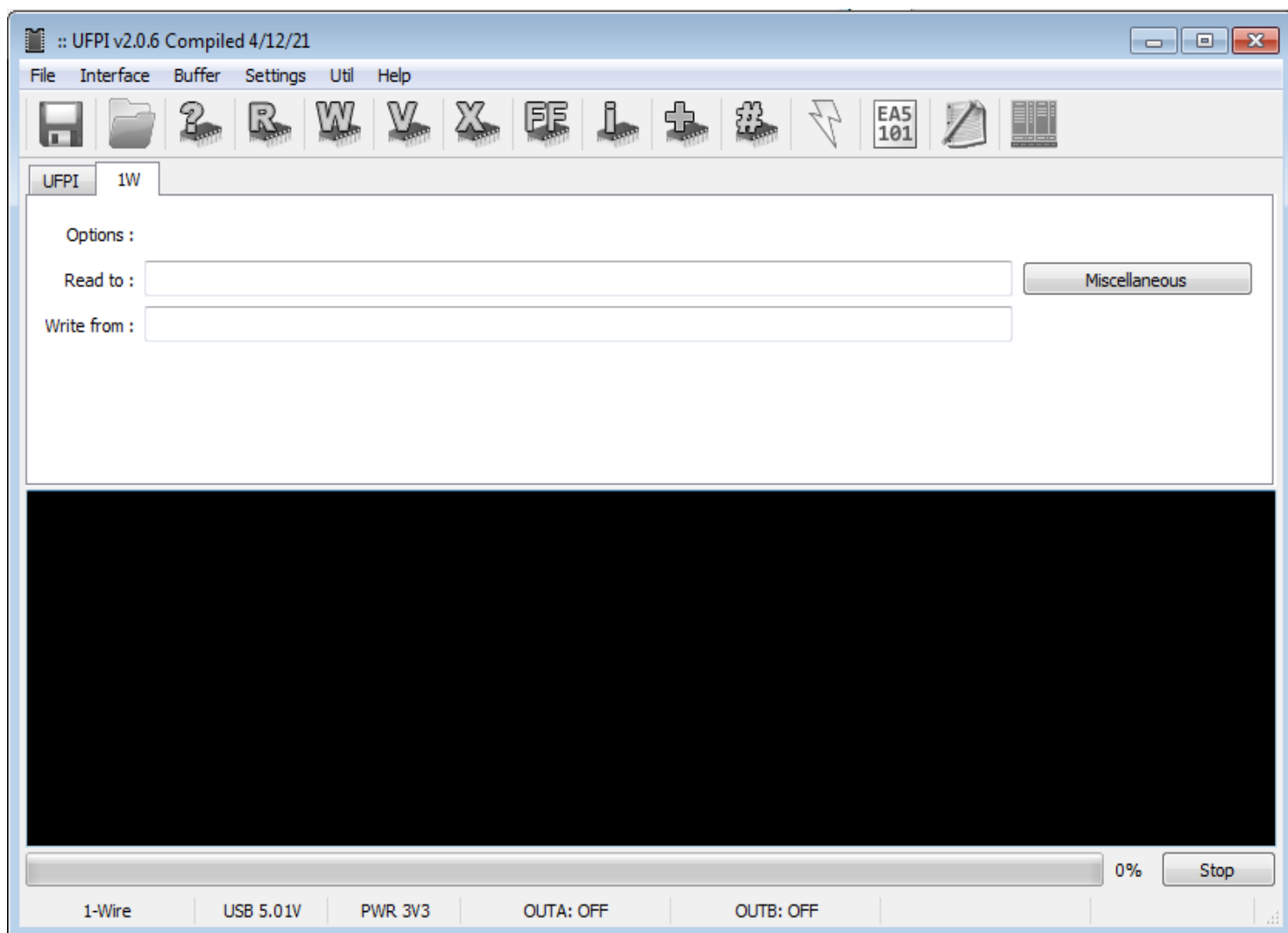
Additional features/Miscellaneous.



You can select «[Async mode](#)», «[Pin test](#)» и «Multi byre write».

Multi-byte writing, if supported in the current chip, allows you to repeatedly reduce recording time at large volumes.

4.5 1-Wire.



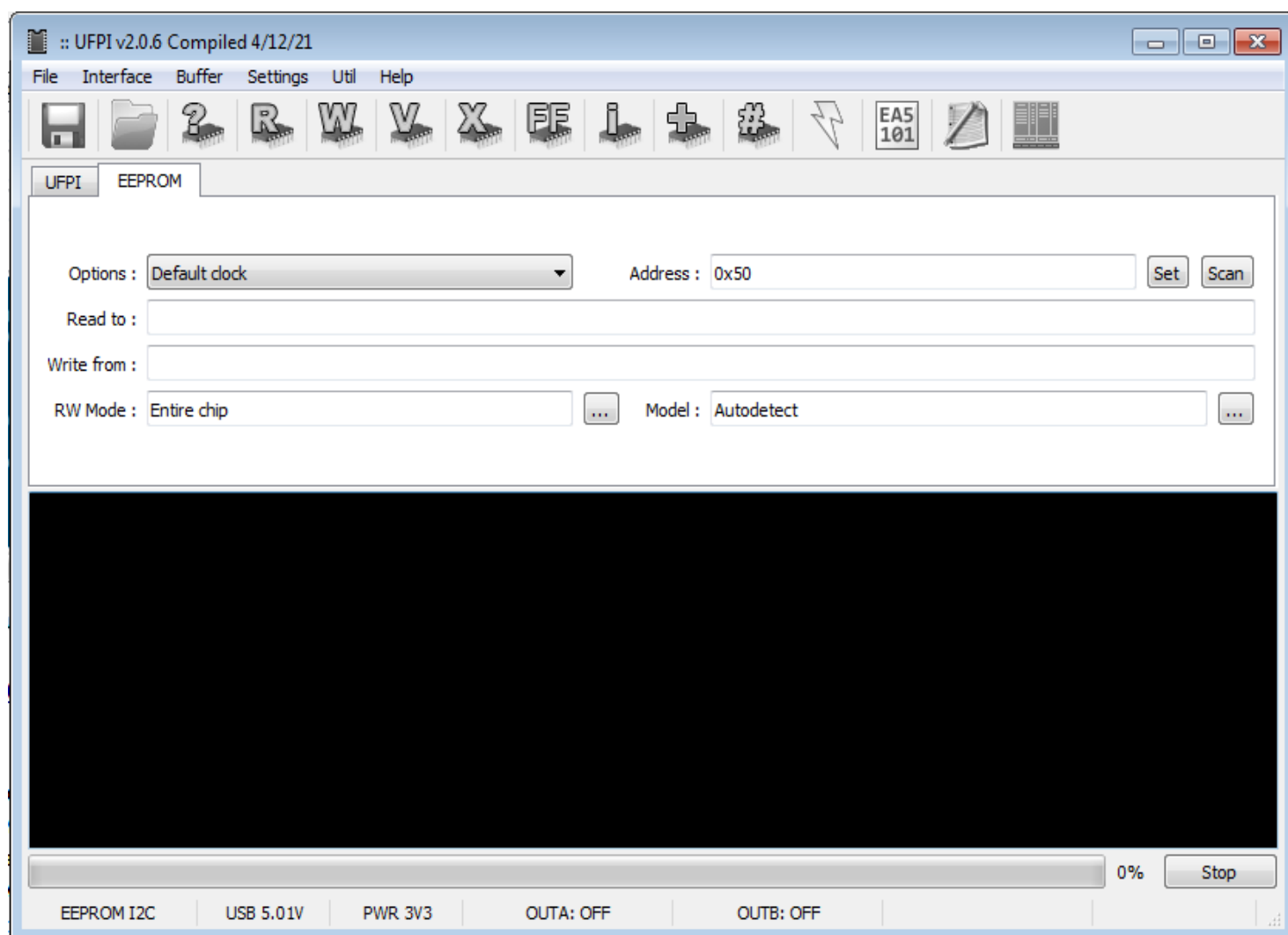
Sockets.

Works with sockets [2046](#) or [2060](#).

Features.

- This module does not have standard read/write and other functions. It works through scripts. The forum has ready to use scripts, descriptions and discussions for them.
- Connected to GND and SDA pins (1W IO). [pinout](#).

4.6 EEPROM I2C.



Sockets.

With sockets [2046](#) or [2060](#).

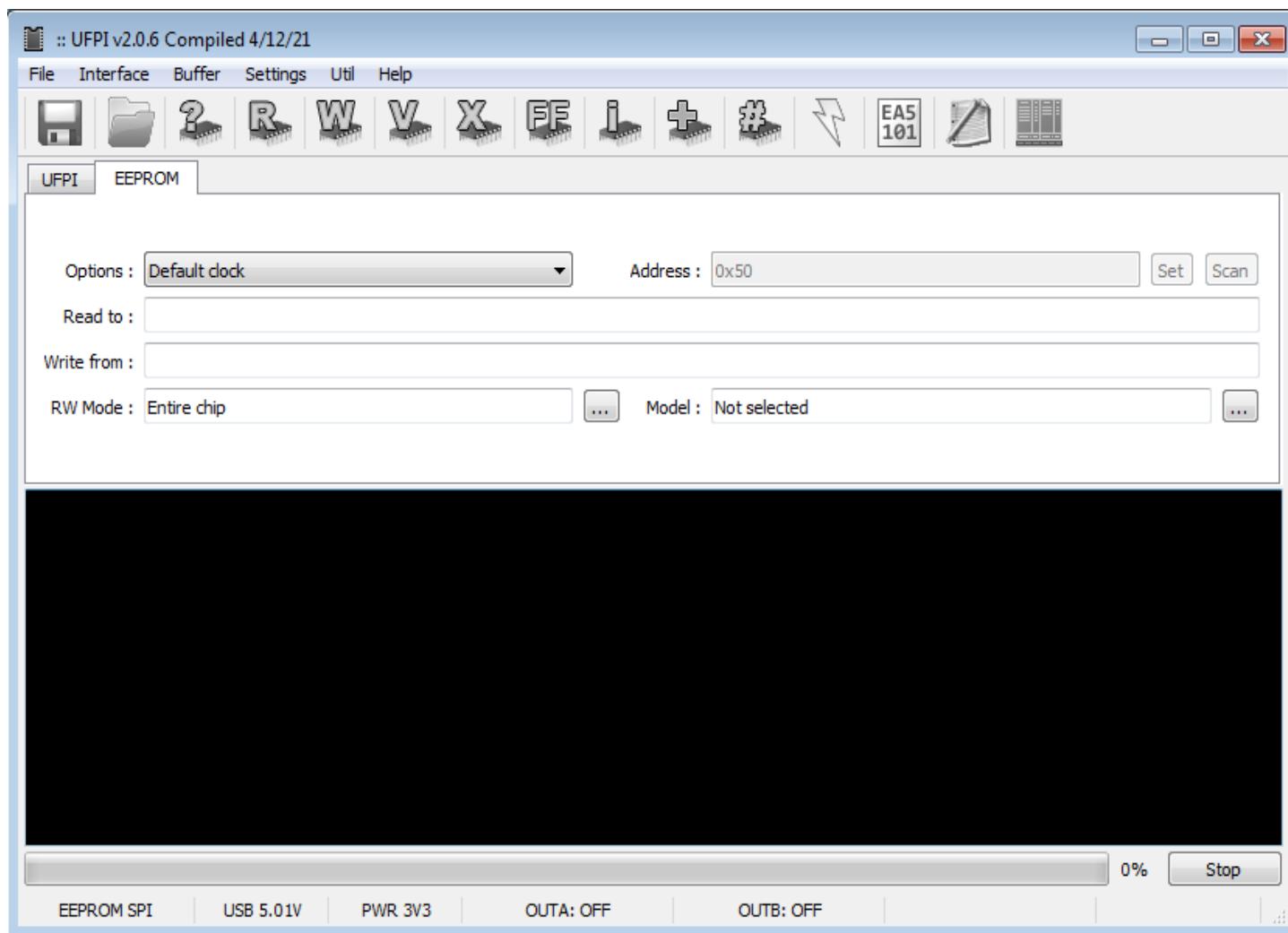
Features.

- In addition to the usual read/write functions, it has a unique method for automatically determining the size of EEPROM 24cXX by pressing the ID read button <? >.
- • 24cXX chips do not require erasure before writing, the erase button in this module writes 0xFF to the entire volume of the chip.
- • Frequency and address for normal 24sXX are determined automatically and do not require changes.
- • Ability to work with individual regions of the chip through [RW Mode](#).
- Ability to work with scripts Chip regions sets by RW Mode.

Address.

In this module is possible to set the arbitrary address of the chip bus. To do this, in the "Address" field, enter a known or received in the log address when scanning and click Install. To scan the addresses, click on the "Scan" button, the log will show the addresses to which the answer was received.

4.7 EEPROM SPI.



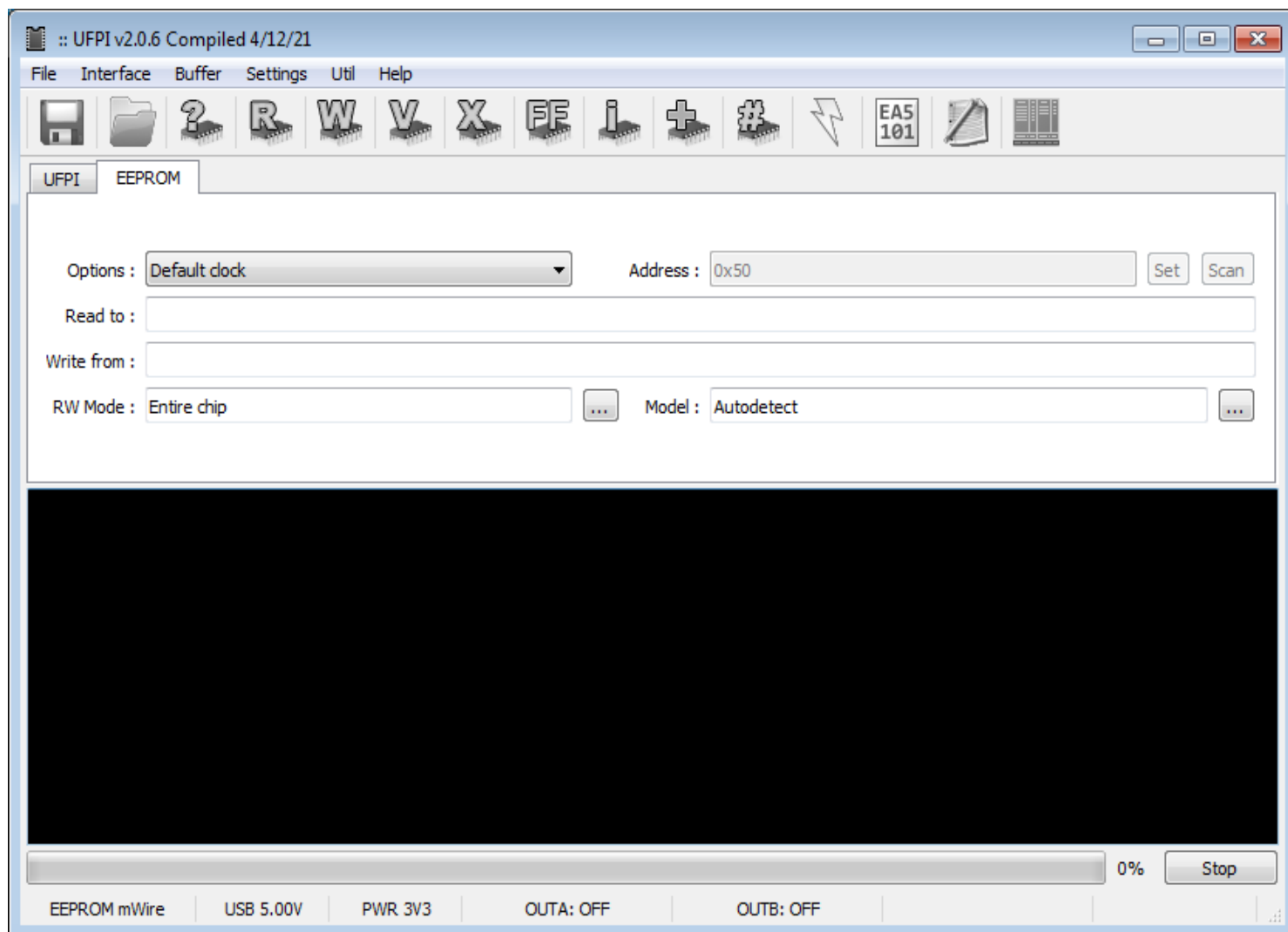
Sockets.

Works with sockets [2046](#) or [2060](#).

Features.

- Designed to work with old 25XXX/95XXX chips. These chips do not have an auto-detection function and you must select a model from the list before working with them.
- Operation is similar to the usual [24cXX EEPROM](#).

4.8 EEPROM Microwire (3-Wire).



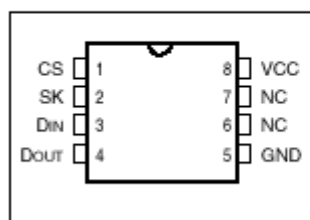
Sockets.

Works with sockets [2046](#) or [2060](#).

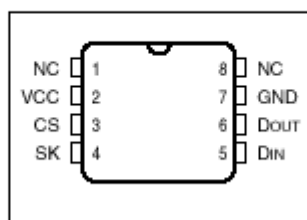
Features.

- Work the same as with [24cXX EEPROM](#)
- 93cXX in addition to voltage and discharge, it can be distinguished by pinout. This should be paid special attention., [check when connect](#).

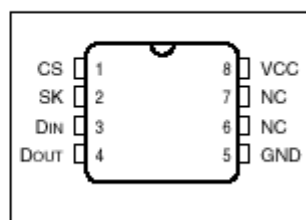
PIN CONFIGURATION
8-Pin DIP



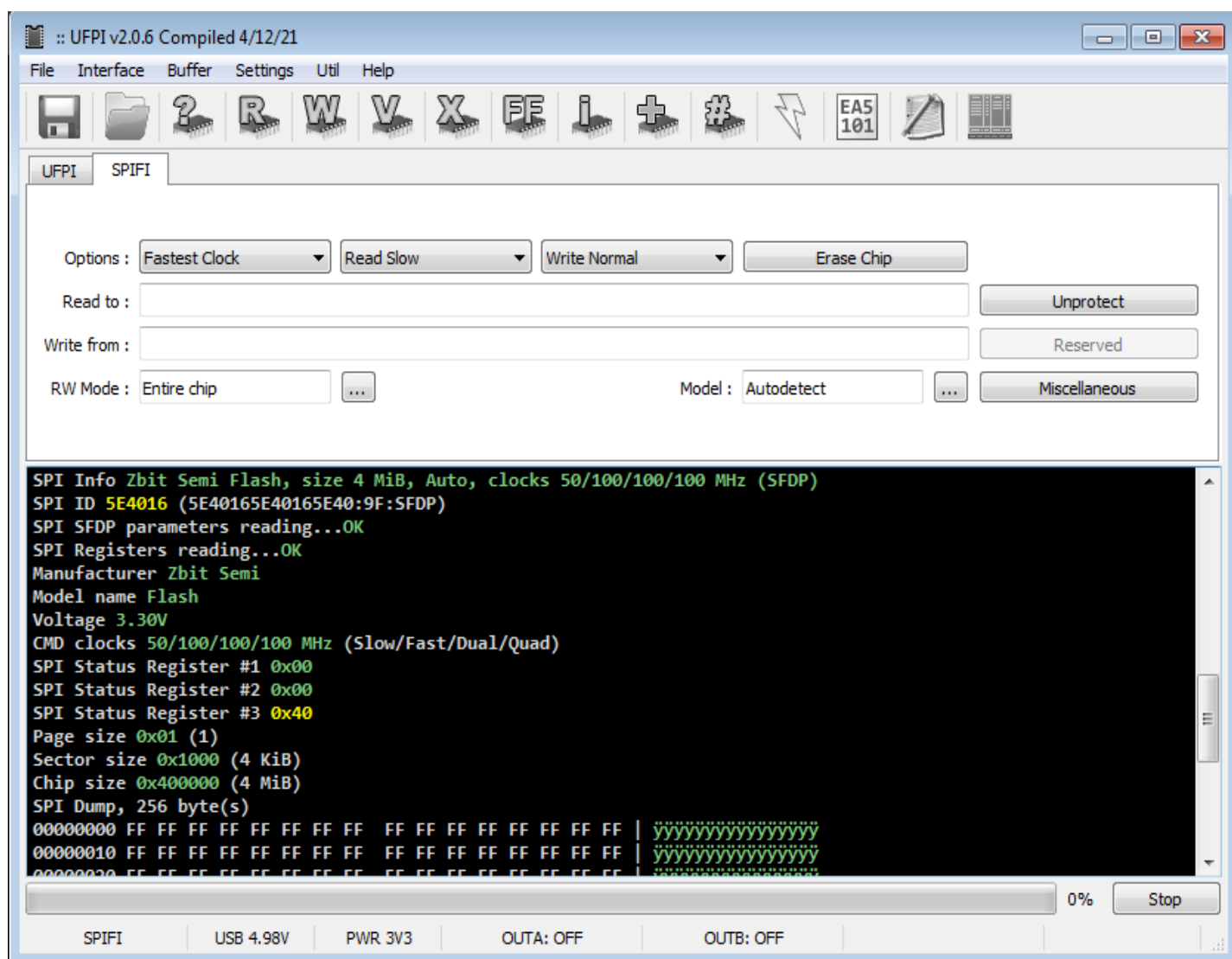
PIN CONFIGURATION
8-Pin JEDEC Small Outline "G"



PIN CONFIGURATION
8-Pin JEDEC Small Outline "GR"



4.9 SPI Flash.



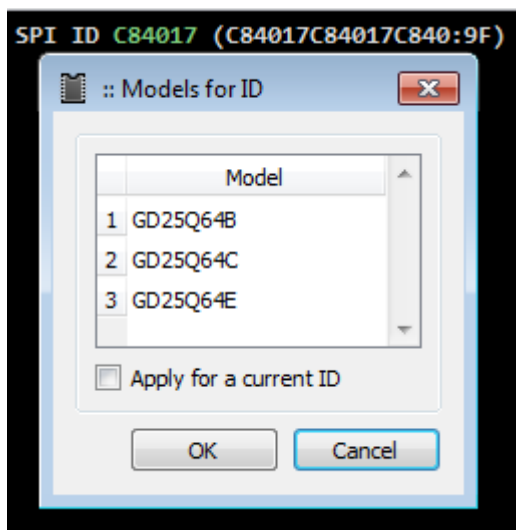
Sockets.

Works with sockets [2046](#) or [2060](#).

Features.

- The module is versatile and does not require the presence of a chip model in the base. For unknown chips, the settings from ID/SFDP/CFI are generally applied automatically. This is usually always enough to read/unlock/erase and write.
- You can [configure manually with button <<+>>](#).
- Allowed working with scripts.

ID reading.



If the read ID matches several different chips, a selection window is displayed to refine the desired model. Select the desired one and click OK. If you want to read the same or the same chip ID multiple times during operation, you can check "Apply to current ID." Then each reading of the ID will apply the selected model and there will be no prompts until the ID is changed (for example, when installing another chip), the program is restarted, or you select "Add. User Model Cleanup» Features

When determining chip by priority, the parameters "configuration file" - "internal base" - "SFDP" - "ID size based on JEDEC standards" are used. If the chips ID is not present in the configuration files or database, the ID log is highlighted in yellow colour

Options:



Frequency set - Values are displayed in the log when reading the "CMD frequency 50/100/100/100 MHz (Slow/Fast/Dual/Quad)" ID. Low/Medium/Maximum/ISP options.

Read mode - Slow/Fast/Dual/Quad. Corresponds to Normal/Fast and Dual/Quad reading modes (some modes may not be supported by flash) according to the chip specification.

Chip Erase - Erase a chip with one command, if it is supported. If you erase using the X button, it will take longer for the entire chip. But it is divided based on compatibility with individual chip blocks.

Unprotect.

The button sends the unlock command, or sets the status registers that prohibit writing to 0. I.e. The recording ban is removed.

Need to know that in some chips these status registers are protected from zeroing and the recorded lock is no longer removed.

You need to know that some old SPI flash have lock registers that are restored if you remove power from the chip. In this case, you need to turn the power on «Always ON».

ACCESS MODE.

Module work is the same as in [others](#).

MODEL.

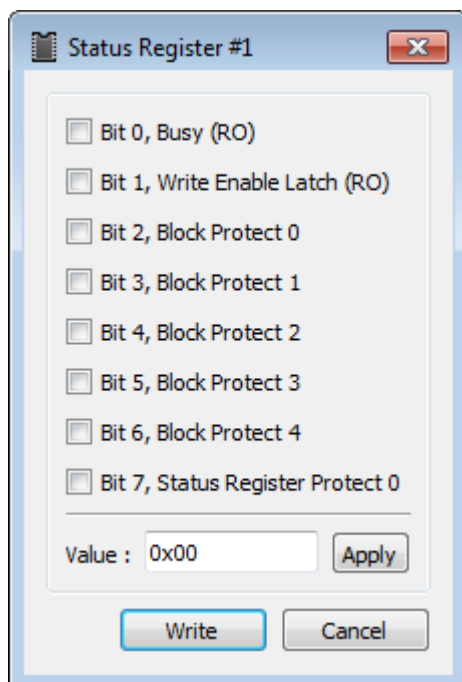
Manual select. AUTO is working by default.

Add.features/Miscellaneous . (some of them appears after ID reading)

The menu may display additional items depending on the current chip model.

Status Register #1/#2/#3...

Depending on the chip model, the available status registers are displayed. All work with manual setting of SPI status register values goes through this menu.



The values are set either by setting the check boxes or by entering a hexadecimal value (do not forget to click the "Apply" button).

All registers in the log (when reading the ID, if detailed output of information is included) and in the installation window are signed according to the documentation for the chip.

When you click the Record button, the recorded status is recorded and reconciled. If the log does not match, the warning is highlighted in yellow colour.

Security Registers (OTP).

Additional registers may contain user data or security registers. If there are such user registers in the chip, the log (when reading the ID, if detailed information output is enabled) will describe its presence and size (for example, "SPI Security Registers 768 bytes" or "SPI OTP 64 bytes"). From this menu, you can select read/write to file/buffer or erase if supported by the chip.

Clear User Model.

Described earlier in ID reading [chapter](#).

SFDP mode.

Default mode. The definition will work as [described](#).

Off. The SFDP check item will be skipped.

«SPI mode», «Feedback clock» и «Data sampling»

These parameters apply to protocol standards and modes. All SPI flash will be in the Default state. These parameters can be useful if you want to change the SPI mode when working with scripts.

Do not write blank data.

Skipping empty pages during recording. Enabled by default as the most common and correct.

Verify during writing.

Automatically, in parallel, during the recording process, reconciles the recorded. The difference with the individual verification after recording is that if there is a check error of the recorded block, the process will stop without waiting for the recording to finish.

Blank check during Erase.

Similar to the previous item, but cleanliness is checked during erasure.

Async IO.

Speed increase [the same](#) as in other modules.

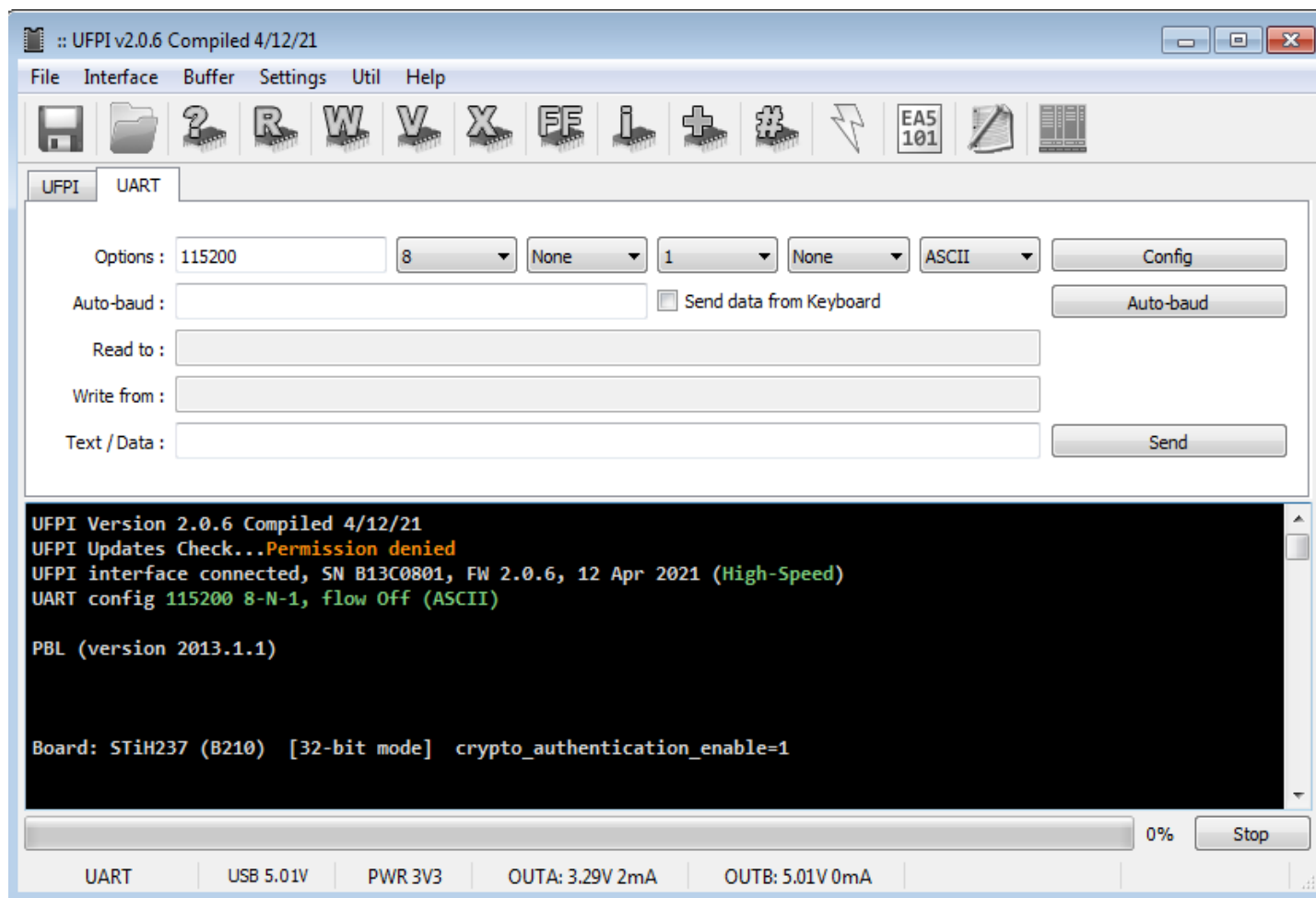
Pin Test

Works the same as other [modules](#).

Configuration files creation

Described in [attachment](#).

4.10 UART.



Socket.

Works with socket [2044](#) and [2060](#).

Features.

- Auto Rate Detection. (baud rate).
- Support working with scripts.

Main features

Intuitive parameters, as in all UART terminal programs:

- UART speed. There is a generally accepted number of standard speeds: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 baud.
- Number of data bits (usually 8).
- Presence of parity bit.
- And 1 stop bit.
- Flow control.
- Type of information displayed in the log (HEX/ASCII).

The picture shows the configuration of UART 9600 8-N-1, stream Off (ASCII). If you select the desired configuration, click Install to apply the settings. The log will write the applied configuration.

Auto-baud.

Autodetect baudrate. It is desirable to start before the start of the exchange for proper synchronization with the starting bit. If successfully detected, the determined speed will be signed in the "Auto Speed" field.

Send data from Keyboard.

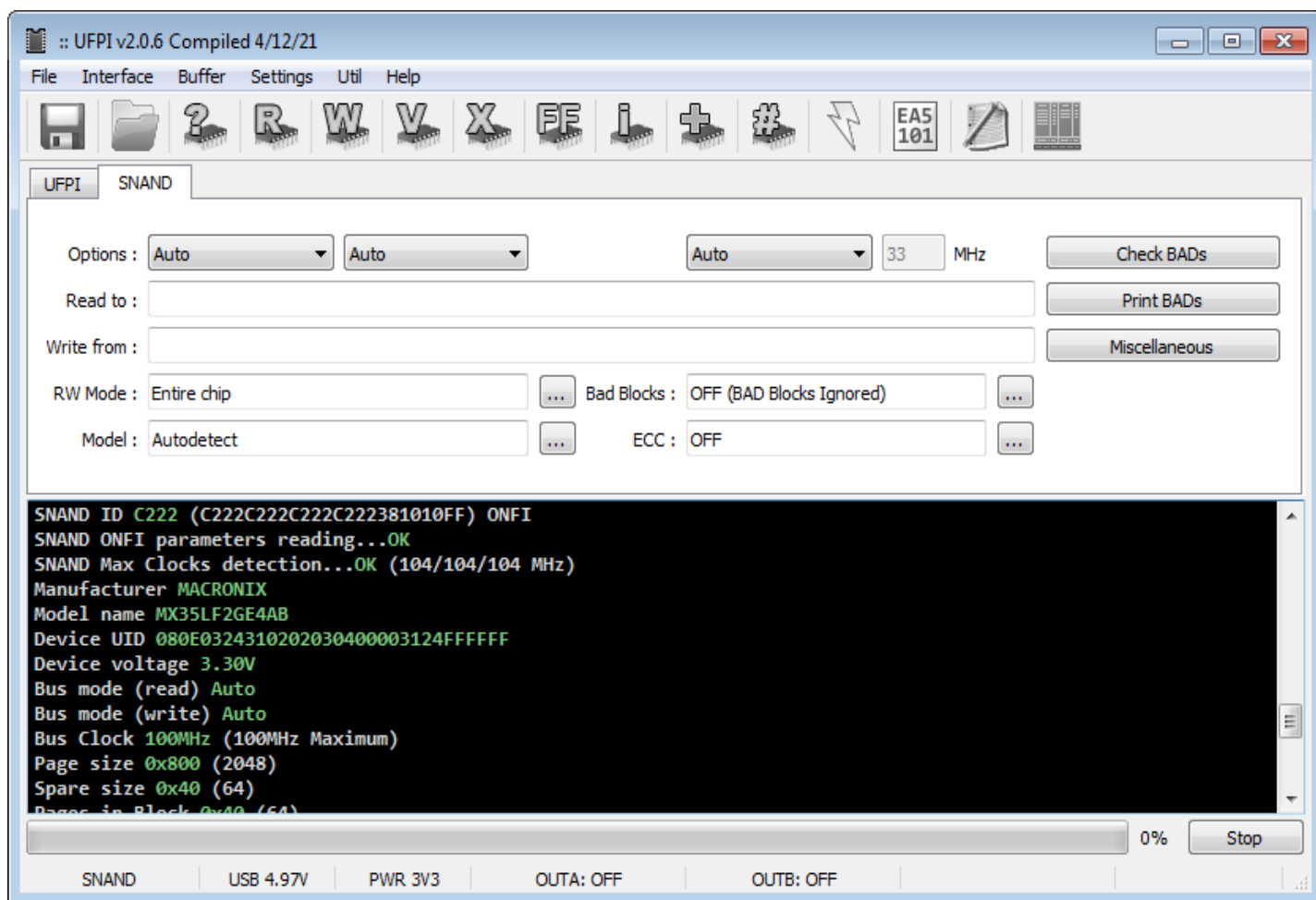
Intercepts keystrokes and sends their code over the TX UART channel.

The Text/Hex field and the Submit button are used to enter and send an entire line. Depending on the format selected in the configuration (HEX/ASCII).

LOG scrolling.

The log's [auto scrolling](#) is customizable

4.11 Serial NAND.



Socket.

Works with sockets [2046](#) or [2060](#). SPI position.

Speed and frequency options.

Options : Auto (dropdown) Auto (dropdown) Auto (dropdown) 33 MHz (input field)

By default, read/write works in single line mode and the default frequency is 85MHz.

You can set the speed by increasing the write, reading modes to x2/x4 and even increasing the frequency to 100MHz, if the chip and current connection allows.

RW Mode

Described in [chapter](#).

Model.

In this module, you do not need to select the model manually, since all the parameters are contained in the ONFI of the flash itself and are applied automatically.

Bad Blocks (BB)

Described in [chapter](#).

Unlike regular NAND, in SNAND WINBOND, BBM LUT can occur. The rest is the same as usual.

"Check Bads" and "Bads info" buttons, as well as in Miscellaneous. The Bad Blocks menu functions work the same as normal [NAND](#).

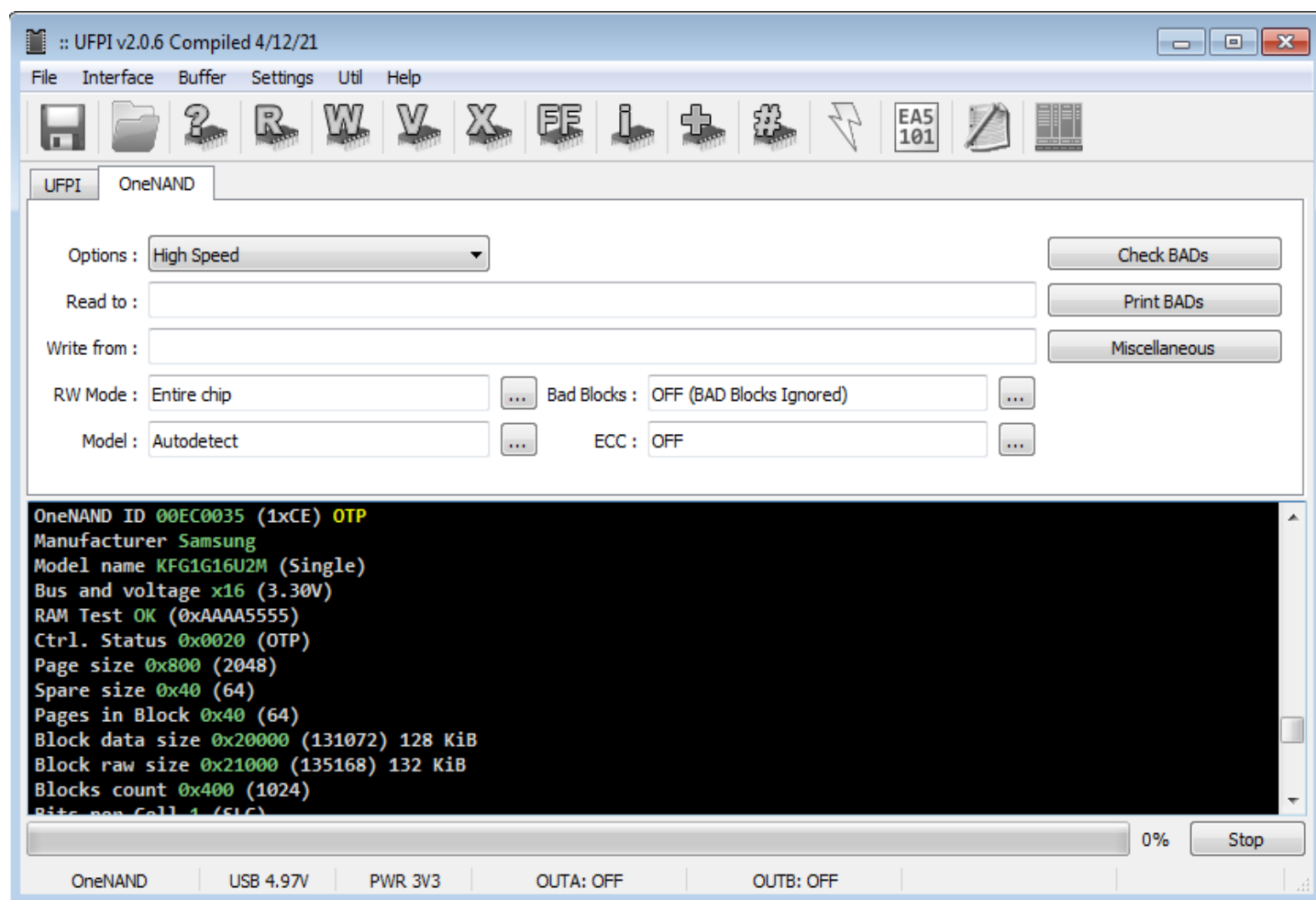
ECC.

Similar to normal NAND, except that internal [ECC](#) control is usually used, which is set as "Chip" (HWECC).

Miscellaneous (add.features)

The filling of functions corresponds to the usual NAND, with the exception of BBM LUT. This item allows you to put the blok into the internal (in the chip controller) [Bad Block Table](#) and process them. The BBM LUT block operation is one-time operation and cannot be canceled. The entered block or table cannot be cleared.

4.12 OneNAND.



Socket.

Works with sockets [2031](#) or [2032](#) and [BGA](#) base.

Features.

Speed selection: Low/Normal/High.

RW Mode.

Described in [chapter](#).

Model.

You do not need to select the model manually, since all flashes are contained in the database and are determined automatically by ID.

Bad Blocks.

Described in [chapter](#).

"Check Bads" and "Bads info" buttons, as well as in Miscellaneous. The Bad Blocks menu functions work the same as normal [NAND](#).

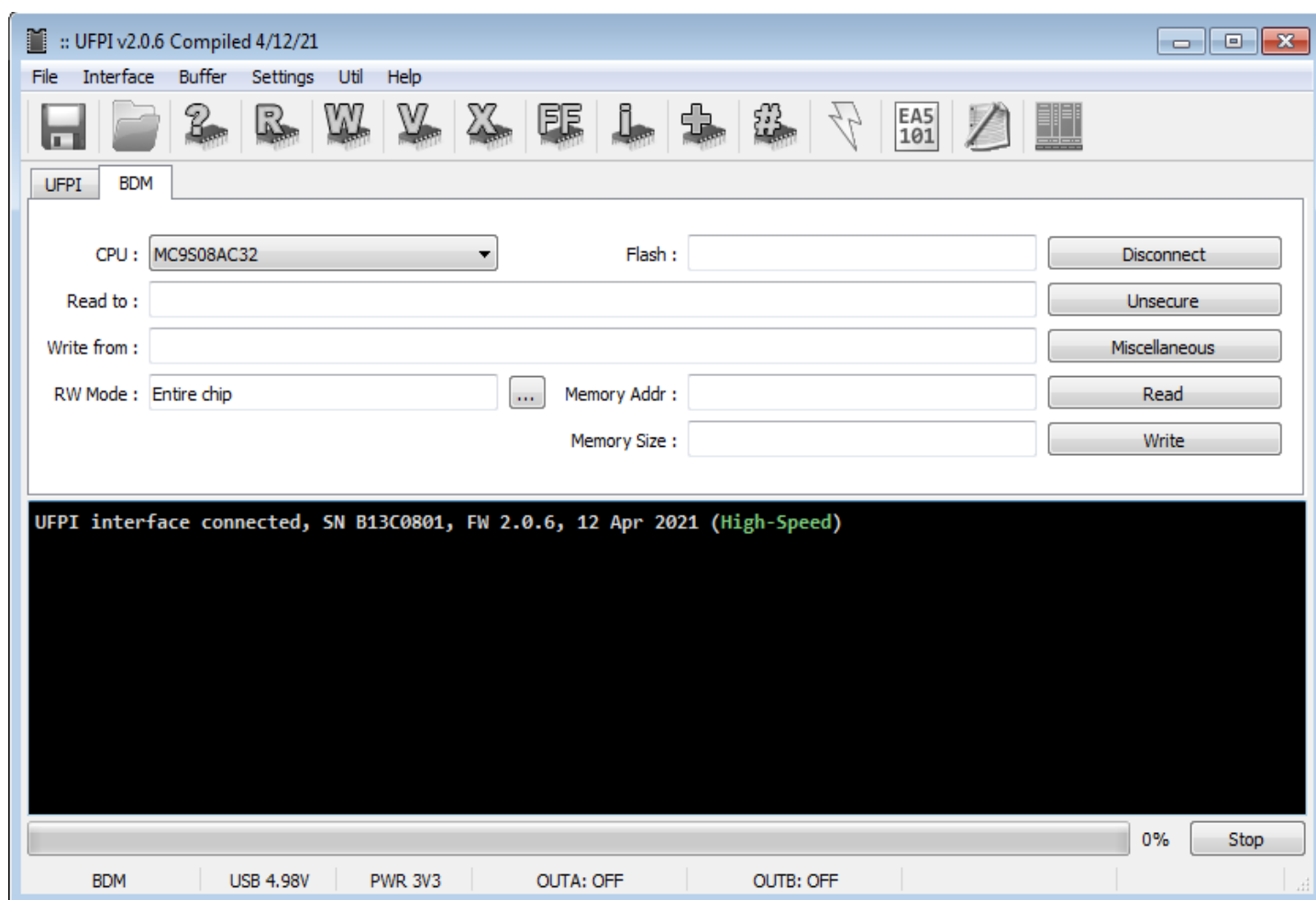
ECC.

The same as with NAND.

Miscellaneous (add.features).

Features corresponds to the usual NAND, with the exception of Flex OneNAND. This item is for switching between chips in the chip housing.

4.13 BDM.



Socket.

Works with socket [2043](#).

Features.

- Can work with scripts.
- Processors supported:

MC9S08AC60

MC9S08AC48

MC9S08AC32

MC9S08AW60

MC9S08AW48

MC9S08AW32

MC9S08AW16

MC9S08GB60

MC9S08GB32

MC9S08GT60

MC9S08GT32

MC9S08GT16

MC9S12XF384

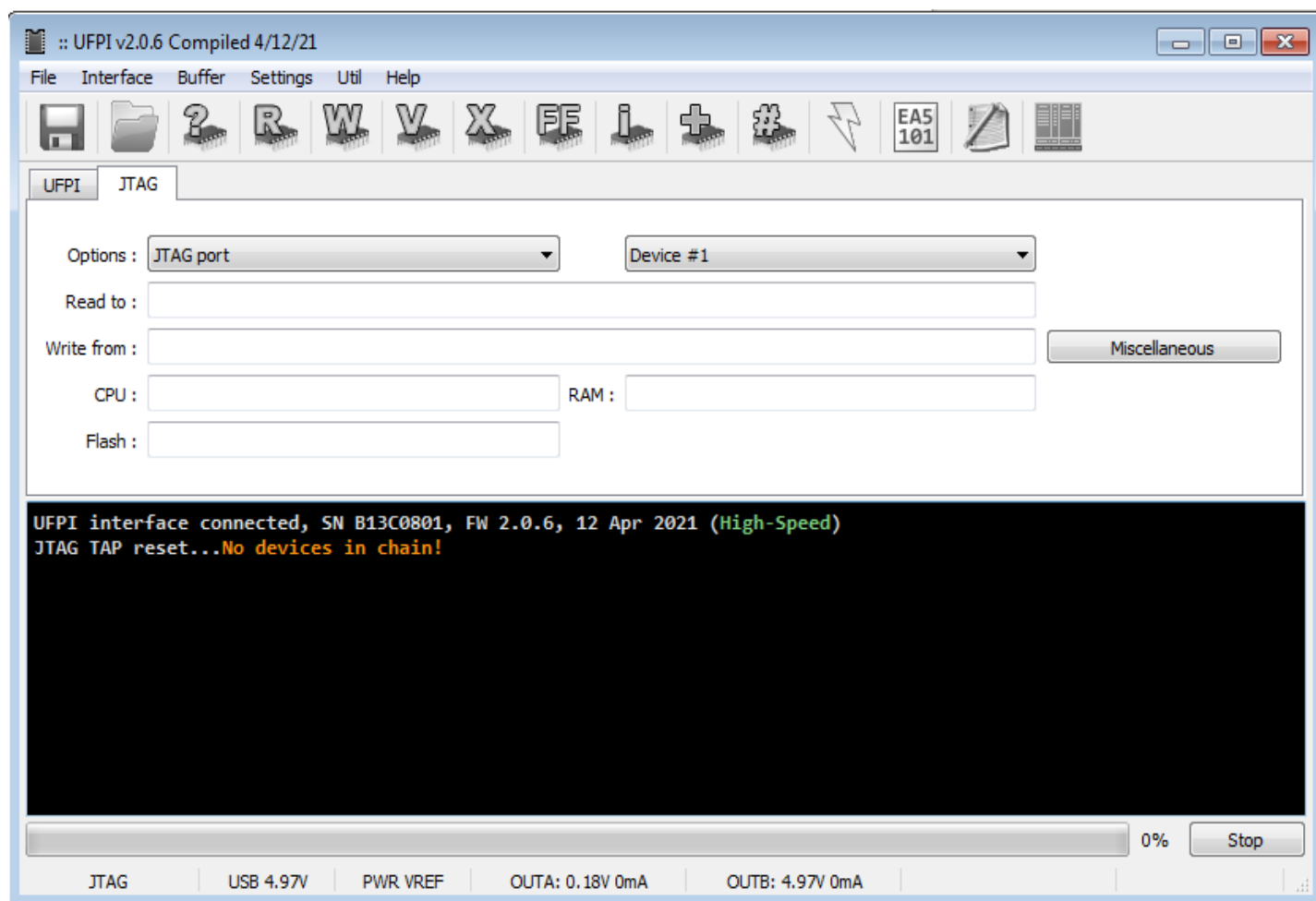
and others.

- Writes with .bin files that can be converted from s-records to "Tools - Conversion" menu.
- More information on [forum](#).

Sequence of actions.

- Select the processor type.
- Read ID.
- Work like with a regular flash drive.
- Erase and remove protection with the "Erase everything" button.

4.14 JTAG.



Socket.

Works with socket [2043](#).

Features.

- This interface creates a .udev file with initialization parameters and chip configuration.
- The VREF level is supplied from an external source according to the signal levels (usually 3.3V).
- After connecting and reading the ID, we open the defined chip tab and work as usual according to the module used.

Supported devices.

At the moment, there are ready UDEV files only for processors S4LJ162X01 and JUPITER4E and their internal flash. How to work with these processors: examples are given in the forum:

[Theme Modes Restore a processor-based printer JUPITER4E](#)

[Restore a processor-based printer S4LJ162X01](#)

It is also possible to run [CMSIS DAP](#) emulator.

4.15 Mount File.

Application.

Allows [attach a file system](#) as a virtual file drive.

Examples is в [attachment](#).

Features.

At the moment, it differs from "Mount Chip" only in that the mount object should be a file. Both modules are basically almost equivalent, the difference is in convenience only. This option may be more convenient if the partitions are read and you want to work with the partitions on the disk before writing to the chip.

4.16 Mount Chip.

Application.

Allows [attach a file system](#) like a virtual disk drive.

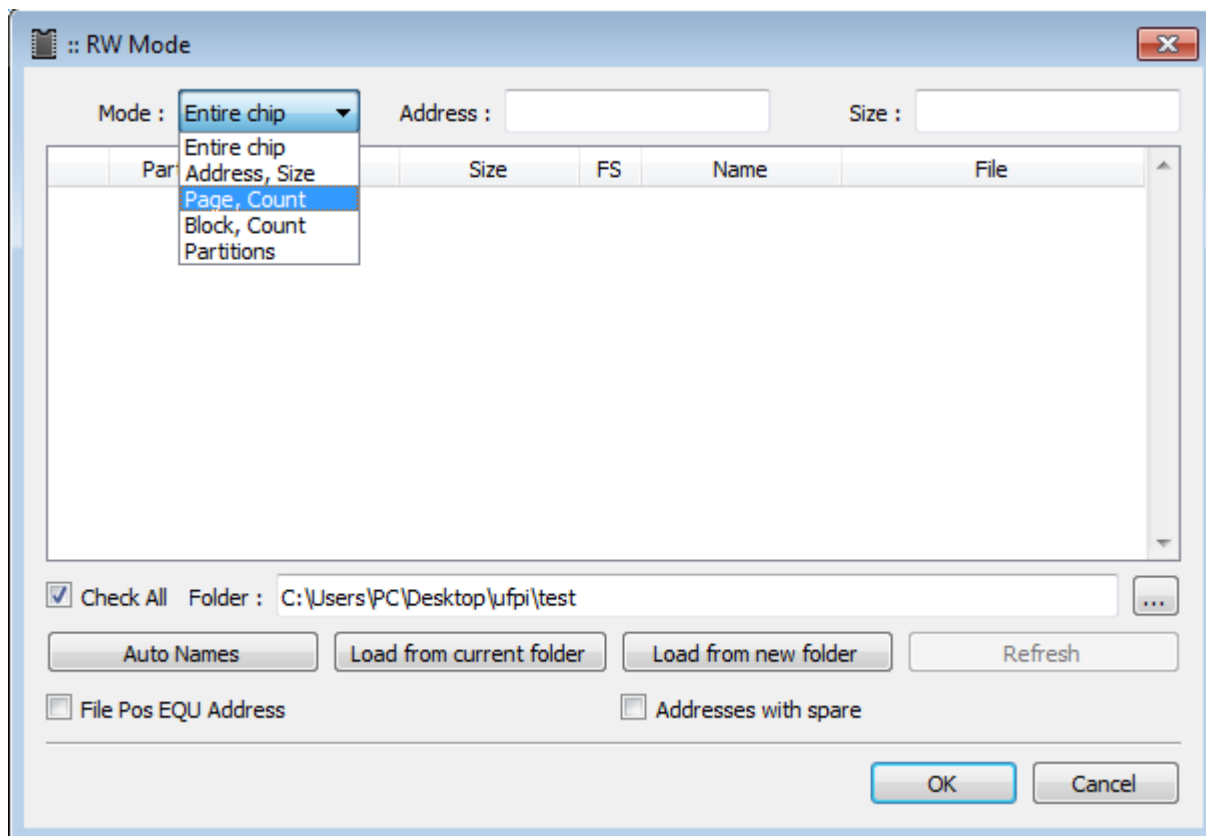
Examples in [attachment](#).

Features.

At the moment, it distinguishes itself from Mount File only by the fact that the installation object should be a chip. Both modules are almost equivalent, the difference is in convenience only. This option can be more convenient if you need to work with partitions without wasting time reading and writing the dump back to disk.

5 RW Mode

Allows you to work with the entire chip, as well as set addresses, page numbers, blocks, partitions separately. It works in almost every module. The following is an example of NAND.



Mode: Address, size.

Size must be entered multiple of pages. Moreover, if the check mark "No spare" is set ON in the menu "Miscellaneous," then the page size must be entered without spare. Otherwise, with spare.

You must enter an address that does not include spare if the check mark is not set to "Addresses with spare" in "Miscellaneous". You can enter in decimal and hexadecimal.

Mode: Page, count.

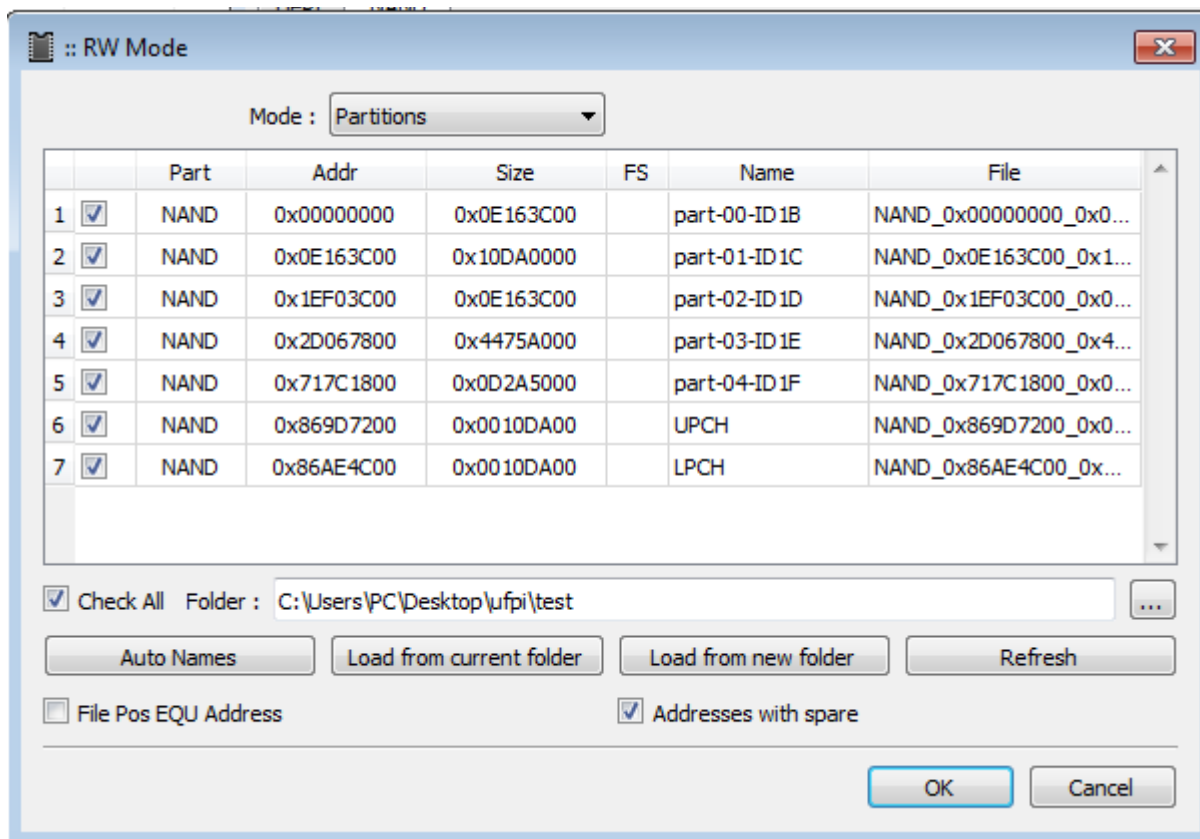
You must first read the ID and define the chip geometry. Enter the start page number of the desired region and the number of pages. *Need to know that the minimum erasure size of NAND is the block size!*

Mode: Block, count.

The same as with pages, but dimensions in blocks.

Mode: Partitions

In this mode, you can work with chip partitions. Partitions can be loaded from a UDEV file, filled manually, or get using dump analysis.



All fields within the table can be edited by double-clicking. *Partition files are written and saved to the specified folder. Files in "Read To" and "Write From" are not used in this mode! When you add a keyboard to a folder path in a row, you must press Enter to apply it.*

Folder : C:\Users\PC\Desktop\ufpi\test

To select the required files/regions (the names are automatically assigned from the name and location of the section unless otherwise specified) that you want to read/write, check the appropriate lines

Right-click menu

	Part	Addr	Size	FS	Name	File
1	<input checked="" type="checkbox"/>	NAND	0x00000000	0x0E163C00	part-00-ID1B	NAND_0x00000000_0x0...
2	<input checked="" type="checkbox"/>	NAI		0	part-01-ID1C	NAND_0x0E163C00_0x1...
3	<input checked="" type="checkbox"/>	NAI		0	part-02-ID1D	NAND_0x1EF03C00_0x0...
4	<input checked="" type="checkbox"/>	NAI		0	part-03-ID1E	NAND_0x2D067800_0x4...
5	<input checked="" type="checkbox"/>	NAI		0	part-04-ID1F	NAND_0x717C1800_0x0...
6	<input checked="" type="checkbox"/>	NAI		0	UPCH	NAND_0x869D7200_0x0...
7	<input checked="" type="checkbox"/>	NAI		0	LPCH	NAND_0x86AE4C00_0x...

Add partition.

When creating, you can add a new partitions to the end of the table.

Remove Partition(s).

Delete the selected partition. You can select multiple through Ctrl or Shift.

Open File.

Opens the dialog for selecting an existing region file to write to in the selected row.

Set File Name.

Opens a dialog box to specify the location and file name that will be created during reading, if it does not exist, or overwritten.

Create Empty Files.

Creates empty files with the specified name and zero size in the working directory from the list. If files exist, they are overwritten with zero sizes.

Save to UDEV file (UTF-8).

Creates [UDEV](#) file according to the compiled partitions in the selected folder.

Addresses with spare.

Enter sizes and addresses when the check mark is set ON, according to the size of chip (as in the dump). And the output in the log of addresses in this case will also be taking with spare and highlighted.

File Pos EQU Address.

If the check box is selected ON, the selected region will be written from the full dump. If you want to use it in a read operation, the file must be of the appropriate size.

Load from new/current folder.

Allows you to load pre-read regions with appropriate names from the folder. To work with partitions and files at once, you must name the partition files "by convention" boot1_0x0_0x1000_zboot.bin and click on the "Load from folder" sections in the RW Mode.

Mode : Partitions

	Part	Addr	Size	FS	Name	File
1	<input checked="" type="checkbox"/>	NAND	0x00000000	0x0E163C00	part-00-ID1B	NAND_0x00000000_0x...
2	<input checked="" type="checkbox"/>	NAND	0x0E163C00	0x10DA0000	part-01-ID1C	NAND_0x0E163C00_0x...
3	<input checked="" type="checkbox"/>	NAND	0x1EF03C00	0x0E163C00	part-02-ID1D	NAND_0x1EF03C00_0x...
4	<input checked="" type="checkbox"/>	NAND	0x2D067800	0x4475A000	part-03-ID1E	NAND_0x2D067800_0x...
5	<input checked="" type="checkbox"/>	NAND	0x717C1800	0x0D2A5000	part-04-ID1F	NAND_0x717C1800_0x...
6	<input checked="" type="checkbox"/>	NAND	0x869D7200	0x0010DA00	UPCH	NAND_0x869D7200_0x...
7	<input checked="" type="checkbox"/>	NAND	0x86AE4C00	0x0010DA00	LPCH	NAND_0x86AE4C00_0x...

Check All Folder : C:\Users\PC\Desktop\ufpi\test

Auto Names Load from current folder Load from new folder Refresh

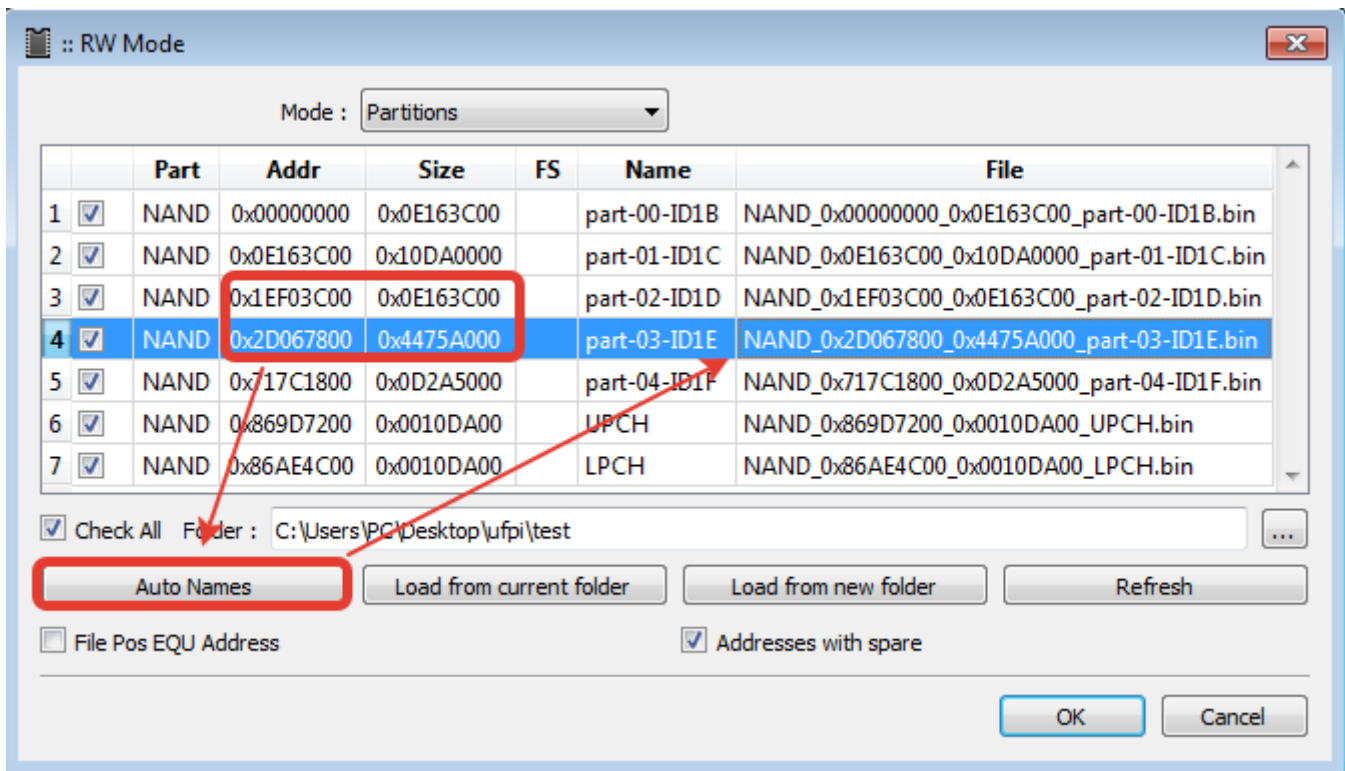
File Pos EQU Address Addresses with spare

test

Добавить в библиотеку Общий доступ Новая папка

Имя	Дата изменения	Тип	Размер
NAND_0x00000000_0x0E163C00_part-00-I...	14.06.2021 23:42	Файл "BIN"	230 799 КБ
NAND_0x0E163C00_0x10DA0000_part-01-...	14.06.2021 23:43	Файл "BIN"	276 096 КБ
NAND_0x1EF03C00_0x0E163C00_part-02-I...	14.06.2021 23:43	Файл "BIN"	230 799 КБ
NAND_0x2D067800_0x4475A000_part-03-I...	14.06.2021 23:45	Файл "BIN"	1 121 640 КБ
NAND_0x86AE4C00_0x0010DA00_LPCH.bin	14.06.2021 23:46	Файл "BIN"	1 079 КБ
NAND_0x717C1800_0x0D2A5000_part-04-...	14.06.2021 23:46	Файл "BIN"	215 700 КБ
NAND_0x869D7200_0x0010DA00_UPCH.bin	14.06.2021 23:46	Файл "BIN"	1 079 КБ

Auto Names.



After filling the table and alignment of sizes and addresses, clicking the button automatically populates the "Files" column. These names will save the region files when you read them.

6 Power supply management.



The left mouse button enables/disables the specified power supply.
Right-click to display the settings menu.

NAND	USB 4.96V	PWR 3V3	OUTA: 3.29V 4mA
------	-----------	---------	-----------------

When power is on, the status line shows the voltage and current consumed by the power supply channel. In SPI, NAND and similar modules, one channel is used for power supply, and for eMMC both channels are involved, and the second channel also shows the voltage and current of the second channel. The main channel on which it is possible to adjust the voltage during operation of the programmer is channel A.

By excess of 500mA current on the channel ([if not disabled](#)) then overload protection will work. The log will show "UFPI interface is off" and on the programmer will start blinking the LED corresponding to the channel that went into an accident ([Channel A- VCC/VREF LED1 or channel B- VUSB LED2](#)).

If "Ignore the accident" is set on the main tab, the box will not stop emergency, it will continue to work, but the message will appear: "UFPI power accident! Output A "

It is not recommended to turn on "Ignore accident" during daily operation in normal conditions! Only in some cases can use with an understanding of the consequences when working in-circuit, when it is required to power the connected device from the box.

For ISP operation with large current consumers, you can use the Vref mode. To do this, switch the switches on the socket to the appropriate position and connect the external power supply to the VCC output of the device chip and connect the socket to the VCC. In this case, when voltage is supplied from an external power source (for example, LDP), the internal output buffers of the box will be powered and the voltage will be displayed in the status line.

The remaining power voltages set by the switches on the socket are described in the [PIN OUT](#).

Right-click on "[zipper button](#)" the menu appears, all check boxes are cleared by default (you can also click "Default voltage" to undo the changes). The voltage when reading the ID is used according to the socket, then after reading the ID, its model is determined and/or the corresponding voltage is set automatically.

Example:

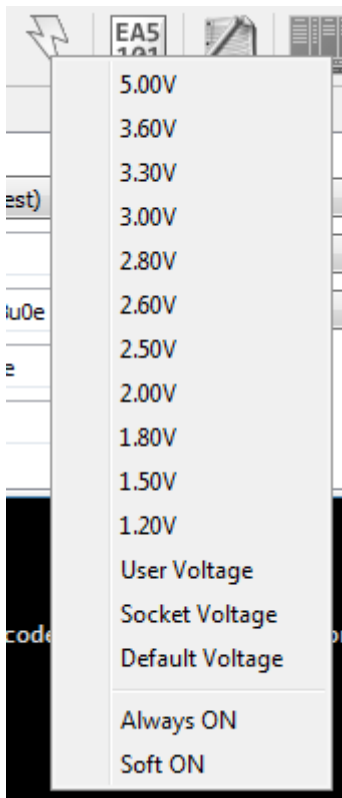
after reading the ID at NAND512R3A in a socket with a set voltage of 3.3V, the program will automatically determine it as 1.8 volt and work with it on 1.8V.

after reading ID at W25Q64FW with 3.3V voltage sets on a socket, we will see in the log that was defined as W25Q64FV as they have absolutely identical ID!!! If you set the voltage on the socket to 1.8v and read the ID again, then the W25Q64FW will already be written in the log. All according to the voltage.

The setting is individually remembered in each module!

You can set any of the listed voltages (the state of the socket power switches should be different from Vref), or in the "User voltage" item you can manually enter numbers in millivolts. If the voltage is set through the menu, change it by the switch on the socket, then it will change in the same way as the voltage is set on the socket.

The voltage on the menu does not prevent it from being put on the socket.



The "Socket Voltage" option switches from the current voltage to the voltage set on the socket.

The "Always ON" option can be useful when working in ISP mode, with scripts, old SPIs (which do not retain status after power off). In the eMMC module, power is always ON by default and this option does not need to be activated.

The "Soft On" option allows you to smoothly raise the voltage when switching on, which often helps with ISP connection, preventing protection from working during the initial charge of capacitors and transients when turning on power from the box.

7 .UDEV devices files.

Application and Features.

UFPI can work with device files (.UDEV). Device files are text files with their own format (similar to INI), which contain the information necessary to work with your device for each particular module. This is the name, processor type, memory address, initialization strings, partition data, etc.

UDEV can be opened using hot keys. In the [INFO] section, you can write any information that will be displayed when the UDEV is loaded. The encoding UTF-8, as well as highlight by using the tags # C1, # C2...

UDEV can be compiled in a regular "notebook", or in scripts (for example, PARTITION FINDER) or in the "Partitions" RW Mode, through the right mouse button menu.

Structure.

The example of NAND for TV SAMSUNG D5500.

```
[DESC]

; Model device
Name = D5500

; Memory type
; eMMC, SD, NAND, ONENAND
FlashType1 = NAND

; Base addr, used for JTAG etc.
FlashBase1 = 0x0

; ECC algoritm
; 0 - OFF
; 1 - Scheme
; 2 - HW
ECCAlgo = 1

; ECC Scheme
ECCScheme = MSTAR_P8K_SP436_CW8_S12L42

; Analysis on ID
; 0 - OFF
; 1 - dump
; 2 - dump, ic
; 3 - ic
; 4 - ic, dump
AnalysisOnID = 1

[BBM]

; Erase Bad Blocks
bbErase = false

; Detection Bad Blocks
; 0 - OFF
```

```

; 1 - During Erase
; 2 - 1st spare marker byte
; 3 - 6st spare marker byte
; 4 - User marker offset in spare
; 5 - User marker offset in page
; 6 - BAD Blocks Table
; 7 - BBM LUT (SNAND)
bbDetect = 2

```

```

; Management
; 0 - Not used
; 1 - Skip BAD Block
; 2 - Use Reserved Area
bbMgmt = 2

```

```

; Marker byte value
bbMarkerVal = 0x00

```

```

; Operation
; 0 - equal
; 1 - not equal
bbMarkerOp = 0

```

```

; User BAD Block marker position
bbMarkerPos = 0

```

```

; Table type
; OFF - Not use
; AUTODETECT
; LG BBMINFO
; Samsung RFS
bbTableType = Samsung RFS

```

```

; Номер блока таблицы плохих Blocks
bbTableBlock = 0

```

```

[NAND]
;offline geometry NAND. Use for mount file and analyse Bad Block.
PageData = 8192
PageSpare = 436
PagesInBlock = 128
Blocks = 2076

```

```

[INFO]
; Any text information that is displayed at Load UDEV

```

```

[PARTITIONS]
; Information for working with partitions

```

UDEV partitions example for eMMC

```

[DESC]
Name = Partitions
FlashType1 = eMMC

```

```
FlashBase1 = 0
```

```
[PARTITIONS]
```

```
;Activate the RW mode "Partition".
```

```
PartitionsMode = true
```

```
;Addr partition, Size artition, Name partition, Part/Partition(user/boot1/boot2/TPM), File name, Offset  
in file, File System.
```

```
0x00000000,0x00200000,Partitions_table,USER,
```

```
0x0000200000,0x00300000,MBOOT,USER,
```

```
...
```

```
0x0003080000,0x00040000,RTPM,USER,
```

```
0x00030C0000,0x2BC00000,system,USER,USER_0x00030C0000_0x2BC00000_system.bin,0,EXT4
```

```
0x002ECC0000,0x40400000,userdata,USER,,,EXT4
```

To work with partition and with files, it is necessary to name the partition files by names "by convention" type: boot1_0x0_0x1000_zboot.bin and press the "Load from folder" sections in RW Mode.

UDEV partitions example for NAND

```
[PARTITIONS]
```

```
PartitionsMode = true
```

```
; if the addresses of the partitions are specified including spare.
```

```
RawAddrMode = true
```

```
; if the sizes of partitions are including spare.
```

```
RawSizeMode = false
```

```
0x00000000,0x1950000,UBOOT
```

```
0x01950000,0x1734000,UBOOTENV
```

```
0x03084000,0x1B4A4000,SYSTEM.UBI
```

```
;0x00000000 - Address partition
```

```
;0x1950000 - Size partition.
```

```
;UBOOT - Partition name
```

Examples of working with partitions can be seen in [attachment](#).

8 GZIP.

One of the pleasant features of UFPI is working with GZIP (open distributed archive format) format on the fly. This means you can read and write directly from the archive. Thus, we save disk space and do not spend time unpacking dumps.

To read to the archive, you must select a file to save with the .gzip extension.

To write from the .gzip archive, you do not need to unpack it, simply select the archive and click "Record." The programmer will determine the format itself, and record its content.

You can pack and unpack not only dumps, but also any files.

You can find the compression ratio in the settings. It can affect the speed of work on weak computers. The default is level 7 as the best.

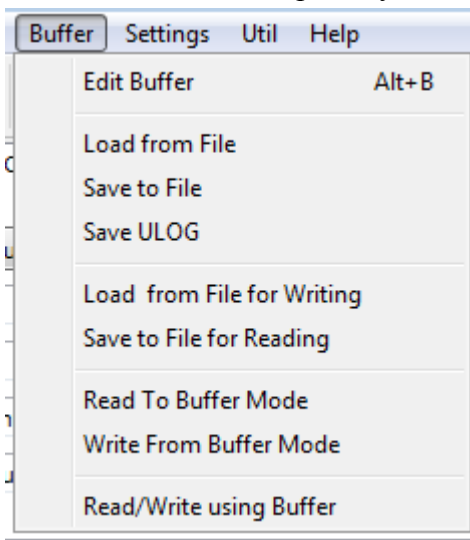
9 Working with buffer.

EAS
101

The buffer allows you quickly evaluate the content, and also you can read, edit and write back to the chip without saving a dump to a file.

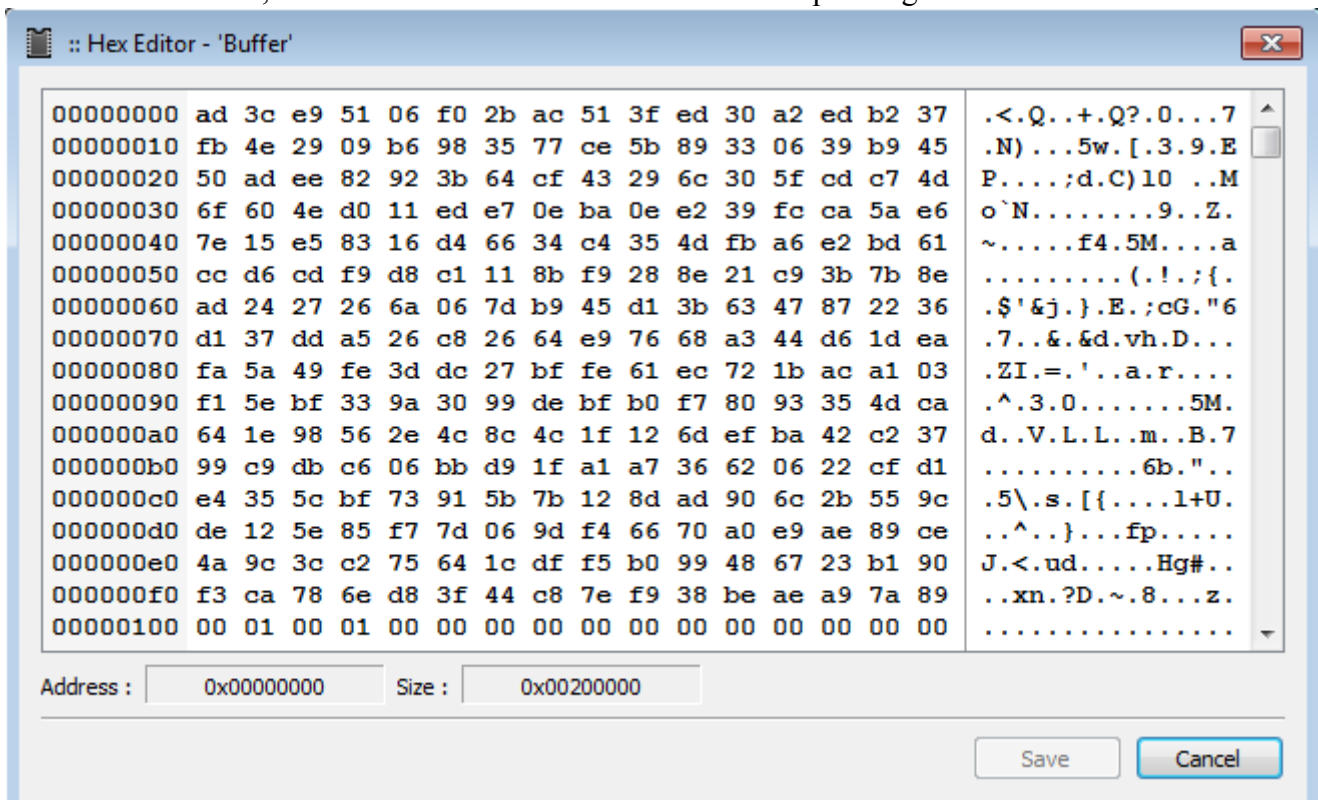
Working with a buffer can be more stable at a high speed than directly with a file on HDD. The buffer is the space allocated in the RAM, that's the reason of the limitations in its size. This is the fastest option To work with **LOGGER**, (after capturing the signal, you can either convert to the .sr format or save it in the native binary format UFPI .ulog).

The Buffer Editor opens by a button on the toolbar, or through the Edit Buffer menu.



The Buffer Read Mode and Buffer Write Mode flags can be set and cleared either individually or simultaneously by the "Buffer Read and Write Mode" option

At the same time, the field "Read to" or "Write from" corresponding to the activated item becomes inactive.



The editor allows you to view, edit, and save. Standard Hotkeys Ctrl+C, Ctrl+V and others work as well.

Options : Autodetect Auto Clock 25 MHz

Read to :

Write from :

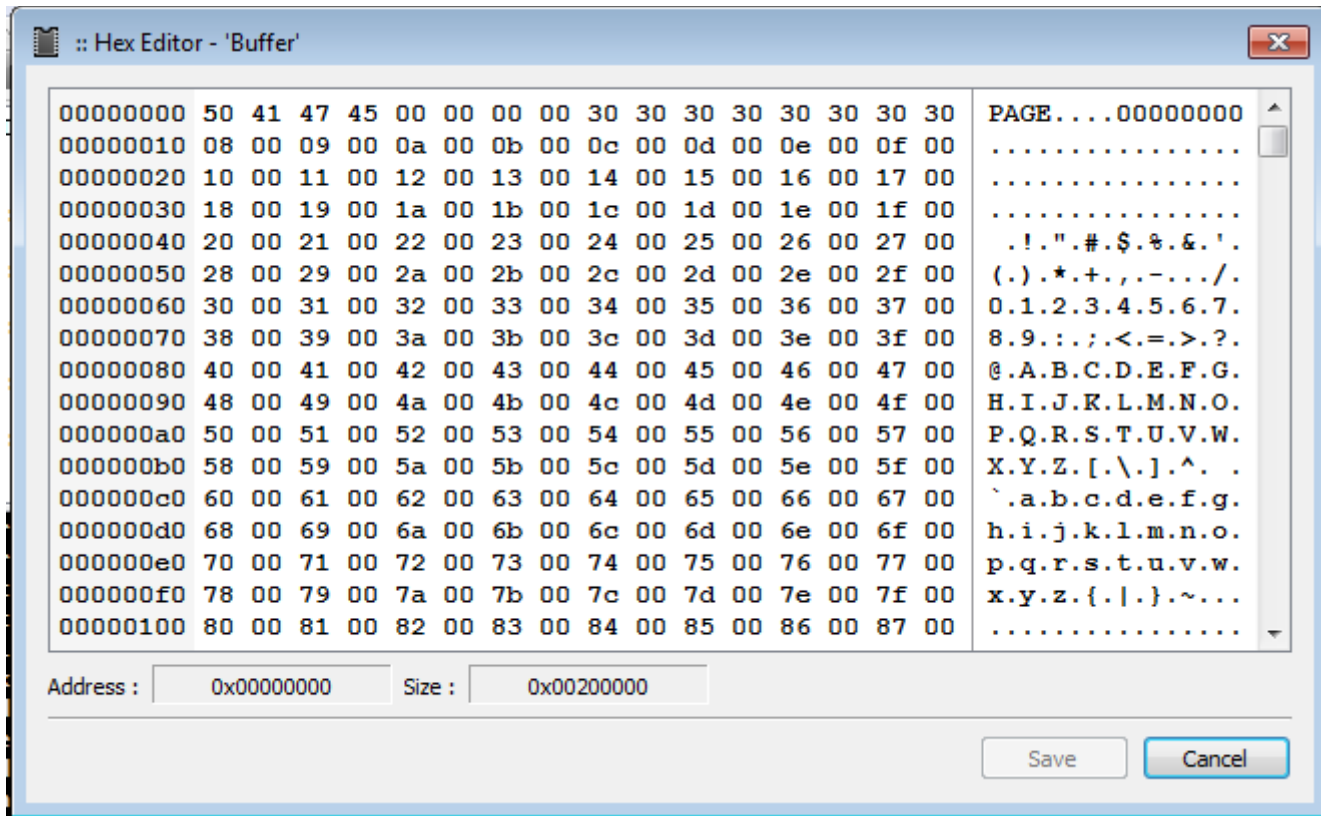
DW Mode : Entire chip Partition :

Buffer size is limited and cannot be more than 1.5 GB.

10 Test FILE

The test file will generate a visually understandable sequence in the dump.

A test dump is generated individually for each module and each chip based on geometry and dimensions and other options.



Visually, the editor shows the separation of pages, codewords, and even spare with the specified ECC

11 File manager and file system (FS) mount.

Features.

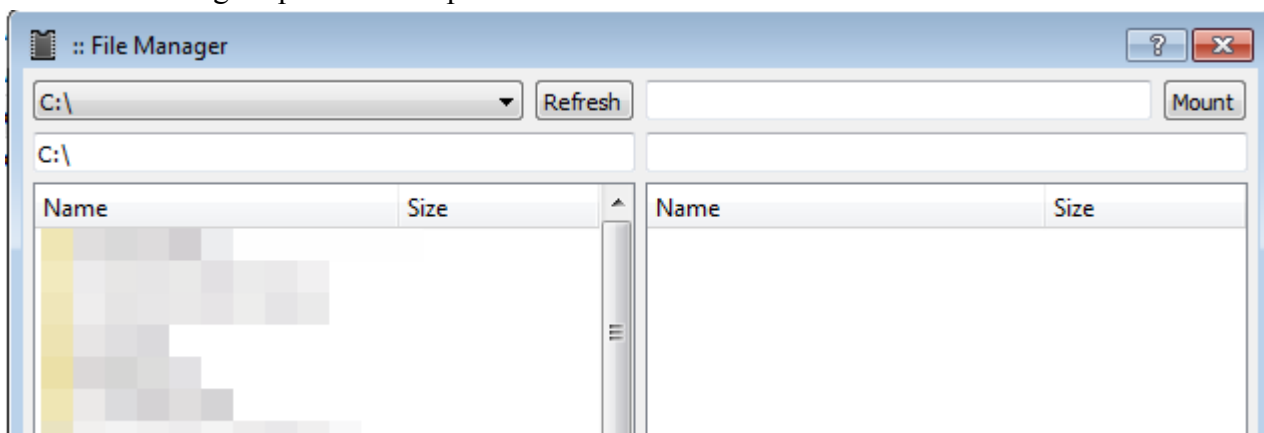
The file manager allows you to mount and work with the file system directly reading and writing by the file system in the file dump or directly in the chip. You can mount both: as partition in a dump and as a separate partition file.

RUN.

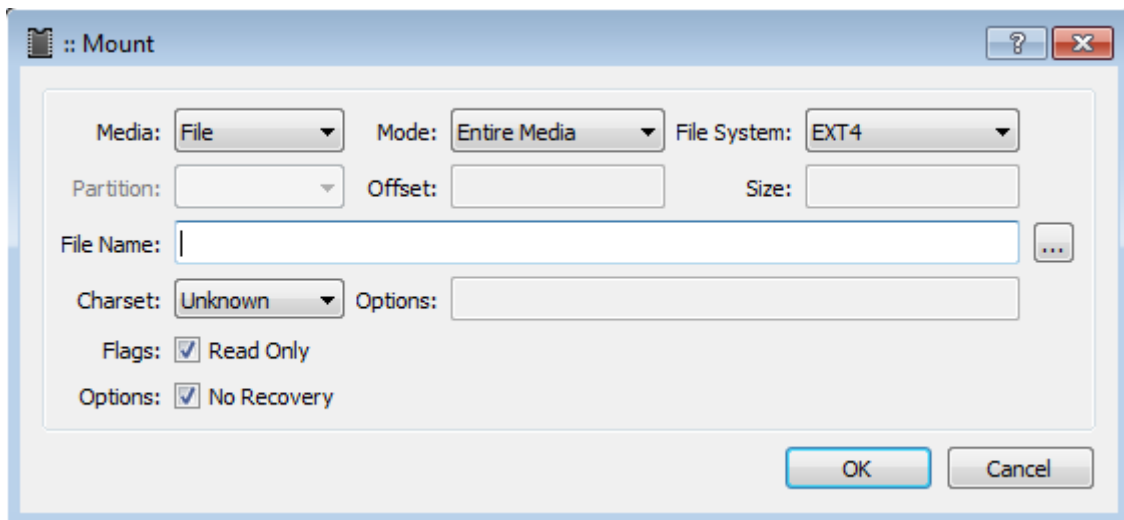
When the programmer is connected, the «START FILE MANAGER» button appears in the toolbar.



The file manager opens at startup.



Click the "Mount" button to mount the partition.



Mount options:

Media - chip/file;

Mode - full file or chip (if the full file or media is a single partition)/part of the media (manually enter the offset and size of the partition in the dump)/partition (by selecting a partition from the list);

File system - selection of the required FS;

Partition - select a partition from the list of chip/dump defined partitions during analysis;

Offset, size - for manual address setting in "partial of media" mode;

File Name - must be specified if you are working with a file;

Working in FILE MANAGER.

Described in [attachment](#).

Examples. (recommended to learn)

[Mount](#).

[Working in FILE MANAGER](#).

Notes.

If the partition is not mounted, need to recheck everything according to the items, restart the software, make sure that the dump is correct or compatible, or take another one.

In "Mount" window after alignment of all fields, re-check all fields before pressing OK. If alignment sequences are incorrect, the fields may change themselves.

The "Mode" field : «Partial Media» field is required to manually specify a partition. If no partitions are found, you can enter them manually, for example <https://mslw.com/bb/showthread.php?tid=2...2#pid35582>

"Mode" field: "Entire media" is required to mount one extracted partition (for example, subtract one partition through "mode: partitions," it can be mounted), the path to which you need to specify in the field below.

The "Read Only" check box is a protector against rash actions. Some FS are Read Only

"No Recovery" check box disables the built-in mechanism of FS structure recovery.

Files only can be copied. There is no folder copy. You can create or delete folders if they are not empty.

You cannot navigate to other folders in the file manager while copying.

If you want to reconnect the box, then you need to restart the program. Otherwise, it will not be mounted. I.e. if you just did box reconnect after the socket, etc. - everything will work, but if the power is lost from it - then restart the software.

Linux and MacOS have no mount. Windows only.

Added hash file validation capabilities (for example, keys and export/import).

Added the ability to recover corrupted keys in QV platforms/chassis.

12 Package.

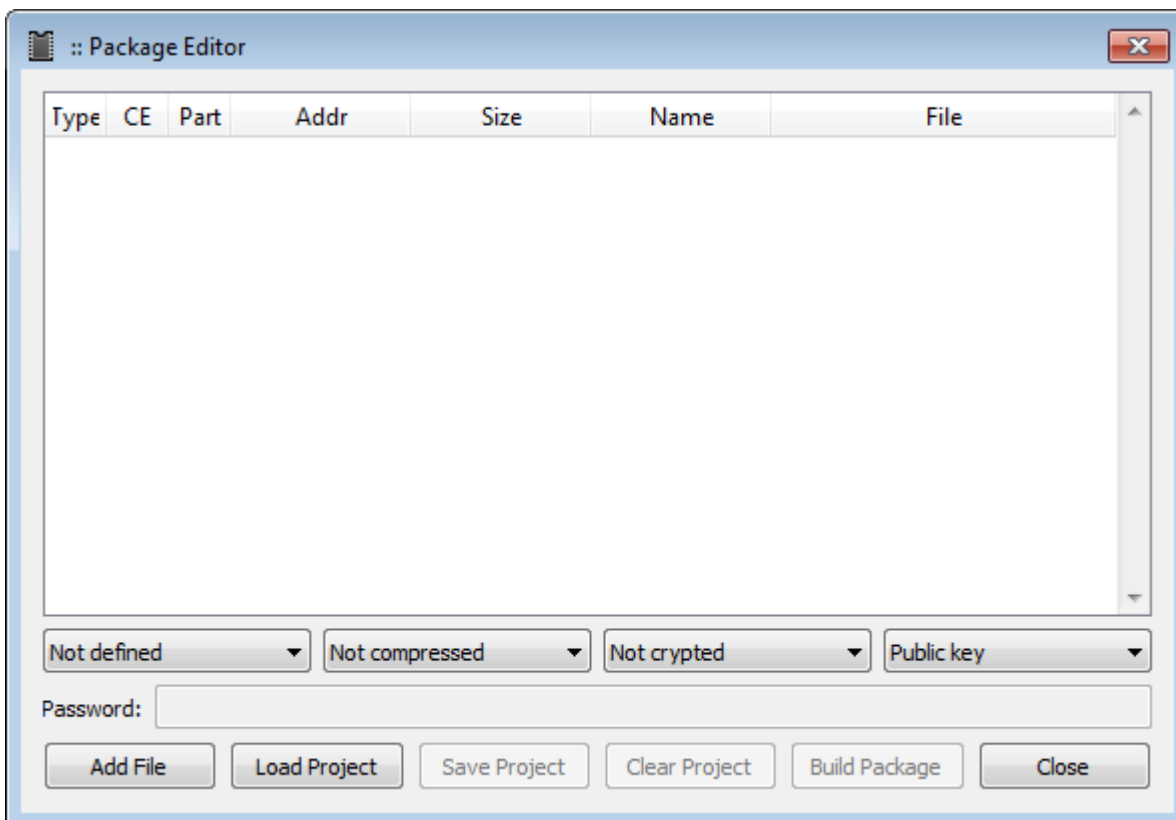
Features.

The main purpose of containers is to store several parts of firmware in one file and the ability to write all these pieces to different addresses from one file. It is also possible to add an image that contains the necessary information or reminder and is displayed when the Package is opened. If desired, the data in the Package may be compressed and encrypted. If a Package is selected as the file for recording, all write/erase/verify operations will be performed on the addresses and dimensions of the binary parts that are contained in the Package. In meaning, this is almost the opposite operation from "[Platform Backup](#)".

When creating a package for NAND, addressing is indicated without spare.

Package Editor.

Run the Package Editor in the menu «[Util-Package Editor](#)».



The drop-down menus set:

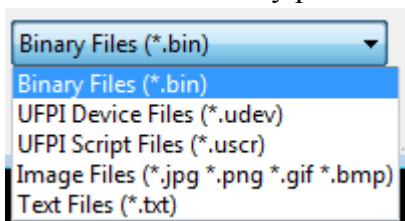
Chip type. If NAND or eMMC, select the appropriate option.

[Compressing level setting.](#)

Encryption and Password ([the same as with scripts](#)).

Buttons:

Add File. It can be any part of the dump, or a picture and others.



Open/Save/Clear Project. Controls the current project.

New Package. Generates a Package file with the selected options.

Open package for writing

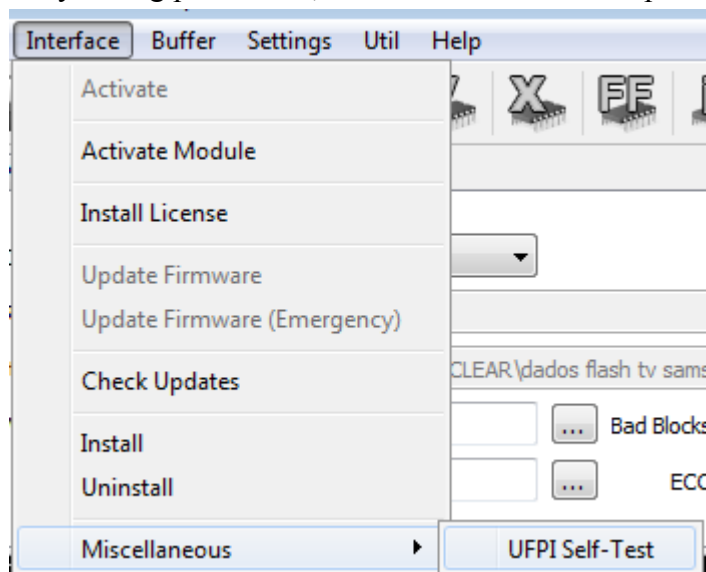
You can open/close a Package by pressing the hot keys, or by clicking File - Open Package and File - Close Package.

EXAMPLE

[The Package assembly and example work are described in Appendix.](#)

13 UFPI Self-Test.

It is designed mainly to detect defects in the assembly of the programmer - breaks, short circuits, incorrect operation of sensors, etc. It was made for internal use, but since sometimes there is a need for verification not only during production, it was taken out into a separate function in the programmer shell.



How it works:

For a complete test, you need to make a small socket with 4 resistors - 910K on ID1, 1K on ID2, 75 Ohm 0.5 W between the ground in VCC, 120 Ohm 0.5 W on VUSB. When the socket is turned on, the yellow LED will light up. To start, press "Interface, add.options, UFPI Self-Test." If something is wrong, then a list of errors will be displayed in the log. First, the processor signal lines on the board are tested, then frequency generation, voltage sensors, then the buffer lines are tested. Only extensions will be tested without a socket.

14 Working with scripts.

There is a separate [GUIDE](#) for writing scripts, which is replenished with current functions.

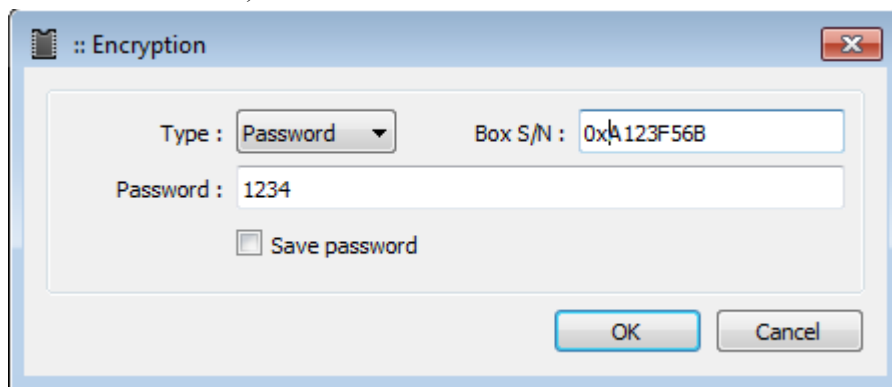
Scripts are common, which can be opened and edited as a text document freely, as well as without restrictions run, as well as encrypted. The file extension is the same .usr.

You can encrypt scripts by:

an open key - just so you can't peek at the content, but use doesn't limit.

Password - A password is required to start.

by the serial number of the box - only the owner of the specified box can bind and run the script (need to know that the serial number in the log is indicated in HEX form, so the entry of the HEX value should look like 0xA123F56B).



However, you need to know that encrypted scripts cannot be connected (# include) in another script.

Writing scripts.

Scripts are written in any convenient text editor (for example, with Notepad++ syntax highlighted).

For ease of debugging, the script can be [bound to a hotkey in the settings](#)

All debugging when writing scripts takes place by starting and checking in the log window. For example, the location and type of error are described:

Line 7, error 'No matching symbol 'setPowerMode''

Line 2, error 'Unexpected token '{''

It is convenient to use the print () output function for debugging and monitoring;

14.1 SCRIPTS.

A lot of programmer capabilities are executed using separate applications, (scripts). They are written by users. Script discussion forum here. This section lists some of them.

(Some older scripts have not been updated after changes or feature additions and may not run on the latest versions of the software, require changes.)

[HOT KEYS.](#) You can associate each script to a specific button.

OneNAND

[Script for finding Partitions in a dump and creating a .udev file](#)

NAND

[The script for quick detection if the BAD block installed in the NAND programmer hits on the block with information in the dump or not](#)

[Script for defining partitions in the NAND dump and creating a UDEV file.](#) What is UDEV and what it is for: [read here](#)

[Partition mini editor \(read/insert partitions from/to dump with support for UDEV files and section sizes based on title + manual data entry mode and light operation mode\).](#) You can work with one or more dams, dividing them into partitions and shuffling these partitions between them.

[Dividing the NAND dump by main and spare, as well as the reverse action - creating the dump from main and spare.](#)

[Script for creating a dump file in which a classic method of block BAD skipping is performed.](#) This bypass principle is very rare in practice, do not hurry to apply it until you understand - how exactly the bypass works for your device.

[The script for determining and setting the optimal parameter ReadRetry](#)

eMMC

[Script for defining partitions in eMMC dump.](#) MBR/EBR, GPT, MSTAR ANDROID, MTK LG, ANDROID TV SONY

[Partitioning dump from a UDEV file](#)

[Editing configuration constants in SMART TV ANDROID dumps based on CPU MSTAR.](#) It is necessary, for example, to activate uart. Also, with this script it was possible to open the uart on the Vestel chassis 17MB95

To work with eMMC dump partitions, if you have a UDEV file, you can use the script that we used to work with NAND dumps

[Partitions mini Editor \(read/insert partitions from/to dump with UDEV file support and partitions sizes based on title + manual data entry mode and light operation mode\)](#)

SPI

[script for working with OTP area](#)

Super I/O

[Script for working with multi-controllers KB9010, KB9012, KB9022](#)

Weltrend

[WT802, WT805, WT806, WT807](#)

Touch Sensor

[CT1C08x](#)

15 FAQ

Notes, which can be useful for quick familiarization with the forum, chips and programmer will be collected and grouped [HERE](#). In the topic of accompanying the guide, you can post new issues that should be included in the guide and FAQ.

15.1 Working with FORUM

How to get into the UFPI group after buying a programmer.

You just need to email your nickname on the forum with a request to add to the UFPI group. The email must be sent from the same e-mail address that was used for the purchase.

General Tips and Recommendations.

This is a technical forum. You should understand that if you write a message in the style "I can't program the SO8 chip" or "help to program the nand from the samsung," you will not receive a normal answer to your question. You will get, at best, a knock and treat you like an illiterate newcomer. This is a Russian forum, there is almost no tolerance.

Personally, no one owes you anything in this forum. You should understand that you can get help here on a voluntary basis, wasting other members personal time. Do not coarse to people who want to help you, even if you think that they are asking for something "wrong." Try to help newcomers yourself if you know the answer to the question.

You do not need to give advice or answer people if you "feel" that you know the answer to the question.

You do not need to give advice when no one asks for them. This is one of the worst features of Russian-language forums.

Query the topic to add the chip.

The first thing to do is to use the forum search. Enter a name or ID. If there is, then all questions will fall away.

Second. You must understand the connection and provide a read ID and, if possible, find the documentation for the chip.

If you cannot understand what this chip is, where to connect, and how to read the ID, then you do not need to ask about it in the topic of queries. Create a separate topic, or ask in the topic "Beginner's Question." «Beginners Question».

Messages in topics on request have a well-defined structure. Title - ID - Documentation - Comment.

TOPIC. Test versions. Errors, discussion for all modules.

Only current versions are discussed in this topic. If this is an error, then the message is formulated clearly to allow everyone repeat what you did and check the result. Describes the features of connecting chip, OS and others that may matter. This is not a topic for discussing your misunderstanding in working with the programmer. If you are going to write in this topic, then you are confident in your skills, use the current software version and are ready to answer counter questions.

The message must contain the following information:

OS version. An example is "Windows 10 x64".

UFPI software version, this is the first line when starting the program. An example is "UFPI Version 2.0.6 Compiled 4/12/21".

UFPI module. Example - "SPI Flash"

UFPI socket type and chip connection. Example - "SPI / I2C socket, socket"

Manufacturer and the name of the IC. Example - "Winbond W25Q80DLIG "

Brief description of the error and how to check / reproduce it

Text log of the program with an error

The log MUST be minimized under the spoiler!

The log should contain only the minimum amount of text with your actions and the error!

TOPIC. Test versions of the program.

Only the developer publishes in this topic. There is no need to write or ask anything here. This topic for tracking the development of programmer functionality.

15.2 General Questions.

Where do I download software, instructions, and drivers?

You can find on the official [website](#) by selecting the appropriate OS version. Also, if required by the [link](#), you can find drivers (not required for Windows 8 and above).

Where and how to connect?

Pay attention to USB cable and slots. This should be a good USB High-Speed cable and use connectors on the motherboard (all USB hubs and front panels of the PC are extremely not recommended), preferably USB 3.0 because it supports a large current.

How do I properly connect adapters during operation?

There is no difference in the sequence in which to connect the chips socket programmer. There are only obvious precautions when the Vcc power is on when the corresponding LED on the socket is on. The voltage Vcc is supplied only during the communication with the chip (the exception is eMMC), the rest cases the voltage is turned off all the time and manipulation is allowed.

How many supported chips are in the programmers database?

Almost any and almost all! If speak about SPIs, you do not always need to have them in the database for work, some of them have a special area with the configuration Serial Flash Discoverable Parameter (SFDP). NAND has ONFI, NOR has CFI, and EMMC also has JESD. Flash chips are automatically detected using the programmer's skills and configurations. You can also always quickly ask the forum to help compile a configuration file (if there is documentation on the chip), or ask the developer to make support. With very difficult cases and exotic tasks, scripts can be used.

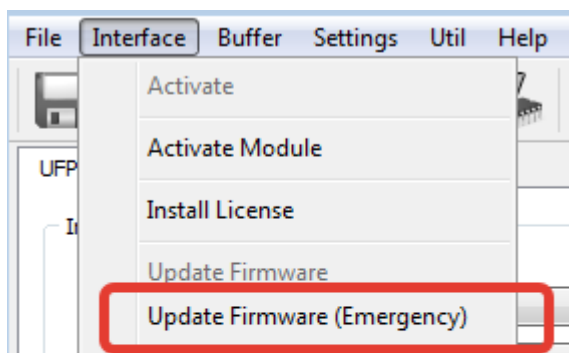
Why should I select a read and write files for reading and writing before operation?

UFPI does not create any temporary files when reading or writing. The process occurs directly with the selected file. This allows you to predict volumes and free disk space even with large flash volumes, as well as not waste time working with temporary files.

How do I downgrade back to the old version after the upgrade? This question often appears after the message "UFPI Outdated version for Box with firmware vX.X.X. Please use latest software version. "

Run the version you want to roll back to. Then press and hold the button on the disabled programmer. Connect the programmer to USB. After the programmer enters the update mode (all three LEDs will flash in yellow at the same time), release the button. [Then in INTERFACE menu press SOFTWARE UPGRADE](#), after that, the programmer will get firmware from the current version and it will be possible to use it with this version. *Important! It is not recommended to abuse this feature, since any chip memory has a target resource for erasure and write.*

Usually, rollback of versions works within the version base. You cannot downgrade back to version 2.0.5 from version 2.0.7.



Task Queue.

UFPI knows how to line up tasks, and then run them in turn. For example, we want to read-erase-record-verify a large flash. After selecting the file and starting the reading, you can immediately click the erase button, then check for cleanliness, select the file to write and click "write..." and the program will take performs all these actions in the specified order without stopping.

Enter HEX/DEC values.

You need to distinguish between the input of hexadecimal (0x100) and decimal (100) numbers in the UFPI program.

Menu for working with files.

Double-clicking on the "Read To" and "Write From" fields will provide a very useful menu in which you can specify AutoName files, work with ECC at NAND, and set One Folder Mode. The contents of the menu differ from module to module.

Check for free space and existing files.

In the settings, the main tab has check boxes check the free disk space before reading and alert if the the file of the same name is there before overwriting.

Where are the program logs located?

Logs are saved in the Logs folder next to the UFPI.exe program file. Logs are divided into files with the dates and names of modules in this folder. Also there are NAND and eMMC folders in which, respectively, the module stores ONFI, JESD, eCSD, CSD, CID, etc., each connected chip in the Logs folder . Nothing will be lost!

Built-in archiver.

The program has a built-in [archiver](#) that will help save disk space and can work without separate unpacking. It is also possible to unpack not only dumps, but other compressed gzip files, for example, UFPI updates.

How do I reset my settings? How to reinstall the program.

No registry or system folder settings or changes are made in Windows. Therefore, to reinstall, it is enough to delete the folder with the program and re-download, unpack to the desired folder.

To reset the settings, you can delete the ufpi.ini file (or the same as the executable file name if it was renamed). And then the next time it starts, it will be created automatically with the default settings.

At the bottom left of the Settings window, there is a Default button that will set the default settings on the current tab.

Contact test on connection.

In almost every module is present in "Miscellaneous.«[Pins Test](#)». Allows you to identify contacts that are closed to each other, or on VCC/GND with its address.

Sometimes when working with an ISP connection, false contacts can be observed. In such cases, you can turn it off. Otherwise, it is useful and is enabled by default.

ASYNC mode.

It allows to obtain a significant increase in speed. However, in some cases, you have to disable it, for example, if you want to read almost dead eMMC or NAND with repetitions in case of errors.

–[highlights in the LOG](#). (By default)

White is the main text. For example, operation name, file name...

Green - Highlight information in the common text. Task to run, values...

Yellow is a warning. These can be NAND bit errors, or inappropriate file size, or other non-critical warnings that should be noticed.

Red is an error. Errors of critical significance and the result in this case is incorrect.

Blue - highlighted selected information. Specific information, for example, in NAND if "Addresses with spare" are set, in this case addresses are highlighted.

[Emulation MODES](#).

When you connect a socket in emulation mode, you will not see the opened tabs of the module, and a new device will appear in Device Manager. If you got a similar problem when connecting a JTAG socket for example, simply disable the emulation mode in the main software tab by selecting "Default" and clicking "Fix."

15.3 Commonly called:

Socket.

This board, which is inserted into the programmer, has switches and ID resistors to set the desired voltage and module. It may be supplied with additionally "base" with the necessary sockets/pins/connectors for connecting arbitrary adapters.

Socket Base.

It is inserted into its corresponding socket and is designed to install the bga or tsop socket, other base or flash directly.

BOX.

UFPI programmer.

Module .

This is the tab in the software, for example NAND/SPI/eMMC... which is active when the appropriate socket is inserted and the license is available.

BB (Bad Blocks) .

Bad Block. Faulty block in NAND chip.

Page, Block, Spare.

Page, which consists of Block. They consist of almost all chips. Their number and size is the size of the chip. NAND still has a service area at the end of the page called Spare.

Chips geometry (parameters/configuration).

It is customary to describe the chip parameters in the sizes Page, Block, as well as Sector, Spare, LUN, CE and others. A collection of parameters and is called "Flash/chip Geometry"

CRC (Checksum).

This is an algorithm, or value, that is designed to verify data integrity.

15.4 Power supply during operation.

Power supply management.

Right-clicking on the zipper button (socket power switch) will show the voltage control menu where you can set any desired mode.

Low USB voltage.

Important! Power supply voltage with normal cable, connector and PC is not less than 5 volts. In case of critical voltage, the readings in the status bar will be highlighted in red! With such subsidence, voltages malfunctions not excluded in operation (especially in ISP mode).

Turn off the current protection of the power supply.

The UFPI has two power channels with the protection of the disconnecting load when it reaches 500mA. Sometimes, in ISP mode, you have to go over this value. At your own peril, you can set the check box "Ignore the accident" "ON" on the desired channel and press the "Fix" button in confirmation

15.5 NAND.

How to work with NAND having multiple CEs.

If the chip is determined by the machine, is, in the programmer database and in the log it is displayed in the correct full size (in the log it is written "Number of chips (CE) 2"), then it will be read by one file. If the chip is unknown and the configuration is added manually, then each CE will need to be read individually by selecting the desired CE from the menu. CE count cannot be set in configuration!

How to read/write chips in partial or partitions.

To do this, there is an "[RW Mode](#)" In each module, it can have its own characteristics. Need to know, that In NAND minimum erasure value is "Block" and the minimum write value is "Page." When reading the ID, they can be viewed in the log for a particular chip.

Addressing.

In the device, work with NAND occurs page by page. Spare is not treated as a data area. In a regular read dump you can see that it consists of Page and Spare. Therefore, addressing in the dump will differ from addressing when working in the "RW Mode," or reading terminal logs, or a programmer logs. There is a parameter "Addresses with Spare" in the menu "Miscellaneous/Add.features." to see the address with Spare in the log.

Write a dump using analysis.

[Dump analysis when ID reading](#) allows you to configure the programmer to work with the necessary (if known) ECC, BB, BBT automatically.

Remember, however, that the dump must be pre-cleaned of BB and match the geometry of the installed NAND.

–[Do Not Write Blank Data](#).

It is the default mode and allows save time. But in some extremely rare cases, the record may be incorrect

for NAND with HW ECC.

"Too many errors when reading. Aborted! "

It can be found when the ECC scheme is incorrectly selected or the dump is corrupted. To read such dump, you need to disable ECC. Or disable Asynchronous mode.

15.6 EMMC.

READING faulty EMMCs.

Read repeats in fault eMMC allows you to read if reading failures. It may be necessary to read some faulty chips. You additionally need to disable in "Miscellaneous" Async IO, in "Read mode" set "Ignore read errors" and "Read mode: Auto" or even in "Single Blocks". Set the number of retries in the module settings. It is also desirable to reduce the frequency and number of lines to 1bit.

Analysis when ID reading (EMMC).

"Miscellaneous" allows you to select an object to be validated and applied to the partition table by internal UFPI algorithms without scripting. You can select a file, or a chip.

EMMC erasing.

According to the specification, the controller of these chips automatically erases them before writing the blocks to the chip. Therefore, it is not necessary to erase chip, some chips does not support such commands at all. But this possibility is present and in different chips either the erase command can be supported, or you can simply write zeros or 0xFF. By "X" button the erasure algorithm corresponds to "Miscellaneous" - Erase Mode. In the Samsung eMMC before v5.0 revision, the formatting function is also supported (Separate item in "Miscellaneous"), which erases both the data and the loading configuration (make factory reset).

15.7 SPI.

Power supply.

Special attention should be paid to chips power supply before starting work. It is not always possible correctly distinguish between 1,8/3,3V when reading ID/SFDP. Exact values can be obtained from the documentation for the chip or by studying the equipment on which the flash is installed.

For ISP mode with 3.3 volt flashes (for example, "pinch"), an attempt to work at voltage levels of 2.8 or even 2,6V is often more successful. At the same time, it is useful to set the power check box "Always on" and several times re-read the ID, monitoring that everything works stably.

15.8 UART.

AUTO speed.

It is very rare and convenient determine the baudrate automatically.

The characters or messages you enter are duplicated in the log.

When typing (with the "Send data from the keyboard" check box set ON) or sending a message through the "Text/Hex" field, you can see repetitions of the type: "hheellpp." Need to remember that this is not a "Terminal," but a log. It shows what is being sent and the echo being received.

"vi" and similar programs may not be displayed correctly in the log for the same reason.

16 Attachment.

16.1 Programmers, sockets, matching and compatibility.

It is easier to navigate by articles in this manual:

UFPI Красный Вернуться к...

UFPI Red Box - комплектация Расширенная.

Артикул 2007



Цена без н
Разм

1 + -

Добав



- 2006 [UFPI Silver](#)
- 2007 [UFPI Red](#)
- 2008 [UFPI Black](#)
- 2037 [UFPI PCB](#)
- 2036 [UFPI Pro PCB \(снят с продажи\)](#)
- 2041 [ICFRIEND 8-Bit Socket](#)
- 2040 [E-MATE X 8-Bit Socket](#)
- 2039 [E-MATE Pro 8-Bit Socket](#)
- 2038 [EASY JTAG eMMC 8-bit Socket](#)
- 2063 [UFPI to Easy JTAG eMMC Socket](#)
- 2018 [NAND TSOP48 Wells-CTI](#)
- 2019 [NAND TSOP48 Pinboard](#)
- 2017 [NAND DIP48 ARIES](#)
- 2020 [NAND Base SMT](#)
- 2021 [NAND BGA63 KZT SMT](#)
- 2009 [NAND BGA63 KZT SMT Base](#)
- 2010 [NAND BGA63 JRS SMT Base](#)
- 2056 [TSOP48 Wells SMT](#)
- 2058 [TSOP48 Pinboard SMT](#)
- 2061 [TSOP56 Pinboard SMT](#)
- 2057 [ANDK TSOP48-0.5](#)
- 2062 [ANDK TSOP56-0.5](#)
- 2031 [OneNAND Base SMT](#)
- 2030 [OneNAND BGA63 KZT SMT](#)
- 2011 [OneNAND BGA63 KZT SMT Base](#)

2059 [OneNAND BGA63 JRS SMT Base](#)
 2023 [NOR Base SMT](#)
 2024 [NOR BGA64-1.0 KZT SMT](#)
 2012 [NOR BGA64-1.0 JRS SMT Base](#)
 2026 [TSOP56 ANDK SMT Universal](#)
 2025 [TSOP56 ANDK SMT J3X Series \(NOR\)](#)
 2027 [TSOP56 ANDK SMT P3X Series \(NOR\)](#)
 2042 [SD Card Socket](#)
 2046 [SPI/I2C/1W DIP24 Socket](#)
 2060 [SPI/I2C/1W/UART DIP24W Aries Socket](#)
 2044 [UART/GPIO Socket](#)
 2043 [JTAG/BDM Socket](#)
 2045 [LOGGER Socket](#)
 2034 [UFPI Breadboard](#)
 2013 [Frame 10.5×13.5 BGA63 KZT](#)
 2014 [Frame 10×13 BGA63 KZT](#)
 2015 [Frame 11×13 BGA63 KZT](#)
 2016 [Frame 9×9 BGA64 KZT](#)
 2035 [eMMC RT809H 8-Bit PCB](#)
 2032 [OneNAND ENTT PCB](#)
 2028 [OneNAND ENTT BGA JRS PCB](#)

In future we will navigate the articles and list the options. I will list the options for some kits that should be paid attention to for understanding.

16.1.1 NAND sets.

Most Common TSOP48 Options.

Most high-quality and durable adapter is 2018.

Most budgetary option is 2019 + 2057.

Newest and universal - 2020 + 2058 + 2057.

The previous option with the Wells socket is 2020 + 2056.

Universal as possible with a high-quality sockets for using adapters with 1:1 printing from other programmers (you can use not only for tsop48, if the pinout is the same) - 2017.

NAND BGA63.

Highest quality socket KZT - 2020 + 2021.

Using your own socket (for example, if there is 2031 + 2030) KZT - 2020 + 2009.

Using your own socket JRS - 2020 + 2010.

OneNAND BGA63.

Highest quality socket KZT - 2031 + 2030.

Using your own socket (for example, if there is 2020 + 2021) KZT - 2031 + 2011.

Using your own socket JRS - 2031 + 2059.

Option using combined with ENTTv2 and your own socket JRS - 2032 + 2028.

16.1.2 SD/eMMC sets.

Socket 2042 is primarily for working with SD cards and is less intended for eMMC and ISP mode.

Sockets 2038, 2039, 2040, 2041 are ready, tested and designed to work with the appropriate sockets, as well as ISP mode. You should be very careful about choosing the socket corresponding to your needs, since they all have differences in pin-outs.

Socket 2035 is an empty board for the socket. Elements and connectors are not included. Purchased and installed by the user according to typical diagrams of eMMC adapters.

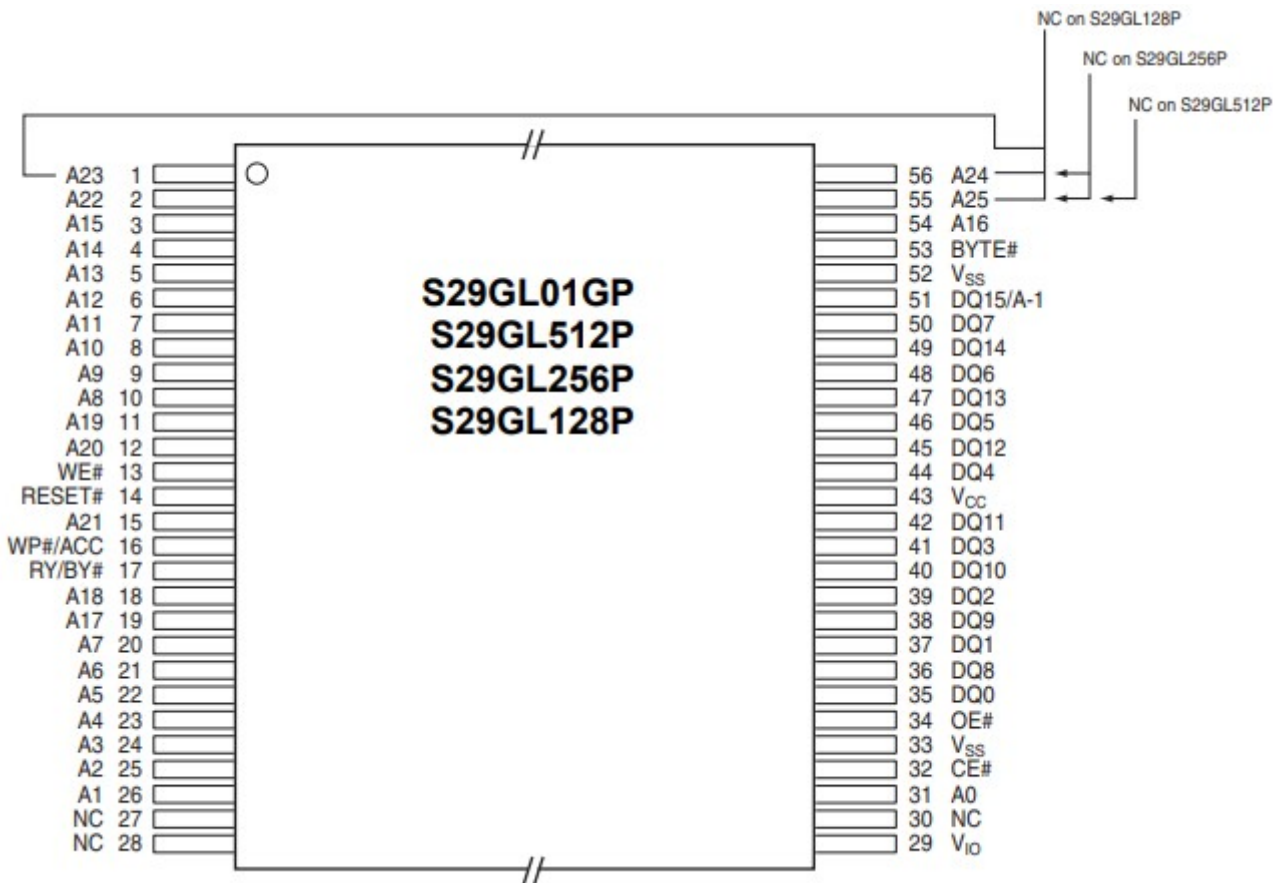
Socket 2063 can be used as a universal UFPI - Easy JTAG adapter. Suitable probably for any eMMC "helicopters"

16.1.3 NOR sets.

The kit for TSOP48 is 2023 + 2058 + 2057.

The kit for TSOP48 with WELLS block is 2023 + 2056.

The kit for TSOP56 (standard printing) - 2023 + 2026 (or 2061) + 2062.



Kit for TSOP56 (P3X) - 2023 + 2027 + 2062.

16.1.4 Other kits

SPI/I2C/1W/UART

Socket 2046 is most common and familiar, but today there is newest 2060. In 2060, It is able to turn on UART and has been added and additional pins for connection.

The position of the switches on the socket corresponds to the following modes:

Режим	Vcc
ON ON ON – SPI Flash	ON ON ON – 3V3
OFF ON ON – I2C EEPROM	OFF ON ON – 1V8
ON OFF ON – SPI EEPROM	OFF OFF OFF – 5V0
OFF OFF ON – mWire EEPROM	ON OFF ON – 2V6
OFF OFF OFF – 1-Wire MODE	
ON OFF OFF – UART	
ON ON OFF – SNAND (SPI NAND)	

NOTE.

- Worth remembering that you can use any of their suitable ones sockets instead of 2057 and 2062 . They're universal.
- Sockets such as 2043, 2044 have a connector for connecting their own wires RJ45. Its Very convenient for quick replacement and organization of a large set of own adapters.
- Empty socket boards (PCBs) are not equipped with connectors or elements.
- For a TV serviceman, usually the following are important:
- SPI/I2C/UART.
- NAND TSOP48.
- EMMC.
- OneNAND BGA63.
- The rest, such as NAND BGA63, NOR are not so actual.

16.1.5 Tips and useful information for selecting your adapter kit.

Universal adapters.

If you have already decided to buy a programmer, then SPI/I2C/1W DIP24 (art. 2060 or 2046) socket is what is commonly called a "must have" adapter. With this universal adapter, the programmer can become a tool for working with various types of memory, protocols and voltages. By selecting the desired mode of operation, you can work with SPI NOR, SPI NAND, SPI EEPROM, I2C EEPROM, Microwire (3-Wire) EEPROM and 1-Wire devices.

Adapters for eMMC.

E-MATE X EMMC BGA 13 IN 1 E-MATE PRO BOX EMMC TOOL



EASY JTAG Socket



ICFRIEND High Speed E-MATE



When choosing, you need to understand the following things. The store does not sell BGA kits for working with eMMC, they are purchased separately on Aliexpress and similar shops. Which of the Chinese BGA kits to choose and buy depends only on your preferences, finances and what is sold there at this moment. The UFPI store sells adapters for these "standard" Chinese kits. So the question of choosing an adapter to work with BGA eMMC depends of which one you will have or already have.

There is a pcb circuit board for the RT809 programmer, that will allow you to independently assemble eMMC socket for UFPI and use existing BGA adapters.

If you already have or plan to buy, for example, EASY JTAG EMMC BGA kit - it means to buy UFPI EASY JTAG eMMC socket (2038), etc. If you only need to work in ISP mode, then there is no difference what eMMC adapter you need, all of them are suitable. If desired, you can assemble the adapter independently, adapter diagrams in the application and on the site.

Adapters for NAND.

When selecting adapters for NAND, you need to understand whether you plan to work with OneNAND, NOR in TSOP48.

If you only need to work with NAND in TSOP48, then the choice is quite simple. There have enough money - buy NAND TSOP48 Wells-CTI (2018). If you need to save money - buy NAND TSOP48 Pinboard (2019) with ANDK TSOP48-0.5 (2057) or NAND DIP48 ARIES (2017) if you have suitable NAND adapters from other programmers.

If you plan to work with BGA and/or not only with NAND, but also with NOR, then the best choice will be the NAND Base SMT adapter with panels from the SMT series (TSOP48 Wells SMT (2056), TSOP48 Pinboard SMT (2058), NAND BGA63 JRS SMT Base (2059)). The advantage of this choice is that the upper plug-in adapters, for example, TSOP48 Wells SMT (2056), can be used with all SMT series adapter bases. So TSOP48 Wells SMT (2056) can be used with both NAND and NOR and SMT OneNAND bases.

Adapters for OneNAND.

It is important to understand that almost all chips of OneNAND in cases BGA63 similar in the size and a view as NAND, this type of memory has absolutely other configuration of signals and power supply which don't coincide with NANDs. Therefore, to work with OneNAND, a basic adapter OneNAND Base SMT is required. If you already have a JRS socket BGA63, you can use NAND BGA63 JRS SMT Base (2010). If have enough money or need better quality - take OneNAND BGA63 KZT SMT (2030) in the store.

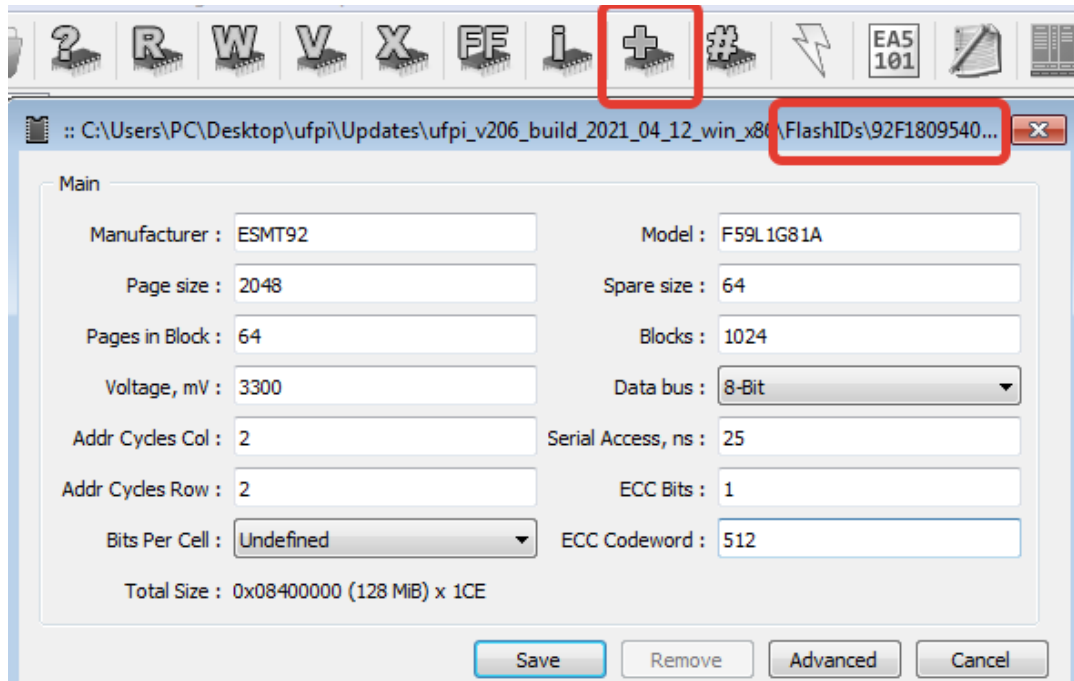
Adapters for NOR.

To work with NOR, a basic NOR Base SMT adapter is required. TSOP48 Wells SMT (2056) or TSOP48 Pinboard SMT (2058) are suitable for working with TSOP48. For TSOP56, you need the Pinboard SMT (2061) TSOP56. The above adapters are designed to work with most NOR chips having "standard" signal mapping. The TSOP56 ANDK SMT J3X Series (2025) board is designed to work with Intel J3x chips. And the TSOP56 ANDK SMT P3X (2027) Series board is designed to work with chips with synchronous NOR P3x series. BGA NOR BGA64-1.0 KZT SMT adapter (2024) is designed to work with NOR chips in BGA version with 1mm increments. If you already have BGA64 socket from JRS, then it is possible to use NOR BGA64-1.0 JRS SMT Base (2012).

16.2 EXAMPLES .

16.2.1 Attachement. Create a NAND Flash configuration file.

If the chip is not determined, or is not determined correctly, the "+" button opens the menu of the configurator, where you need to enter the main parameters of the chip for operation.



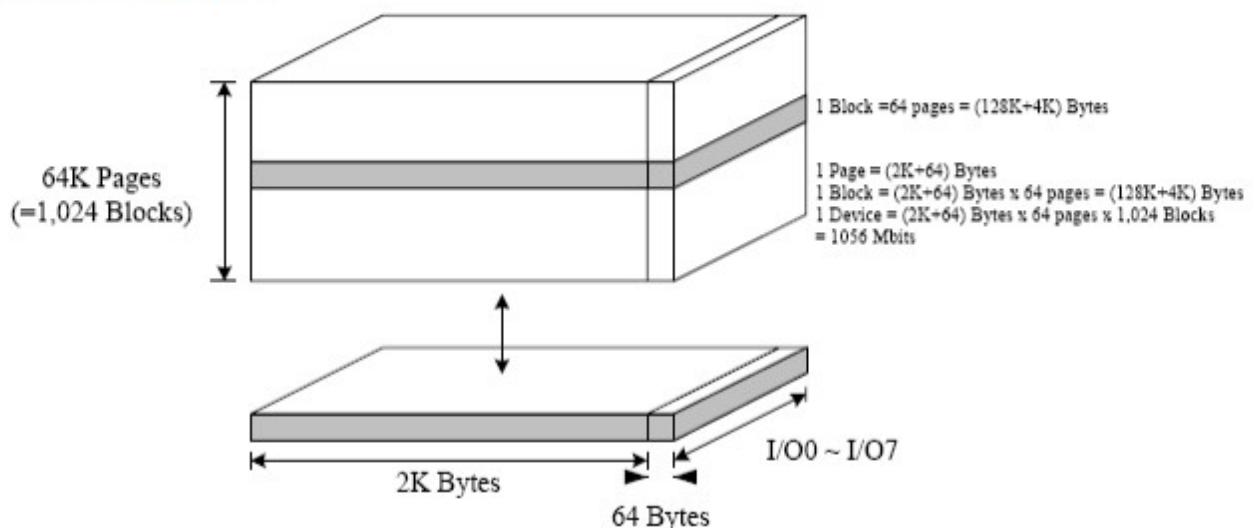
The official documentation is the main reliable source from the manufacturer, which is usually freely distributed on the Internet.

Lets see ESMT F59L1G81A SPI ID 92F18095407F7F7F chip. Free access [documentation](#).

Usually everything you need is written on the front pages. The first page says: 1 Gbit. The full size of 1Gbit can be represented as 128Mbyte or as $128 * 2^{20} = 134217728$ byte or translated into a hexadecimal system of 0x8000000 bytes. This size is without the spare area.

The 3,3V voltage for writing is converted to millivolts 3300.

Next, the first page lists the basic parameters of the chips geometry. Also, usually there is a visual picture of the "ARRAY ORGANIZATION," in the documentation for each chip similar to given in the current documentation on page 4.



Page size «- Page Program: (2K + 64) bytes» (it consists of a page + spare size).

Page = 2K = $2 \times 2^{10} = 2048 = 0x800$ byte;

Spare = 64 = 0x40 byte.

The total page size in the read dump is 0x840 bytes.

Block size: «- Block Erase: (128K + 4K) bytes» (sum of page size+spare).

128K = $128 \times 2^{10} = 131072 = 0x20000$ byte. Block size without spare;

4K = $4 \times 2^{10} = 4096 = 0x1000$ bytes;

Summary block size: $0x20000 + 0x1000 = 0x21000$ byte;

To calculate the number of pages in a block, you need to divide the block size (considering or not considering spare) by the page size: $0x20000 / 0x800 = 0x21000 / 0x840 = 0x40 = 64$ (pages in block).

The number of blocks can be obtained by dividing the total chip size to the block size. It was noted above that the full size was considered excluding spare. So to get the number of blocks, it must be divided by the size of the block without spare: $0x8000000 / 0x20000 = 0x400 = 1024$ blocks.

The data bus is determined by pinout. For example, for TSOP48 NAND chips, terminals 29-32 and 41-44 are standard, this is distributed I/O0-I/O7 i.e., 8 bits of the data bus. If in addition to them, there are still I/O8-I/O15 nearby, then it will already be x16 bit.

Access time. This value is named as tREA in this documentation and is 20ns in the "AC Characteristics for Operation" table on page 8. Page 13 illustrates the position of the tREA among the signal sequence. This is the time at which the internal chip buffer puts the byte on the I/O data bus and is ready to read.

Col/Row address loops. The easiest thing is to find the "Address Cycle Map" table on page 4, which usually always clearly describes the necessary values. It shows that the first two lines (the number is limited to * L) are "Column Address" addressing inside the page.

The second two are "Row Address" for addressing pages/blocks, etc.

It is easy to calculate that if the full page size of the 0x840 chip in binary form can be represented by 12 bytes (100001000000) this is A0-A11 indicated in the first lines.

Also, if you calculate, the total number of pages is 0x10000. Two bytes are required to enumerate them from 0x to 0xFFFF. This is the second two 8-bit rows in table A12-A27.

ECC bit. This value indicates the number of valid errors during guaranteed chips recording cycles. Based on this, the equipment manufacturer uses the ECC correction algorithm. So on the first page of the documentation it is indicated: "- 100K Program/Erase Cycles (with 1 bit/528 bytes ECC)" and means that during 100,000 erase/write cycles the number of errors will not exceed 1 bit by 528 bytes. In the case of a configurator, it is rounded to 512. This value is not critical to the operation of the programmer and is used only in verification to evaluate chip condition after recording.

Bit in a cell. In the case of F59L1G81A, this is SLC. Among other data on the first page is indicated: "Memory Cell: 1bit/Memory Cell."

Small hint:

Single-Level Cell or SLC (1 bit per cell)

Multi-Level Cell or MLC (2 bits per cell)

Triple-Level Cell or TLC (3 bits per cell)

Quad-Level Cell or QLC (4 bits per cell)

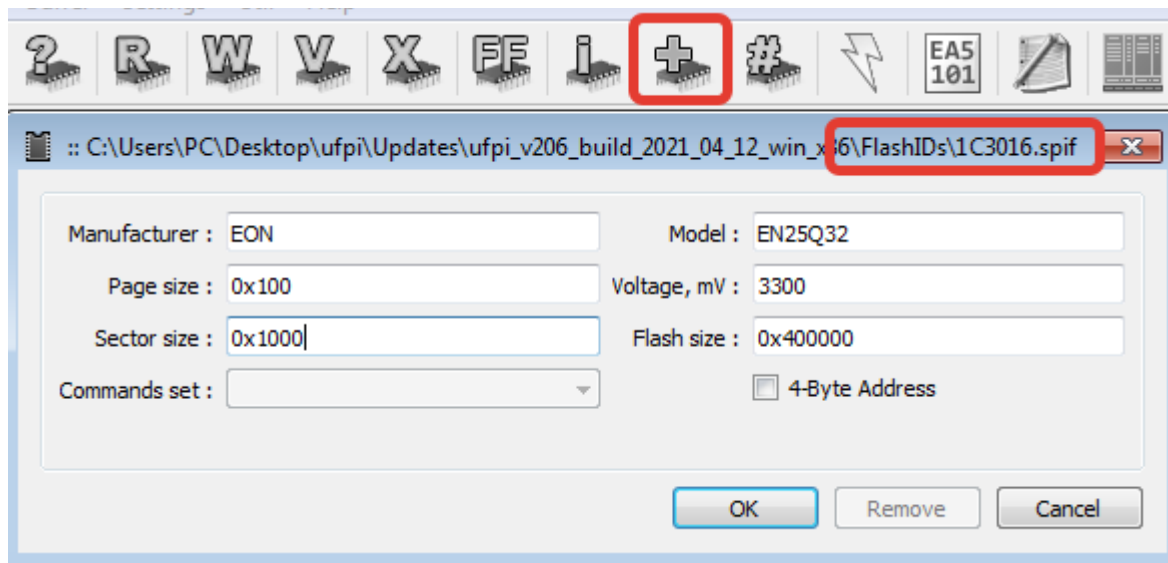
Penta-Level Cell or PLC (5 bits per cell)

When all fields are correctly filled in, the full size will be calculated below, coinciding with the calculations above.

The "Save" button saves the configuration file to the folder FlashIDs and this configuration will already be triggered by default when reading the ID. To delete, you can use the "Delete" button or delete the file specified in the header of the configurator window.

16.2.2 Attachment. Compiling the SPI Flash configuration file.

If the chip is not determined, or is not determined correctly, the "+" button opens the menu of the configurator, where you need to enter the main parameters of the chip for operation.



The main reliable source is official documentation from the manufacturer, which is usually freely distributed on the Internet.

Consider the EON EN25Q32B SPI ID 1C3016 flash (1C30161C30161C30: 9F). Documentation is available on the official website.

Usually everything you need is written on the front pages. First page says 32 Megabit. The full size of 32Mbit can be represented as $32/8 = 4\text{Mbyte}$ or as $4 * 2^{20} = 4194304\text{byte}$ or translated into a hexadecimal system of $0x400000$ bytes.

The voltage is further indicated as "Full voltage range: 2.7-3.6 volt." In millivolts, it is common to introduce 3300mV into the configurator.

The page size is placed on the first page: "256 bytes per programmable page." You can either fit into the configurator 256 or convert to a hexadecimal view of $0x100$.

The size of the sector is slightly lower: "1024 sectors of 4-Kbyte." This translates to bytes $4 * 2^{10} = 4096 = 0x1000$.

You must also check with the identified ID on page 15.

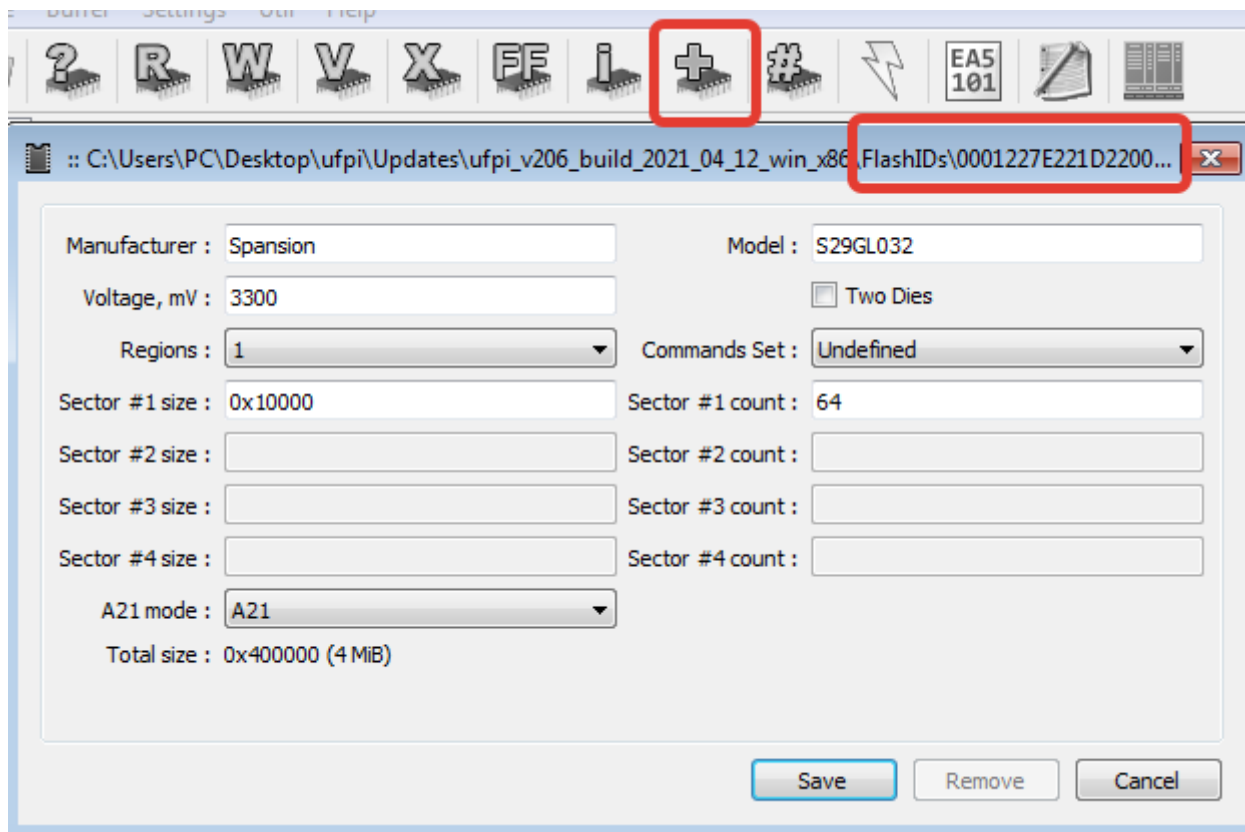
ID in the log "SPI ID 1C3016."

4-byte address will be needed for large flashes only.

For example, on the 22 documentation page there is a description of the reading command and the line: "The instruction code for the Read Data Bytes (READ) instruction is followed by a 3-byte address (A23-A0)." The illustration also shows these 24 bytes. This chip addressable limit will be three bytes from 0 to $0xFFFFFFFF = 0x1000000$ bytes. When transferred to Mebibayty $16777216/2^{20} = 16\text{MiB}$ or to bits $16 * 8 = 128\text{Mib}$. I.e. for SPI, a flash of more than 128Mib (for example 25Q256) will already be described in the documentation about the 4 byte address.

16.2.3 Attachment. Creating NOR Flash Configuration File.

If the chip is not determined, or is not determined correctly, the "+" button opens the menu of the configurator, where you need to enter the main parameters of the chip for operation.



The main reliable source is official documentation from the manufacturer, which is usually freely distributed on the Internet.

Let's take a look at the Spansion/Cypress S29GL032N90FFI02 flash. NOR ID 0001227E221D2200 AMD Algo (CFI).

[Documentation](#). From official webpage.

The volume and operating voltage can be seen on the first page. Here, the 3V is written in the configurator window as 3300 (mV).

The full size of 32Mbit can be represented as $32/8 = 4\text{Mbyte}$ or as $4 * 2^{20} = 4194304\text{byte}$ or translated into a hexadecimal system of 0x400000 bytes.

Please note the full marking. On page 12, you can determine that MODEL NUMBER = 02 = x8/x16, VCC = VIO = 2.7 - 3.6 V, Uniform sector, WP #/ACC = VIL protects lowest addressed sector.

Memory organization. On page 19 there is a table with the listed sectors. It can be seen from the table that all 64 sectors of the same size 64KB. As such, the configurator selects one region and the size and number of sectors, respectively.

Similarly, we convert 64KV to bytes by multiplying by $2^{10} = 64 * 1024 = 65536 = 0x10000$.

If to see 20 page for the neighboring memory model, then the table shows that there are two regions. The first counts 63 sectors by 64KV and the second 8 sectors by 8KV.

Pay attention to the numbering of sectors, since it can go both from the beginning and from the end. The "Top Boot Sector" and "Bottom Boot Sector." The order of regions will depend on this. The size of the flash corresponds to the size from the beginning of the first to the end of the last sector and can be checked in the column "8-bit Address Range." We get from 0 to 3FFFFFFh in total we get 0x400000 bytes. This corresponds to the one previously calculated.

Checking ID. Page 29 shows the table. It is always worth checking the ID against the documentation. Because an incorrect ID may indicate a bad contact, malfunction, or mismatch with this documentation. Source ID 0001227E221D2200.

Manufacturer ID: Cypress Products are always the first

DQ8 to DQ15 = 0x00;

DQ7 to DQ0 = 0x01;

then lines for S29GL032N

Cycle 1 = 0x227E;

Cycle 2 = 0x221D;

Cycle 2 = 0x2200;

The remaining columns can be verified from the full ID in the log. There is also useful information.

Command set. The most usable are AMD. On page 47, you can see the commands in the table. The first cycles in the commands in the case of AMD are always inserted similar 555/AAA sets of bytes.

In the case of an INTEL instruction set, there will be no such incremental bytes and there will be simply instructions. For example, see the Intel 28F256J3 documentation on page 35.

Do not think that if Intel flash, then it has Intel command system. That's not always truth.

21 mode is selected based on the chips pinout. For TSOP56, this is 15 (or 13 for TSOP48) leg of the chip. In the case of c S29GL032N, it acts as the address line A21.

If you open the documentation on Intel 28F256J3 on page 15, then on the 15 leg will be Vpp/Vpen. Table 7 of this documentation describes that it is normally connected to Vcc (2.7 V-3.6 V)

When all fields are correctly filled in, the full size will be calculated below, coinciding with the calculations above.

The "Save" button saves and, when reading the ID, the configuration is already triggered, filled and saved in a file in the program folder by default. To delete, you can use the Delete button or delete the file specified in the header of the configurator window.

16.2.4 Attachment. File manager and FS (file system) mount.

Partition layout and table.

After installation of EMMS, and ID reading (in "Miscellaneous "can be selected for dump analysis or EMMC), if we recognize markup successfully, we get something similar in the log:

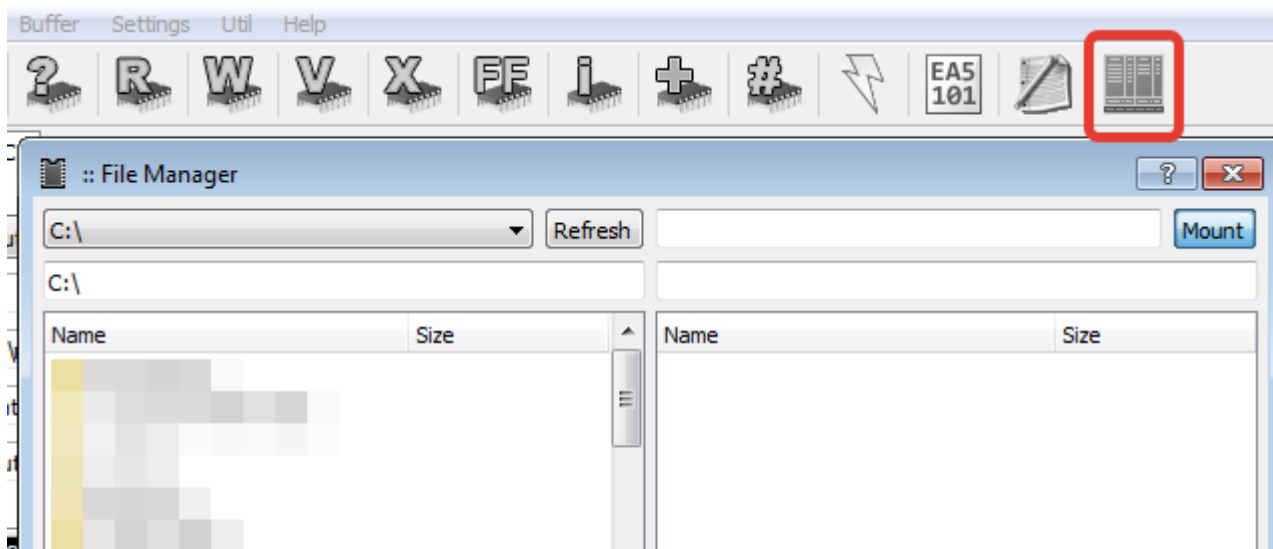
```
eMMC Dump check...LG
swue SQUASHFS
swue SQUASHFS
cert SQUASHFS
...
otncabi SQUASHFS
otycabi SQUASHFS
fonts SQUASHFS
smartkey SQUASHFS
db8 EXT4
data EXT4
apps EXT4
eMMC Dump analysis done. Extracted 49 part(s)
```

If nothing is found:

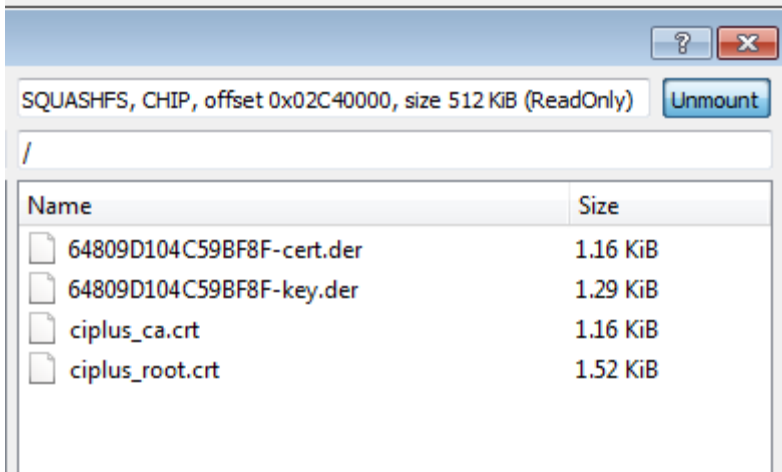
```
eMMC Dump check...No Partitions found!
```

Checking the log. The found partitions, number and type of FS are described, which you can work with at the moment. A complete list of partitions can be seen in the "RW Mode" window (Alternatively, you can always use udev or a script to create it on a previously read dump. After working out the script, do not forget to exit the script)

With The button in the panel opens file manager and click "Mount" for mounting.



If the mounting is successful, the content and description of the partition will appear in the file manager.



File Manager allows you to go through directories, copy, replace, and many other operations. You can perform operations on selected files by right-clicking context menu.

16.2.5 Attachment. Reading and writing partitions/regions.

Reading and writing partitions by **RW Mode**, „Partitions” mode».

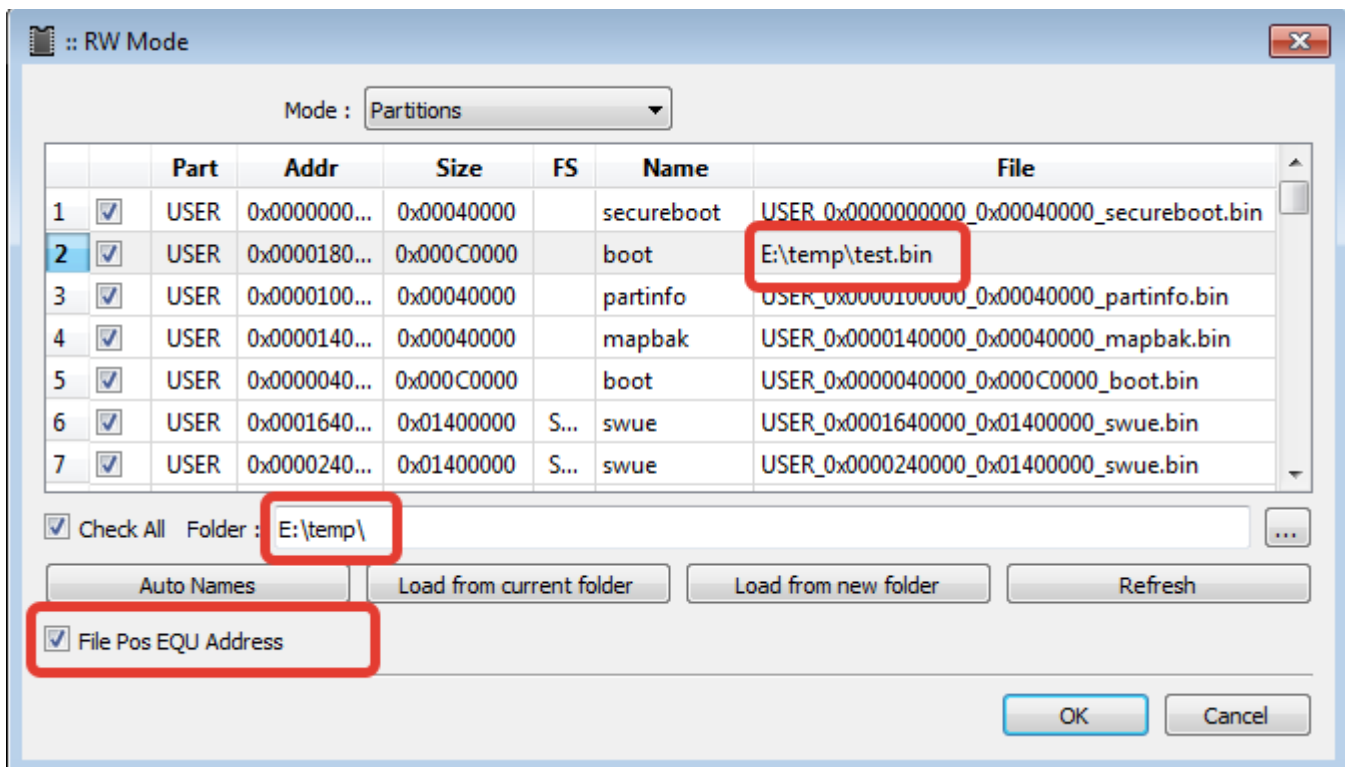
The most convenient and visual way to work is to divide into partitions using analysis when reading ID, or using scripts, or .UDEV files.

Open the "RW Mode" window, select the "partitions" mode, check the necessary ones, select the working folder, update and OK. Next, we read/write with the usual read/write buttons. All regions will be separate files in the selected folder.

Write/restore one partition from a dump.

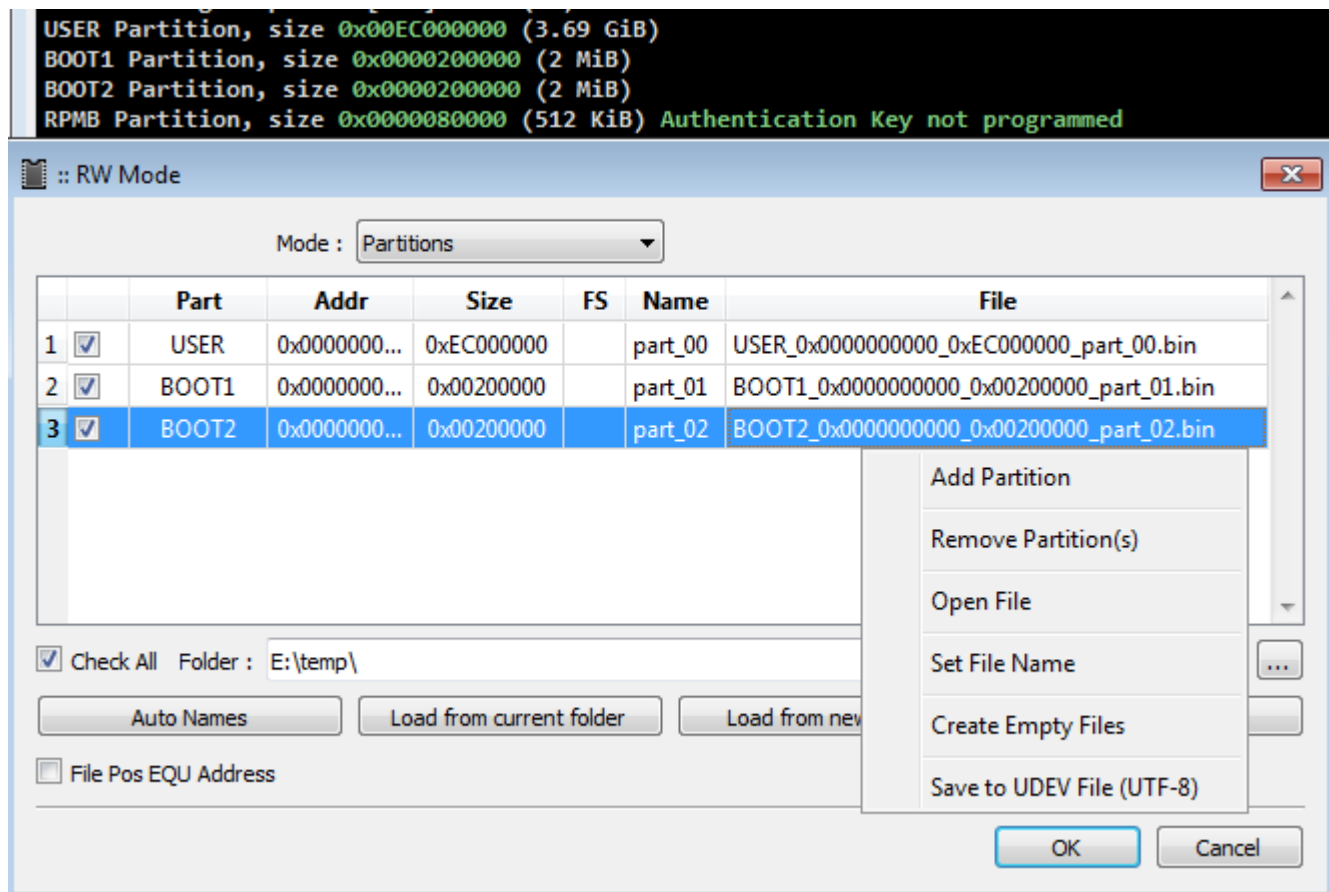
In addition to the convenient "Platform Backup", you can write separately a partition from the entire dump of the donor. To do this, in the "RW Mode", select a full dump file in the line of the desired partition. The folder is working accordingly. And most importantly, set the check box "File Pos EQU Address".

Press "OK" and recording. Only the partition we need will be recorded from the full dump file.



16.2.6 Attachment. Full Backup with RW Mode and .UDEV.

The RW Mode allows you to save/write partitions not only from USER eMMC, but also to combine.



Read the flash ID and fill in the partition sizes in the "Partitions" RW Mode window:

So you can read and write. You can select files from the folder where you read "Full Backup" to write all the required partitions at once.

You can similarly compose or save a .UDEV file:

[DESC]

Name = Partitions

FlashType1 = eMMC

FlashBase1 = 0

[PARTITIONS]

PartitionsMode = true

0x0000000000,0x1D200000,part_00,USER,USER_0x0000000000_0x1D200000_part_00.bin

0x0000000000,0x00400000,part_01,BOOT1,BOOT1_0x0000000000_0x00400000_part_01.bin

0x0000000000,0x00400000,part_02,BOOT2,BOOT2_0x0000000000_0x00400000_part_02.bin

0x0000000000,0x00080000,part_03,RPMB,RPMB_0x0000000000_0x00080000_part_03.bin

16.2.7 Attachment. Working with Partitions. Split, replace, record.

Reading partitions from chip to folder.

Insert chip, in "Miscellaneous " set "Analysis on ID - eMMC" and click "?" (Read ID).

If partitions are found, open "RW Mode" - "Partitions", select the working folder, tick the required sections, click "OK" and read. The folder will be filled with subtracted partitions.

If no partitions are found automatically, you can either upload a .UDEV file describing the partitions, or read a full dump and run the [Partition finder script](#).

Divides the dump into Partitions.

Select a full dump and run the «Partition finder» script. Then follow the prompts:

Button 'Read' - analyze dump from path 'Read to'.

Button 'Write' - search for file systems in dump from path 'Read to'.

Button 'Verify' - enter list of parts to extract.

Button 'Erase' - extract parts.

Button 'Blank Check' - split NAND dump from 'Read to' into main and spare.

Button 'Info' - print parts for extract.

Buttons 'Stop' or 'Execute script'- exit from script.

Following the hints, if you select R in the Read In field and the dump is read, and if everything is determined, we get a list of partitions and a .udev file based on the analysis in the dump folder.

Search for partitions in the dump.

Input File 'E:/temp/123/2/bbk40lex5026.User.bin'.

File size = 0xE9000000

Found MSTAR ANDROID marking.

Number of parts 18.

№ Offset Length Name

1 0x00000000 0x00200000 Partitions_table

2 0x00200000 0x00300000 MBOOT

3 0x00500000 0x00200000 MPOOL

4 0x00700000 0x00080000 misc

...

16 0xDB4C0000 0x04200000 tvcustomer

17 0xDF6C0000 0x06400000 factory

18 0xE5AC0000 0x03540000 gap_01

UDEV file created with the name

E:/temp/123/2/partitions_2021-05-05_14-03-55.udev

To extract, follow the hints press "V" to select partitions or immediately "X" to extract the full list.

Working with partitions in Explorer.

Replacing partitions in two dumps can be carried out in any convenient way as with files. For example, a regular conductor, or in the Total Commander, etc...

Writing partitions.

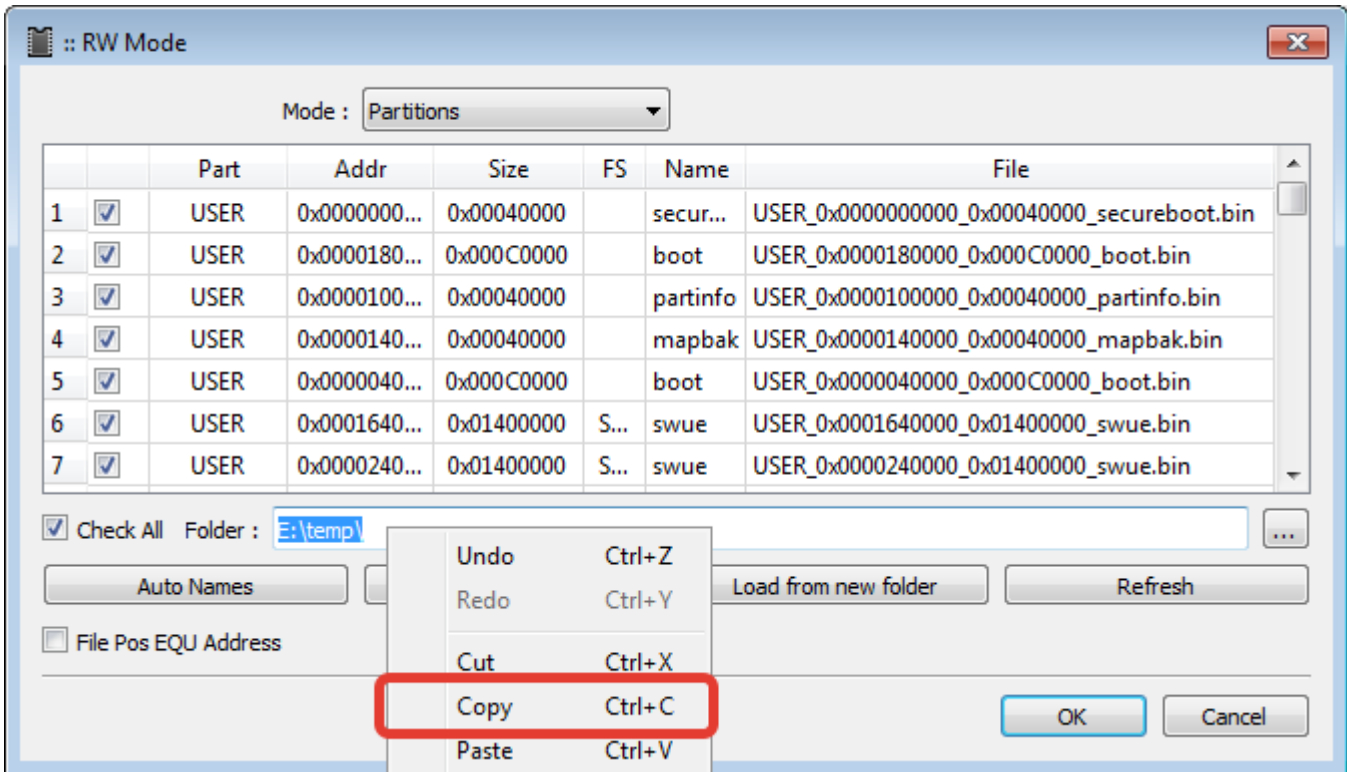
Open "RW Mode - Partitions", click "Load from New Folder", or select a working folder and click "Load from Current". Click OK and you can write. All partitions will be recorded sequentially. The auto complete fields is based on a consistent file name that specifies the location, address, and size.

Dump assembly from partitions.

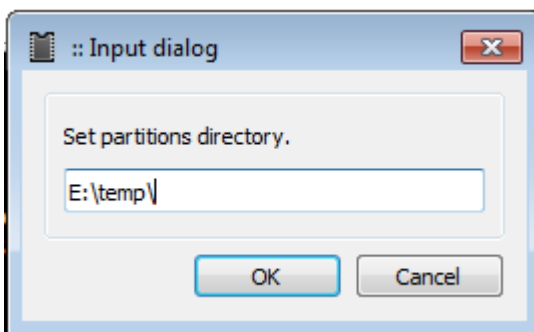
You can use a script to merge files.

You can use the auto-build script to build back into a single dump, if the partition name structure matches. This requires a working folder path that contains the partition files:

Run the script, enter the path of this folder, and click OK



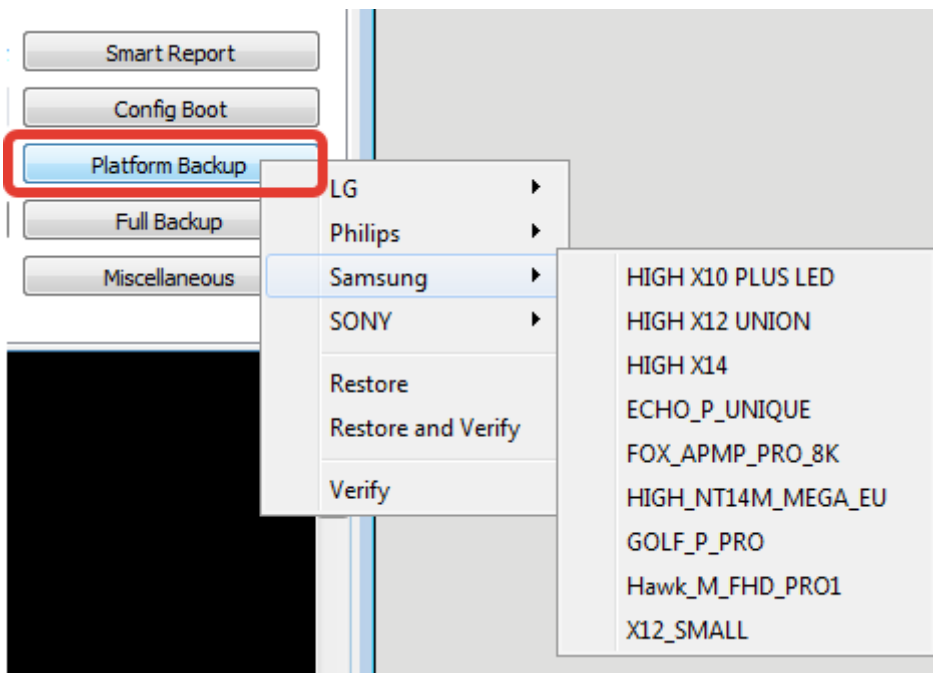
The script next to it will collect everything into one file in the FULL DUMP folder.



16.2.8 Attachment. Working with "Platform Backup".

After pressing the "?" button and reading the ID, the "Platform Backup" menu button becomes available.

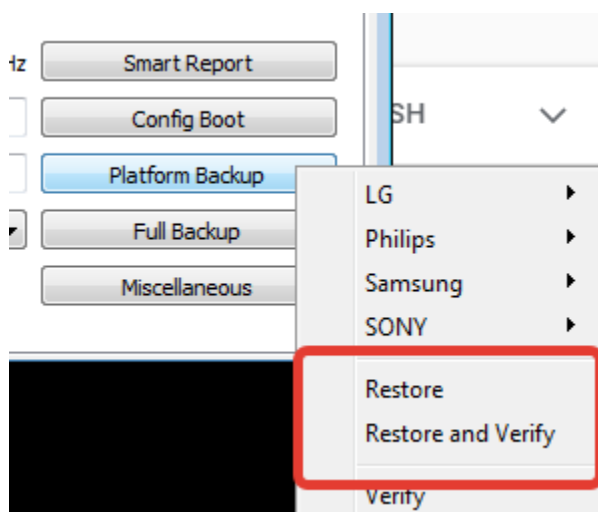
When you click the "Platform Backup" button, a menu appears that is based on the contents of .pbi files from the PBI folder in the UFPI program directory.



After selecting the platform, the specified data will be stored in the UBAK file.

```
eMMC Platform Backup for 'LC_LD_LE42B_42G'...
Data saving to File 'E:\temp\LC_LD_LE42B_42G.ubak'...OK
Reading eMMC USER from 0x03F00000, size 0x00080000...OK
Reading eMMC USER from 0x03F80000, size 0x00080000...OK
Completed in 0.06 second(s). Code download speed 15.87 MiB/sec.
```

When recovering from a UBAK file, all these parts will be written back



(You can choose Restore/Restore + Verification and the regions will be written. Or simply choice Verification of the attachment region.)

```
eMMC Platform Restore from 'E:\temp\LC_LD_LE42B_42G.ubak'...MMC Rev.1
Backup name 'LC_LD_LE42B_42G', size 1 MiB, created 16 Июнь 2021 10:32:41
```

```
Backup CID 110100303034474530001CA7F8DD3361 (Toshiba 004GE0, SN 0x1CA7F8DD (480770269), Mar 2016)
Backup CSD D05E00320F5903FFFFFFFFE7924000E3
Writing eMMC USER from 0x03F00000, size 0x00080000...OK
Writing eMMC USER from 0x03F80000, size 0x00080000...OK
Completed in 0.13 second(s). Code download speed 8 MiB/sec.
eMMC Platform Verification with 'LC_LD_LE42B_42G.ubak'...MMC Rev.1
Backup name 'LC_LD_LE42B_42G', size 1 MiB, created 16 Июнь 2021 10:32:41
Verifying eMMC USER from 0x03F00000, size 0x00080000...OK
Verifying eMMC USER from 0x03F80000, size 0x00080000...OK
Completed in 0.06 second(s). Code download speed 15.87 MiB/sec.
```

16.2.9 Attachment. Checking hashes, keys, restoring and extracting keys and BOOT using the example of the QV14/15 platform.

The mounting functionality acquires additional useful functions. Such as key verification, boot (see example QV14)

Well, for example, we have EMMC from the platform QV14/15. We turn on the analysis on ID reading (Settings - programmer - modules - SDMMC).

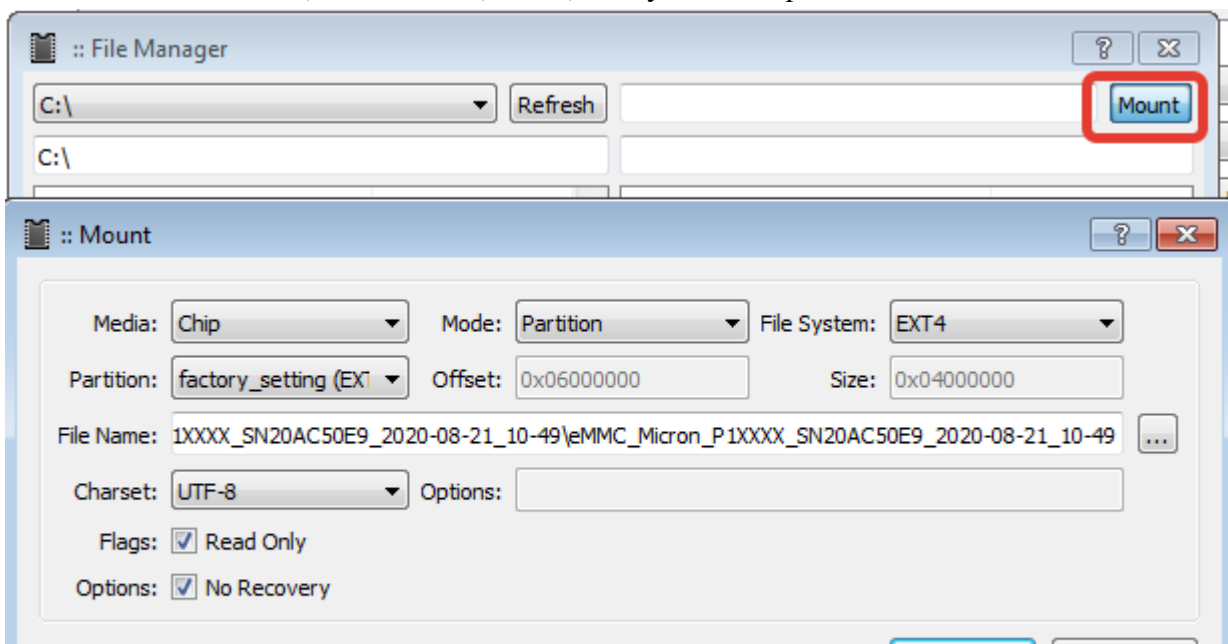
We define thechip and see the partitions available for mounting in the log:

```
eMMC Dump check...GPT Partition, offset 0x01
factory_setting EXT4
system EXT4
user_setting EXT4
cache EXT4
firmware EXT4
factory_data EXT4
userdata EXT4
eMMC Dump analysis done. Extracted 13 part(s)
```

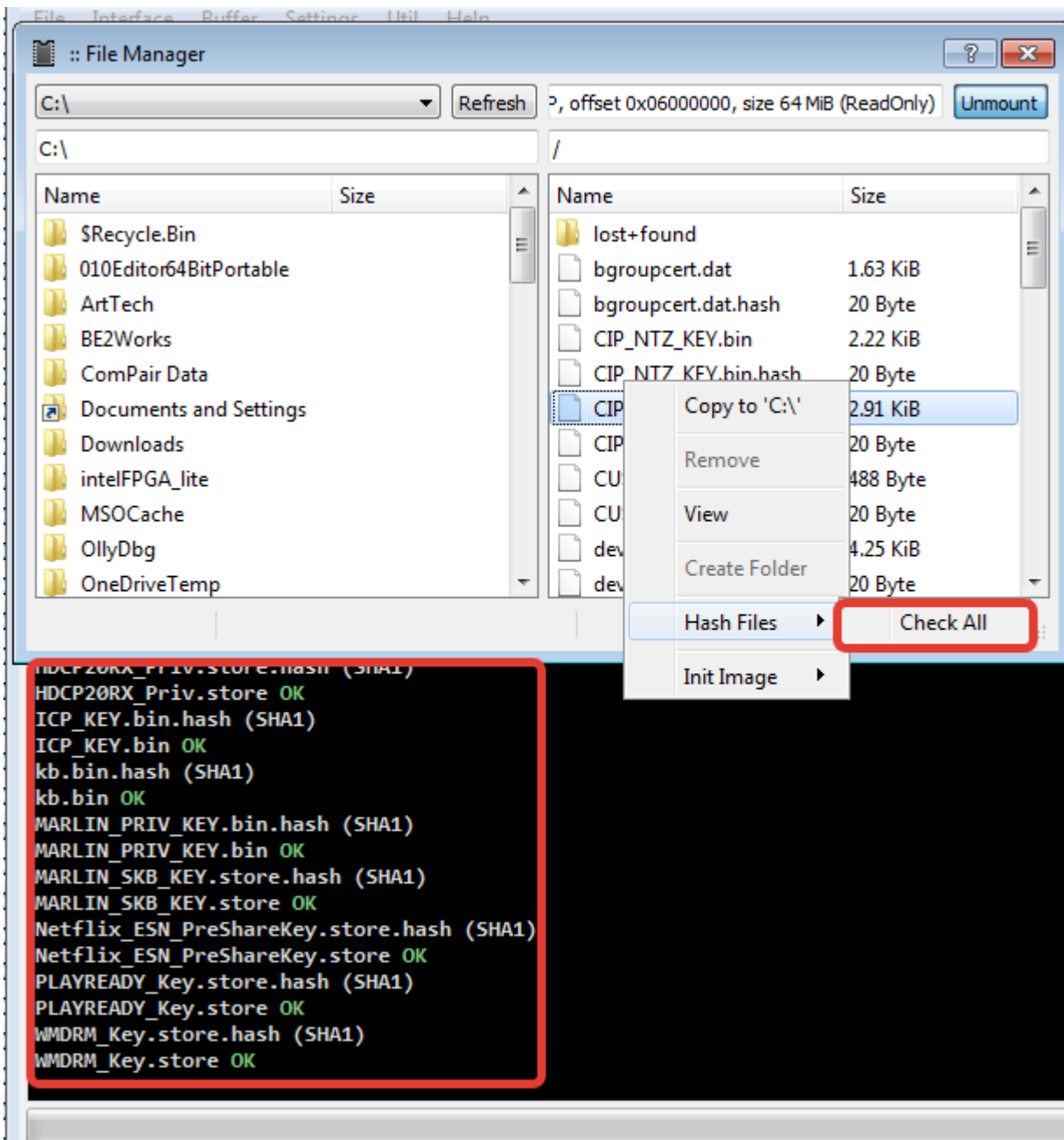
Click the File Manager button



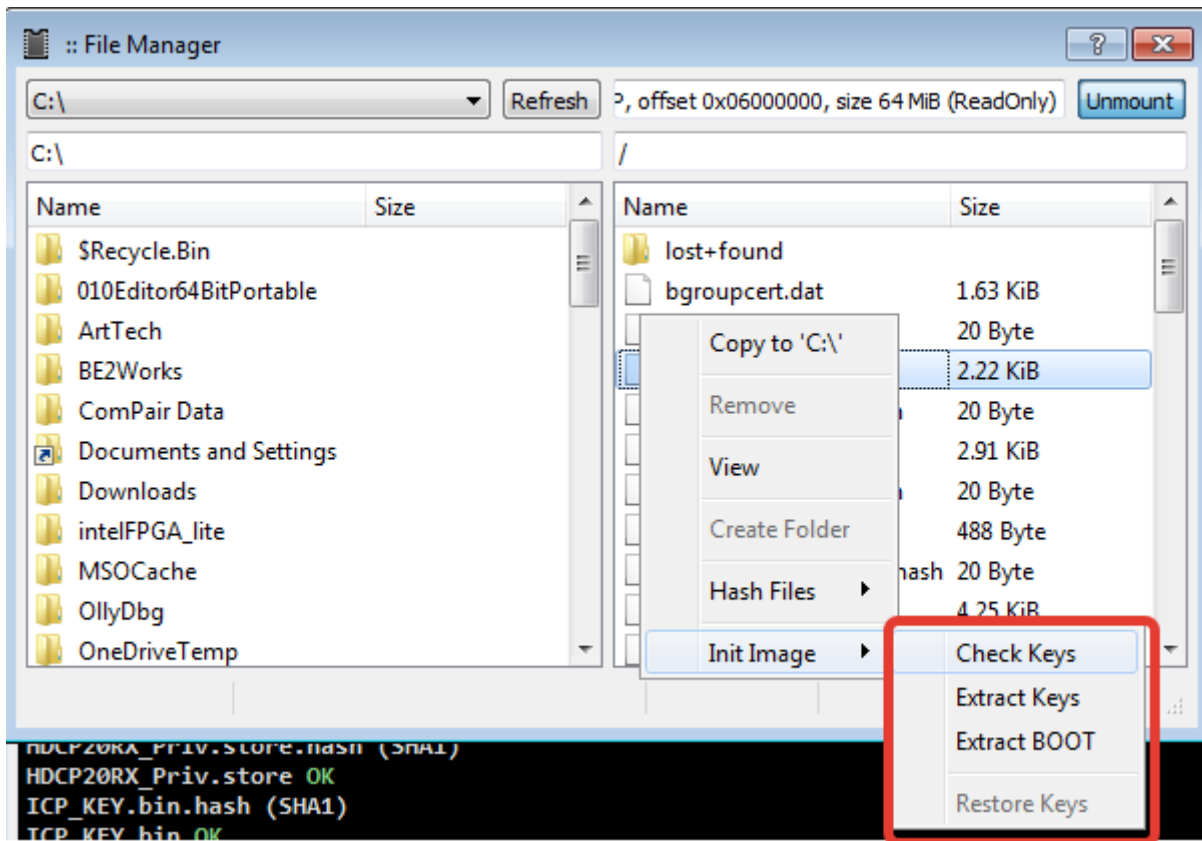
Press "mount" button, select media, mode, file system and partition.



Press "OK" and go to file manager with mounted section. You can then check the file hashes in the folder from the PCM menu.



You can check/extract boot and keys.



And the most interesting thing is that if you disactive the “Read-Only” tick box in the mount settings, then the remaining items are available and the damaged keys can be magically restored by the machine.

16.2.10 Attachment. Fix the dump and write the NAND dump using the example of Samsung D5500.

Correcting for errors and clearing of bad blocks in the dump.

Connect the programmer with the chip.

In ECC settings, select "Autodetect"

In the Util menu, click "Dump Analysis".

In the "Mode" field, select NAND with spare and Autodetect.

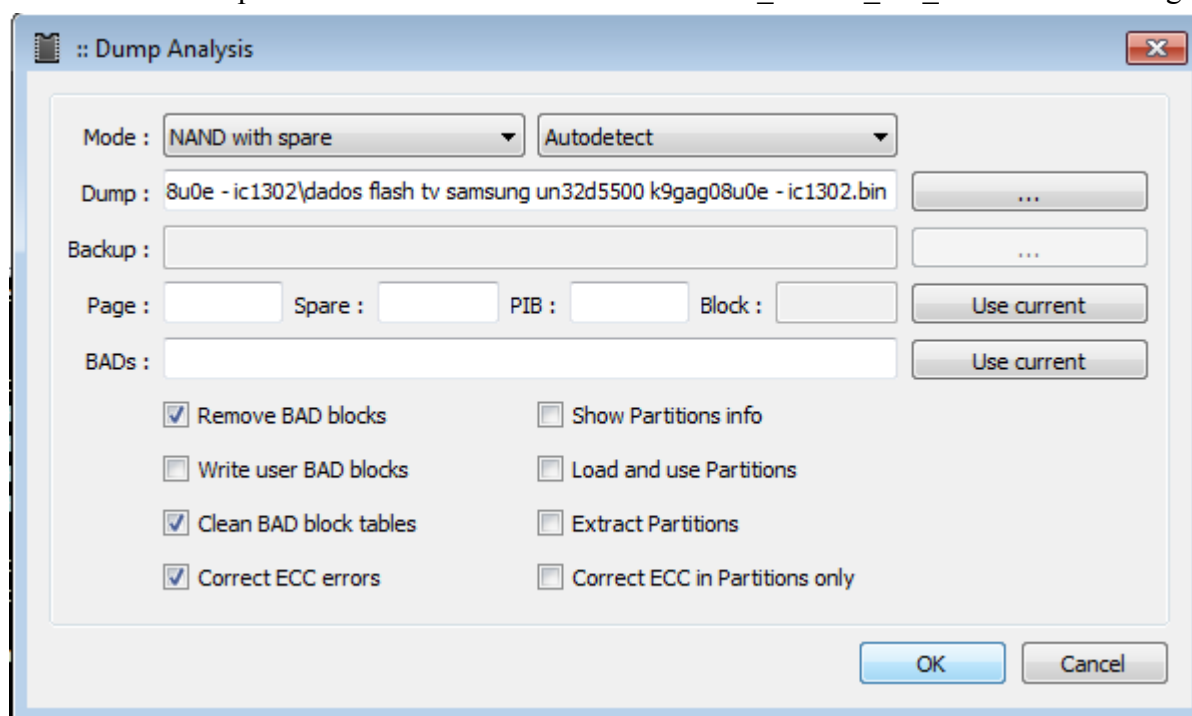
In the Dump field, select the dump to be corrected. The demonstration will be used by everyone known dump K9GAG08U0E from Dados for UN32D5500. *You must know that an incomplete dump read through ENT is often found on the network. This dump is not suitable for processing and use! The normal dump has the size "Total size with spare 0x88A7D800 2.13 Gib."*

Page, Spare, PIB is no need to fill. The dumps of these devices are determined automatically.

You do not need to set the "BADs" fields and the "Write user BAD blocks" check box.

Check "Remove BAD blocks," "Clear BAD block tables" and "Correct ECC errors" and click "OK".

The clean dump file will be saved with the name "fixed_nobads_ecc_clean.bin". The log will describe the



details of the dump BB table and the number of corrected errors.

Write "clean" dump with "Use reserved area" control.

Enable "Dump Analysis on ID Reading", "Asynchronous," and "Do not write blank data" in "Miscellaneous."

"

Enable BB detection by marker. "First byte token in spare" is 0x00.

BB Management set "Use spare zone," table type "Auto-determination."

ECC select a schema with Autodetect.

All these settings can be set by downloading the .UDEV file.

Select a "clean" dump to record.

Click "Read ID," check that there are no errors when analyzing the dump in the log.

```

NAND socket x8/x16, 4xCE, 3.30B
NAND ID ECD584725042 (ECD584725042ECD5), CE1 (*)
Manufacturer Samsung
Model name K9GAG08U0E
Bus and voltage x8 3.30V
Serial Access 30ns
Address Cycles 2/3 (Col/Row)
Page size 0x2000 (8192)
Spare size 0x1B4 (436)
Pages in Block 0x80 (128)
Block data size 0x100000 (1048576) 1 MiB
Block raw size 0x10DA00 (1104384) 1.05 MiB
Blocks count 0x81C (2076)
Bits per Cell 2 (MLC)
ECC Requirement 24/1024 (Bits/CW)
Dies per Chip 1
Chip (CE) count 1
Total data size 0x81C00000 2.03 GiB
Total raw size 0x88A7D800 2.13 GiB
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42 (no ECC errors in Page #0)
NAND BAD Blocks Table Samsung RFS (Block #0)
BAD blocks detection (Spare bytes #0 == 0x00)
BAD blocks management Use Reserved Area
BAD blocks request...OK
BAD blocks count: 2
Block 0x0714 (1812) addr 0x71400000 (0x77470800) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
NAND Dump, 256 bytes
00000000 46 53 52 5F 53 54 4C 00 01 01 02 01 FF FF FF FF | FSR_STL.....ÿÿÿÿ
00000010 FE FF FF FF 00 08 00 12 00 00 01 00 01 00 02 00 | þÿÿÿ.....
00000020 01 00 10 00 00 00 04 00 01 00 00 00 01 00 00 00 | .....
00000030 02 00 D4 00 04 00 01 01 C9 00 0A 00 07 00 C9 00 | ..Ô.....É.....É.
00000040 00 48 06 00 80 64 00 00 01 00 00 02 09 00 00 00 | .H...d.....
00000050 02 00 03 00 04 00 05 00 FF FF FF FF FF FF FF FF | .....ÿÿÿÿÿÿÿÿ
00000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
NAND Dump '***dados flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin' Analysis, size
0x88A7D800...
NAND Page 8192+436 (0x21B4), PIB 128, Block 0x10DA00 (1104384), 0x081C (2076) Blocks, IC geometry
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42
NAND BAD Blocks Table Samsung RFS, Block 0x07FD (2045)
NAND Partition Info Samsung RFS, Block 0x07FE (2046)
NAND LPCH Block 0x07FE (2046) at 0x86AE4C00
NAND LPCH TPCB 0x07FC (2044), Ages 1-1
NAND UPCH Block 0x07FD (2045) at 0x869D7200, 1 record(s) (Active)

```

```

NAND UPCH TPCB 0x07FC (2044), Ages 1-2, инфo 0x0000
NAND RBA 0x07FB-0x0785 (2043-1925), 0x77 (119) Blocks
NAND Partition #1, 'part-00-ID1B', Block 0x0 (0), count 0xD6 (214)
NAND Partition #1, addr 0x00000000 (0x00000000), size 0x0D600000 (0x0E163C00)
NAND Partition #2, 'part-01-ID1C', Block 0xD6 (214), count 0x100 (256)
NAND Partition #2, addr 0x0D600000 (0x0E163C00), size 0x10000000 (0x10DA0000)
NAND Partition #3, 'part-02-ID1D', Block 0x1D6 (470), count 0xD6 (214)
NAND Partition #3, addr 0x1D600000 (0x1EF03C00), size 0x0D600000 (0x0E163C00)
NAND Partition #4, 'part-03-ID1E', Block 0x2AC (684), count 0x410 (1040)
NAND Partition #4, addr 0x2AC00000 (0x2D067800), size 0x41000000 (0x4475A000)
NAND Partition #5, 'part-04-ID1F', Block 0x6BC (1724), count 0xC8 (200)
NAND Partition #5, addr 0x6BC00000 (0x717C1800), size 0x0C800000 (0x0D2A5000)
NAND Partition #6, 'UPCH', Block 0x7FD (2045), count 0x1 (1)
NAND Partition #6, addr 0x7FD00000 (0x869D7200), size 0x00100000 (0x0010DA00)
NAND Partition #7, 'LPCH', Block 0x7FE (2046), count 0x1 (1)
NAND Partition #7, addr 0x7FE00000 (0x86AE4C00), size 0x00100000 (0x0010DA00)
Loading IC BAD Blocks list...2 BAD Block(s)
Writing new BAD Blocks list...
NAND BAD Block 0x0714 (1812) replaced with 0x07FB (2043)
NAND BAD Block 0x0819 (2073) ignored
Encoding ECC in the patched pages...MSTAR_P8K_SP436_CW8_S12L42
BAD blocks management Use Reserved Area

```

If necessary, enter a list of additional bad blocks via "Miscellaneous "-" BAD blocks "-" Mark BADs "and read ID again.

Check the BB list. Dump analysis will configure the programmer to continue working with the chip and dump.

Erase the chip if the "Erase Before Recording" check box is not set ON in "Miscellaneous "

Write, verify. Dump analysis in the log has no errors.

```

Data source File '***dados flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin'...Normal
User task flags Async IO, DNWRB, PACKUSB
Стирание NAND from 0x00000000, size 0x88A7D800...
Block 0x07FB (2043) addr 0x7FB00000 (0x867BBE00) Пропуск!
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) Пропуск!
Запись NAND from 0x00000000, size 0x88A7D800...
Completed in 1 min. 38 second(s) Code download speed 22.12 MiB/sec
*****
Data source File '***dados flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin'...Normal
User task flags Async IO, ECC
Verifying NAND from 0x00000000, size 0x88A7D800...
Block 0x00DE (222) addr 0x0DE00000 (0x0E9D0C00) ECC, 1 bit(s) corrected//these are common bit errors in
page (s), they have correction codes and are corrected.
...
Page 0xED0C (60684), Block 0x01DA (474) addr 0x1DA18000 (0x1F353870) Uncorrectable ECC error!// this is an
error in the data page, with which the ECC correction algorithm failed. Blocks with such errors should be
marked as BAD.
...
Block 0x02BF (703) addr 0x2BF00000 (0x2E46A600) 1 Bit error(s) in 1 page(s)! // this is an error in a
blank page, similar ones do not have a correction code and cannot be fixed.
...
Block 0x06EA (1770) addr 0x6EA00000 (0x74834400) ECC, 4 bit(s) corrected
Block 0x07FB (2043) addr 0x7FB00000 (0x867BBE00) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
ECC checked in 36766 page(s), 228706 blank page(s) skipped

```

Corrected 193 ECC Bit error(s), 5 in ECC data

Completed in 3 min. 52 second(s) Code download speed 9.42 MiB/sec

If uncorrectable errors are found during the verification process, add additional bad blocks, re-read the ID and re-write.

If r old flash is used, with erased old BB markers, then it takes additional time to identify and mark its bad blocks. To do this, after recording, postpone the flash for several days, the longer, the more clearly bad blocks will appear.

Additional check.

After a few days insert into the programmer, set all ECC, BB (or load .UDEV) and dump settings as described above. In the module settings, set "Do not stop verification for errors" and for clarity you can set "Show errors for whole blocks."

Read ID.

Start verification and check for bads with uncorrectable errors.

NAND socket x8/x16, 4xCE, 3.30B

NAND ID ECD584725042 (ECD584725042ECD5), CE1 (*)

Manufacturer Samsung

Model name K9GAG08U0E

Bus and voltage x8 3.30V

Serial Access 30ns

Address Cycles 2/3 (Col/Row)

Page size 0x2000 (8192)

Spare size 0x1B4 (436)

Pages in Block 0x80 (128)

Block data size 0x100000 (1048576) 1 MiB

Block raw size 0x10DA00 (1104384) 1.05 MiB

Blocks count 0x81C (2076)

Bits per Cell 2 (MLC)

ECC Requirement 24/1024 (Bits/CW)

Dies per Chip 1

Chip (CE) count 1

Total data size 0x81C00000 2.03 GiB

Total raw size 0x88A7D800 2.13 GiB

NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42 (no ECC errors in Page #0)

NAND BAD Blocks Table Samsung RFS (Block #0)

BAD blocks detection (Spare bytes #0 == 0x00)

BAD blocks management Use Reserved Area

BAD blocks request...OK

BAD blocks count: 20

Block 0x00D8 (216) addr 0x0D800000 (0x0E37F000) BAD Block

Block 0x00DF (223) addr 0x0DF00000 (0x0EAEDE600) BAD Block

Block 0x01DA (474) addr 0x1DA00000 (0x1F33A400) BAD Block

Block 0x01DB (475) addr 0x1DB00000 (0x1F447E00) BAD Block

Block 0x02B1 (689) addr 0x2B100000 (0x2D5ABA00) BAD Block

Block 0x02B7 (695) addr 0x2B700000 (0x2DBFD600) BAD Block

Block 0x02BA (698) addr 0x2BA00000 (0x2DF26400) BAD Block

Block 0x02BC (700) addr 0x2BC00000 (0x2E141800) BAD Block

Block 0x02BD (701) addr 0x2BD00000 (0x2E24F200) BAD Block

Block 0x02C1 (705) addr 0x2C100000 (0x2E685A00) BAD Block

Block 0x033D (829) addr 0x33D00000 (0x3691F200) BAD Block

Block 0x036B (875) addr 0x36B00000 (0x39991E00) BAD Block

```

Block 0x0371 (881) addr 0x37100000 (0x39FE3A00) BAD Block
Block 0x0379 (889) addr 0x37900000 (0x3A850A00) BAD Block
Block 0x037D (893) addr 0x37D00000 (0x3AC87200) BAD Block
Block 0x06BE (1726) addr 0x6BE00000 (0x719DCC00) BAD Block
Block 0x06C0 (1728) addr 0x6C000000 (0x71BF8000) BAD Block
Block 0x06C1 (1729) addr 0x6C100000 (0x71D05A00) BAD Block
Block 0x0714 (1812) addr 0x71400000 (0x77470800) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
NAND Dump, 256 bytes
00000000 46 53 52 5F 53 54 4C 00 01 01 02 01 FF FF FF FF | FSR_STL....ÿÿÿÿ
00000010 FE FF FF FF 00 08 00 12 00 00 01 00 01 00 02 00 | þÿÿÿ.....
00000020 01 00 10 00 00 00 04 00 01 00 00 00 01 00 00 00 | .....
00000030 02 00 D4 00 04 00 01 01 C9 00 0A 00 07 00 C9 00 | ..Ô.....É.....É.
00000040 00 48 06 00 80 64 00 00 01 00 00 02 09 00 00 00 | .H...d.....
00000050 02 00 03 00 04 00 05 00 FF FF FF FF FF FF FF FF | .....ÿÿÿÿÿÿÿÿ
00000060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
NAND Dump 'E:\temp\damp\Новая папка\SAMSUNG\D55xx\dados flash tv samsung un32d5500 k9gag08u0e - ic1302\
CLEAR\dados
flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin' Analysis, size 0x88A7D800...
NAND Page 8192+436 (0x21B4), PIB 128, Block 0x10DA00 (1104384), 0x081C (2076) Blocks, IC geometry
NAND ECC correction MSTAR_P8K_SP436_CW8_S12L42
NAND BAD Blocks Table Samsung RFS, Block 0x07FD (2045)
NAND Partition Info Samsung RFS, Block 0x07FE (2046)
NAND LPCH Block 0x07FE (2046) at 0x86AE4C00
NAND LPCH TPCB 0x07FC (2044), Ages 1-1
NAND UPCH Block 0x07FD (2045) at 0x869D7200, 1 record(s) (Active)
NAND UPCH TPCB 0x07FC (2044), Ages 1-2, инфo 0x0000
NAND RBA 0x07FB-0x0785 (2043-1925), 0x77 (119) Blocks
NAND Partition #1, 'part-00-ID1B', Block 0x0 (0), count 0xD6 (214)
NAND Partition #1, addr 0x00000000 (0x00000000), size 0x0D600000 (0x0E163C00)
NAND Partition #2, 'part-01-ID1C', Block 0xD6 (214), count 0x100 (256)
NAND Partition #2, addr 0x0D600000 (0x0E163C00), size 0x10000000 (0x10DA0000)
NAND Partition #3, 'part-02-ID1D', Block 0x1D6 (470), count 0xD6 (214)
NAND Partition #3, addr 0x1D600000 (0x1EF03C00), size 0x0D600000 (0x0E163C00)
NAND Partition #4, 'part-03-ID1E', Block 0x2AC (684), count 0x410 (1040)
NAND Partition #4, addr 0x2AC00000 (0x2D067800), size 0x41000000 (0x4475A000)
NAND Partition #5, 'part-04-ID1F', Block 0x6BC (1724), count 0xC8 (200)
NAND Partition #5, addr 0x6BC00000 (0x717C1800), size 0x0C800000 (0x0D2A5000)
NAND Partition #6, 'UPCH', Block 0x7FD (2045), count 0x1 (1)
NAND Partition #6, addr 0x7FD00000 (0x869D7200), size 0x00100000 (0x0010DA00)
NAND Partition #7, 'LPCH', Block 0x7FE (2046), count 0x1 (1)
NAND Partition #7, addr 0x7FE00000 (0x86AE4C00), size 0x00100000 (0x0010DA00)
Loading IC BAD Blocks list...20 BAD Block(s)
Writing new BAD Blocks list...

```

```

NAND BAD Block 0x00D8 (216) replaced with 0x07FB (2043)
NAND BAD Block 0x00DF (223) replaced with 0x07FA (2042)
NAND BAD Block 0x01DA (474) replaced with 0x07F9 (2041)
NAND BAD Block 0x01DB (475) replaced with 0x07F8 (2040)
NAND BAD Block 0x02B1 (689) replaced with 0x07F7 (2039)
NAND BAD Block 0x02B7 (695) replaced with 0x07F6 (2038)
NAND BAD Block 0x02BA (698) replaced with 0x07F5 (2037)
NAND BAD Block 0x02BC (700) replaced with 0x07F4 (2036)
NAND BAD Block 0x02BD (701) replaced with 0x07F3 (2035)
NAND BAD Block 0x02C1 (705) replaced with 0x07F2 (2034)
NAND BAD Block 0x033D (829) replaced with 0x07F1 (2033)
NAND BAD Block 0x036B (875) replaced with 0x07F0 (2032)
NAND BAD Block 0x0371 (881) replaced with 0x07EF (2031)
NAND BAD Block 0x0379 (889) replaced with 0x07EE (2030)
NAND BAD Block 0x037D (893) replaced with 0x07ED (2029)
NAND BAD Block 0x06BE (1726) replaced with 0x07EC (2028)
NAND BAD Block 0x06C0 (1728) replaced with 0x07EB (2027)
NAND BAD Block 0x06C1 (1729) replaced with 0x07EA (2026)
NAND BAD Block 0x0714 (1812) replaced with 0x07E9 (2025)
NAND BAD Block 0x0819 (2073) ignored
Encoding ECC in the patched pages...MSTAR_P8K_SP436_CW8_S12L42
BAD blocks management Use Reserved Area
*****2021.05.18 01:17:56:1756*****
Data source File 'E:\temp\damp\Новая папка\SAMSUNG\D55xx\dados flash tv samsung un32d5500 k9gag08u0e -
ic1302\CLEAR\dados flash tv samsung un32d5500 k9gag08u0e - ic1302_fix_nobads.bin'...Normal
User task flags Async IO, ECC, SADDR
Verifying NAND from 0x00000000, size 0x88A7D800...
Block 0x0006 (6) addr 0x00600000 (0x00651C00) ECC, 2 bit(s) corrected
Block 0x00DB (219) addr 0x0DB00000 (0x0E6A7E00) 2 Bit error(s) in 1 page(s)!
Block 0x00DE (222) addr 0x0DE00000 (0x0E9D0C00) ECC, 2 bit(s) corrected
Block 0x00E0 (224) addr 0x0E000000 (0x0EBEC000) ECC, 1 bit(s) corrected
Block 0x01D9 (473) addr 0x1D900000 (0x1F22CA00) 2 Bit error(s) in 2 page(s)!
Block 0x01DB (475) addr 0x1DB00000 (0x1F447E00) 1 Bit error(s) in 1 page(s)!
Block 0x01DC (476) addr 0x1DC00000 (0x1F555800) ECC, 2 bit(s) corrected
...
Block 0x0216 (534) addr 0x21600000 (0x2326BC00) ECC, 1 bit(s) corrected
Block 0x0218 (536) addr 0x21800000 (0x23487000) ECC, 2 bit(s) corrected
Block 0x02B1 (689) addr 0x2B100000 (0x2D5ABA00) 1 Bit error(s) in 1 page(s)!
Block 0x02B7 (695) addr 0x2B700000 (0x2DBFD600) 3 Bit error(s) in 3 page(s)!
Block 0x02BA (698) addr 0x2BA00000 (0x2DF26400) 3 Bit error(s) in 3 page(s)!
Block 0x02BC (700) addr 0x2BC00000 (0x2E141800) 3 Bit error(s) in 3 page(s)!
Block 0x02BD (701) addr 0x2BD00000 (0x2E24F200) 3 Bit error(s) in 3 page(s)!
Block 0x02BF (703) addr 0x2BF00000 (0x2E46A600) 6 Bit error(s) in 6 page(s)!
Page 0x1608C (90252), Block 0x02C1 (705) addr 0x2C118000 (0x2E69EE70) Uncorrectable ECC error!
Block 0x02C2 (706) addr 0x2C200000 (0x2E793400) ECC, 1 bit(s) corrected
...
Block 0x0390 (912) addr 0x39000000 (0x3C08A000) ECC, 3 bit(s) corrected
Block 0x0392 (914) addr 0x39200000 (0x3C2A5400) ECC, 3 bit(s) corrected
Block 0x0393 (915) addr 0x39300000 (0x3C3B2E00) ECC, 6 bit(s) corrected
Page 0x3600C (221196), Block 0x06C0 (1728) addr 0x6C018000 (0x71C11470) Uncorrectable ECC error!
Block 0x06C0 (1728) addr 0x6C000000 (0x71BF8000) 1 Bit error(s) in 1 page(s)!
...
Block 0x06ED (1773) addr 0x6ED00000 (0x74B5D200) ECC, 4 bit(s) corrected

```



```

Block 0x07E9 (2025) addr 0x7E900000 (0x854C6A00) BAD Block
Block 0x07EA (2026) addr 0x7EA00000 (0x855D4400) BAD Block
Block 0x07EB (2027) addr 0x7EB00000 (0x856E1E00) BAD Block
Block 0x07EC (2028) addr 0x7EC00000 (0x857EF800) BAD Block
Block 0x07ED (2029) addr 0x7ED00000 (0x858FD200) BAD Block
Block 0x07EE (2030) addr 0x7EE00000 (0x85A0AC00) BAD Block
Block 0x07EF (2031) addr 0x7EF00000 (0x85B18600) BAD Block
Block 0x07F0 (2032) addr 0x7F000000 (0x85C26000) BAD Block
Block 0x07F1 (2033) addr 0x7F100000 (0x85D33A00) BAD Block
Block 0x07F2 (2034) addr 0x7F200000 (0x85E41400) BAD Block
Block 0x07F3 (2035) addr 0x7F300000 (0x85F4EE00) BAD Block
Block 0x07F4 (2036) addr 0x7F400000 (0x8605C800) BAD Block
Block 0x07F5 (2037) addr 0x7F500000 (0x8616A200) BAD Block
Block 0x07F6 (2038) addr 0x7F600000 (0x86277C00) BAD Block
Block 0x07F7 (2039) addr 0x7F700000 (0x86385600) BAD Block
Block 0x07F8 (2040) addr 0x7F800000 (0x86493000) BAD Block
Block 0x07F9 (2041) addr 0x7F900000 (0x865A0A00) BAD Block
Block 0x07FA (2042) addr 0x7FA00000 (0x866AE400) BAD Block
Block 0x07FB (2043) addr 0x7FB00000 (0x867BBE00) BAD Block
Block 0x0819 (2073) addr 0x81900000 (0x88754A00) BAD Block
ECC checked in 36768 page(s), 226376 blank page(s) skipped, 14 blank codeword(s), 192 codeword(s) with
blank ECC
Corrected 992 ECC Bit error(s), 56 in ECC data
Detected 2 uncorrectable ECC error(s) in 2 page(s)
Total Bit errors count during verification: 25
Completed in 3 min. 22 second(s) Code download speed 10.78 МБ/сек

```

The log contains blocks with uncorrectable errors. Such blocks must then be marked as "Bad Blocks" in "Miscellaneous." If there are more than one, you can list their numbers using a comma.

During verification, the BB bypass using the reserved area is enabled and need to know that in the verification log the block 705 in which uncorrectable errors are present is already marked as BB. In this case, 2034 blocks protrude behind the 705 block. This is described above with the bad block list. This needs to be additionally labeled as a BB.

You can also set the "Show corrected ECC errors" option in the module settings and see blocks with a particularly large number of corrected errors in the log. These are also candidates for adding to BB list, if there are more than 20-30 errors.

Read ID again, erase and write.

16.2.11 Attachment. Package. Build and writing.

Purpose.

For example, the BIOS dump is taken from the laptop and you need program in SPI flash without erasing the Windows key.

Preparing.

Finding the key offset, dump is divided into three parts with dimensions multiples of the sector. The sector in the SPI is the minimum erasable part. For example, if it were NAND flash, it would be blocks.

Two regions are cut - before Winkey and after it, and then saved as separate files with the .bin extension from the dump to be written. All regions must be multiples of the minimum erasable size - block.

Example:

0x000000-0x41F000 – region to key. Save to a separate file.

0x41F000-0x423000 – region with key

0x423000-0x800000 – region after key. Save to a separate file.

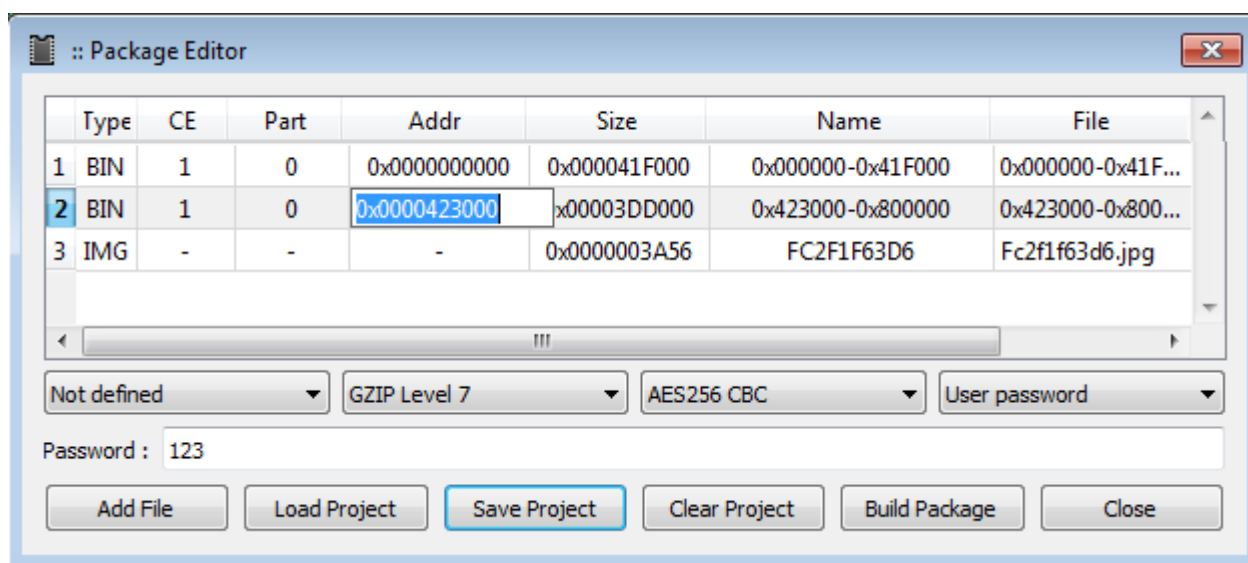
Creating the PROJECT

Open Tools - Package Editor.

Click Add File and select the first file in the 0x000000-0x41F000 region in order. Leave the address and location values in the table because they match the actual values.

Click Add File again and select the second region file 0x423000-0x800000.

Double-click the "Addresses" value of the second file and set 0x423000.



You can then add a picture, a .udev file, or a text document to open when the Package is opened..

Set the chip type for SPI as Undefined.

You can immediately select the compression level for the archiver so that the file takes up less space. In the picture, "GZIP level 7" is selected.

You can encrypt and set a password for protection.

As all settings are completed and parts are added, you must save the .upkj project by clicking the appropriate button.

After you save the project, you can modify it, or you can create a Package with the appropriate button.

Working with the Package.

You can start the Package either through "File - Open Package" or by clicking the "Open File To Write" button by selecting the .upkg file type.

To record regions from the Package, you need to read the ID flash, and with the usual buttons click erase and record. All operations will be performed only with regions of files installed in the Package, and the region will not be affected 0x41F000-0x423000.

When finished, close the Package through the file menu. Alternatively, the Package itself closes if you select another file to write.

IMPORTANT to know.

To avoid errors, all parts of the files and the project must be placed in the same folder.

When you open a Package with .udev added file, its settings are automatically applied. For example, you set the RW Mode to "Partitions" with a list of partitions included. The working folder with partitions must be next to the .upkg Package file.

When you open a Package containing a .txt file, its contents are logged. It could be a description.

If a picture has been added to the Package, it opens parallel to the program window. This may be an illustration or a diagram.

16.2.12 Attachment. Using LOGGER and protocol analysis.

Under construction...

16.3 Pinouts and diagrams.

Actual information is available [HERE](#)

UFPI IO socket pinout:

PIN01 - IO00

PIN02 - IO01

...

PIN24 - IO23

PIN25 - ID1 (Module. Resistor between
PIN25 and GND)

PIN26 - ID2 (Voltage, resistor between
PIN26 and GND)

PIN27 - VCC/VREF

PIN28 - VUSB (+5V)

PIN29 - GND

PIN30 - GND

ID1 (Module) resistors values:

JTAG - 0R

NAND - 1K

SPI Flash - 2K

EEPROM I2C - 3K

EEPROM SPI - 4K

EEPROM mWire - 5K

Serial NAND - 7K1

1-Wire - 9K1

OneNAND - 20K

NOR - 24K

SD/eMMC - 30K

E-Mate - 33K

UART - 51K

LOGGER - 68K

BDM - 75K

ID2 (Voltage) resistors values:

VREF - 0R или 8K2 (EXT power supply)

3V3 - 1K

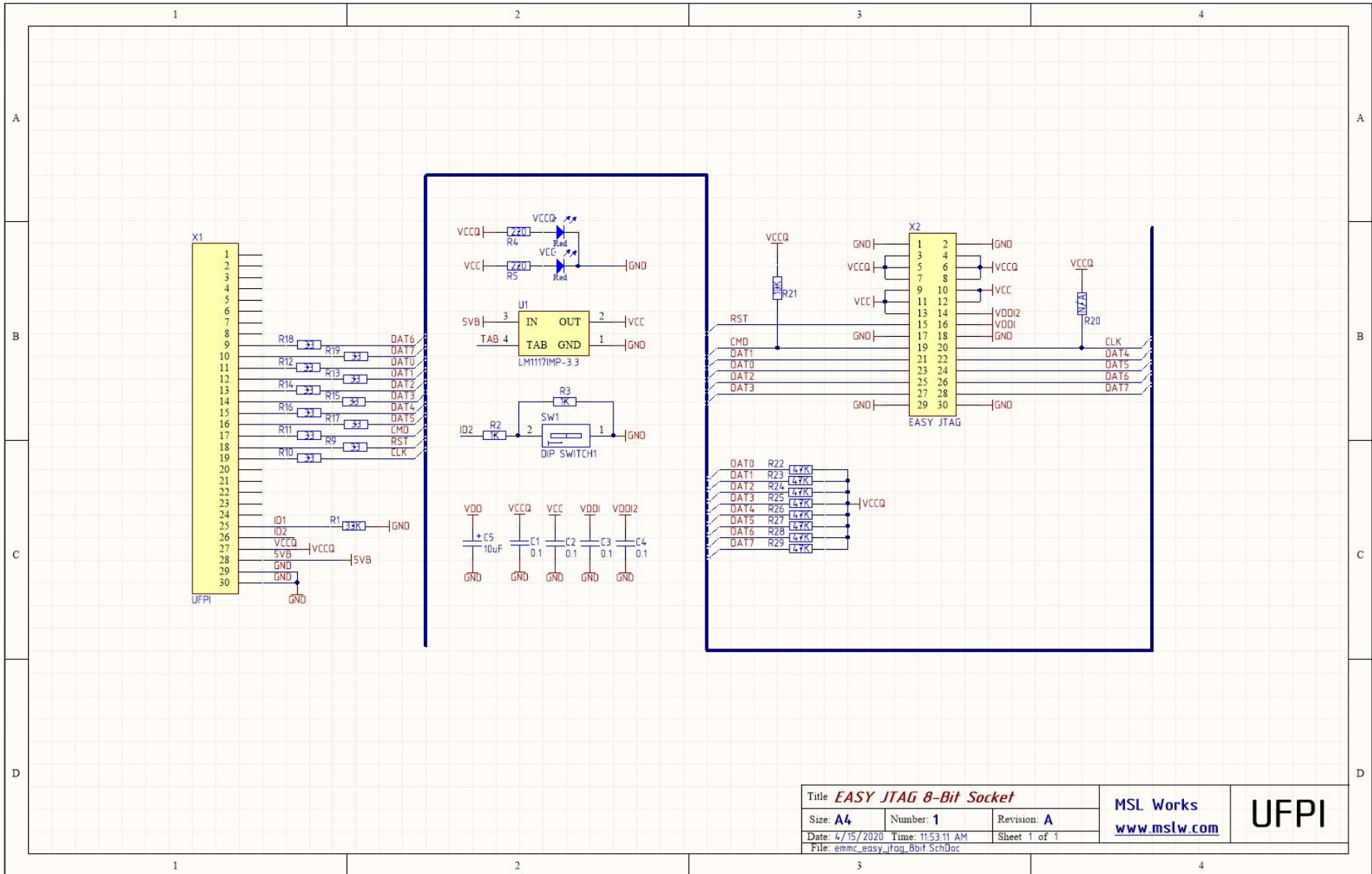
1V8 - 2K

2V6 - 3K

5V0 - 9K1



16.3.1 eMMC EASY JTAG 8-Bit Socket schematics

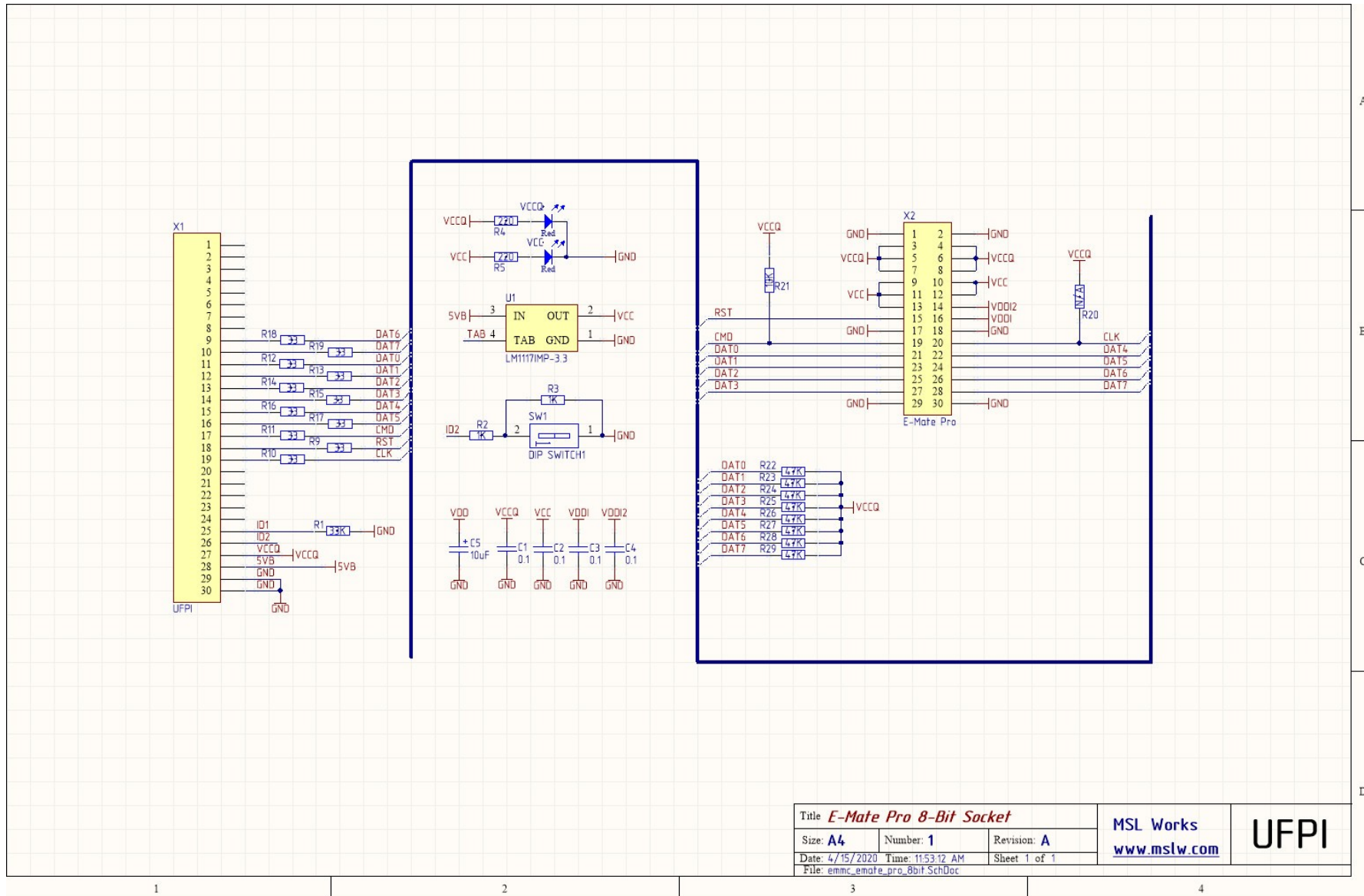


Title EASY JTAG 8-Bit Socket		
Size: A4	Number: 1	Revision: A
Date: 4/15/2020	Time: 11:53:11 AM	Sheet 1 of 1
File: emmc_easy_jtag_8bit_SchDoc		

MSL Works
www.mslw.com

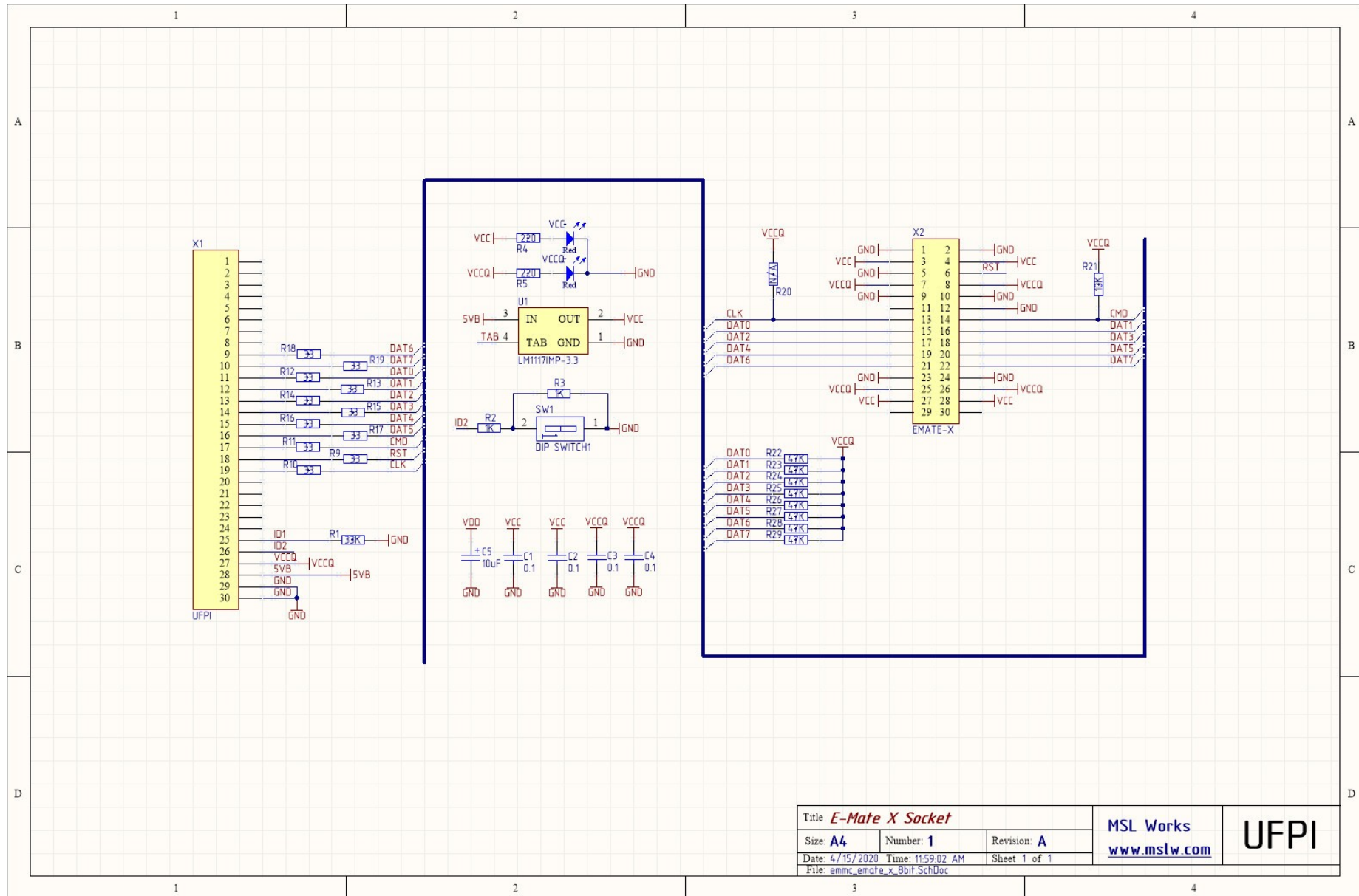
UFPI

16.3.2 eMMC E-Mate Pro 8-Bit Socket schematics



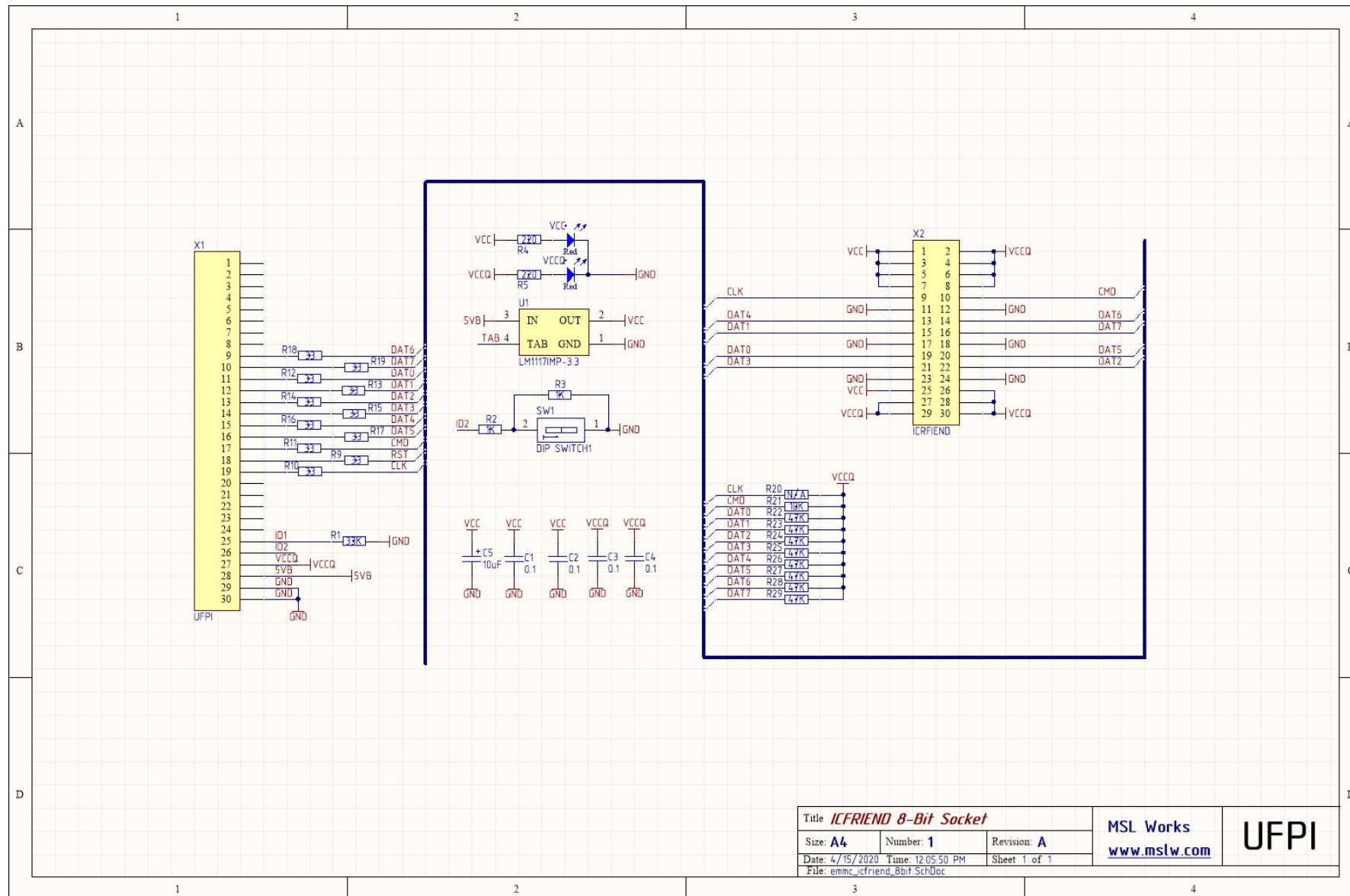
Title <i>E-Mate Pro 8-Bit Socket</i>			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: A		
Date: 4/15/2020	Time: 11:53:12 AM	Sheet 1 of 1		
File: emmc_emate_pro_8bit SchDoc				

16.3.3 eMMC E-Mate X 8-Bit Socket schematics



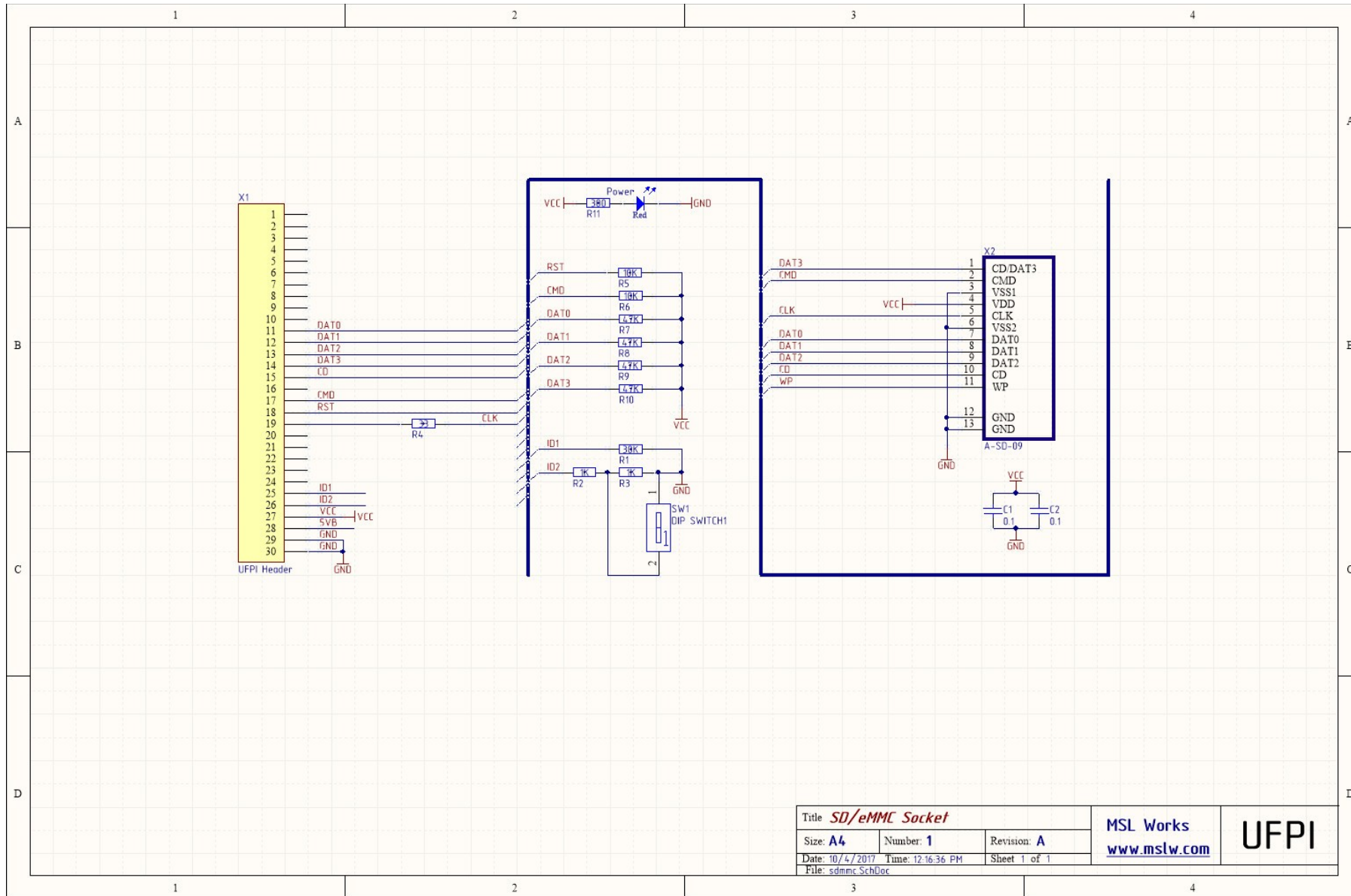
Title <i>E-Mate X Socket</i>			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: A		
Date: 4/15/2020	Time: 11:59:02 AM	Sheet 1 of 1		
File: emmc_emate_x_8bit SchDoc				

16.3.4 eMMC ICFRIEND 8-Bit Socket schematics



Title <i>ICFRIEND 8-Bit Socket</i>		MSL Works www.mslw.com	UFPI
Size: A4	Number: 1		
Date: 4/15/2020 Time: 12:05:50 PM		Sheet 1 of 1	
File: emmc_icfriend_8bit.SchDoc			

16.3.5 SD/eMMC Socket schematics



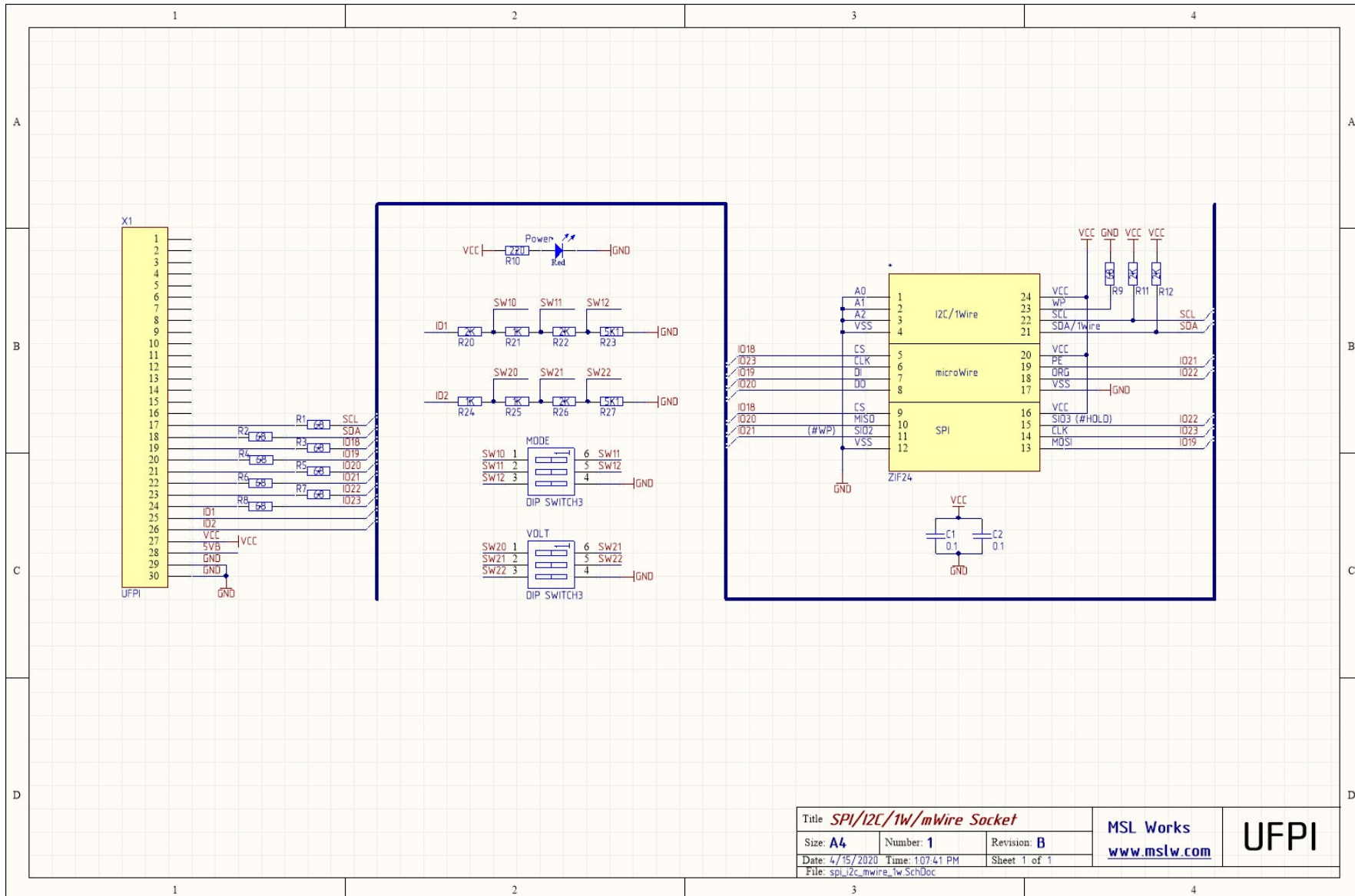
Title *SD/eMMC Socket*

Size: **A4** Number: **1** Revision: **A**
 Date: 10/4/2017 Time: 12:16:36 PM Sheet 1 of 1
 File: sdmmc.SchDoc

MSL Works
www.mslw.com

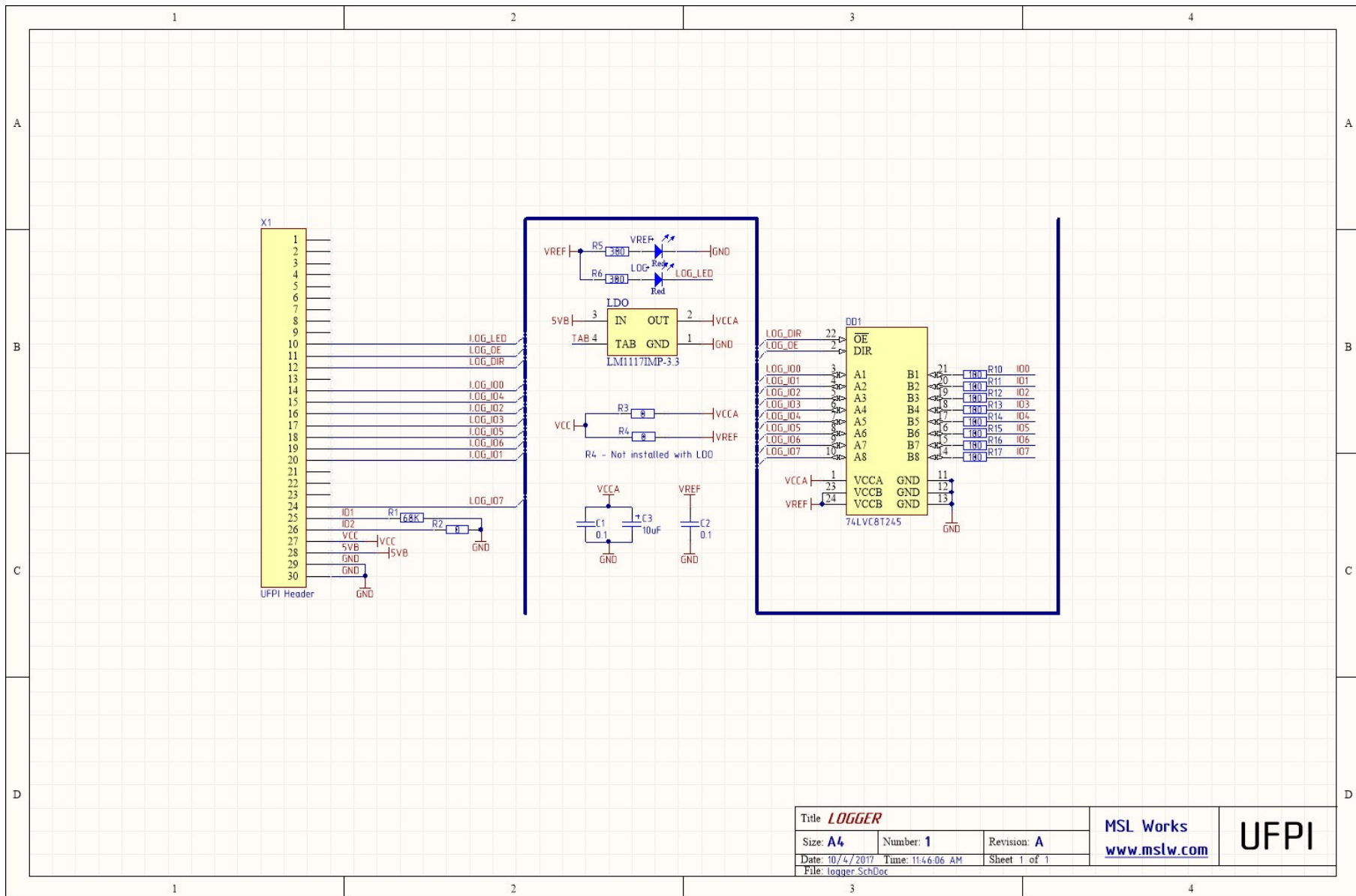
UFPI

16.3.6 SPI/I2C/mWire/1Wire DIP24 Socket schematics



Title <i>SPI/I2C/1W/mWire Socket</i>			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: B		
Date: 4/15/2020	Time: 107:41 PM	Sheet 1 of 1		
File: spi_i2c_mwire_1wSchDoc				

16.3.7 LOGGER Socket schematics

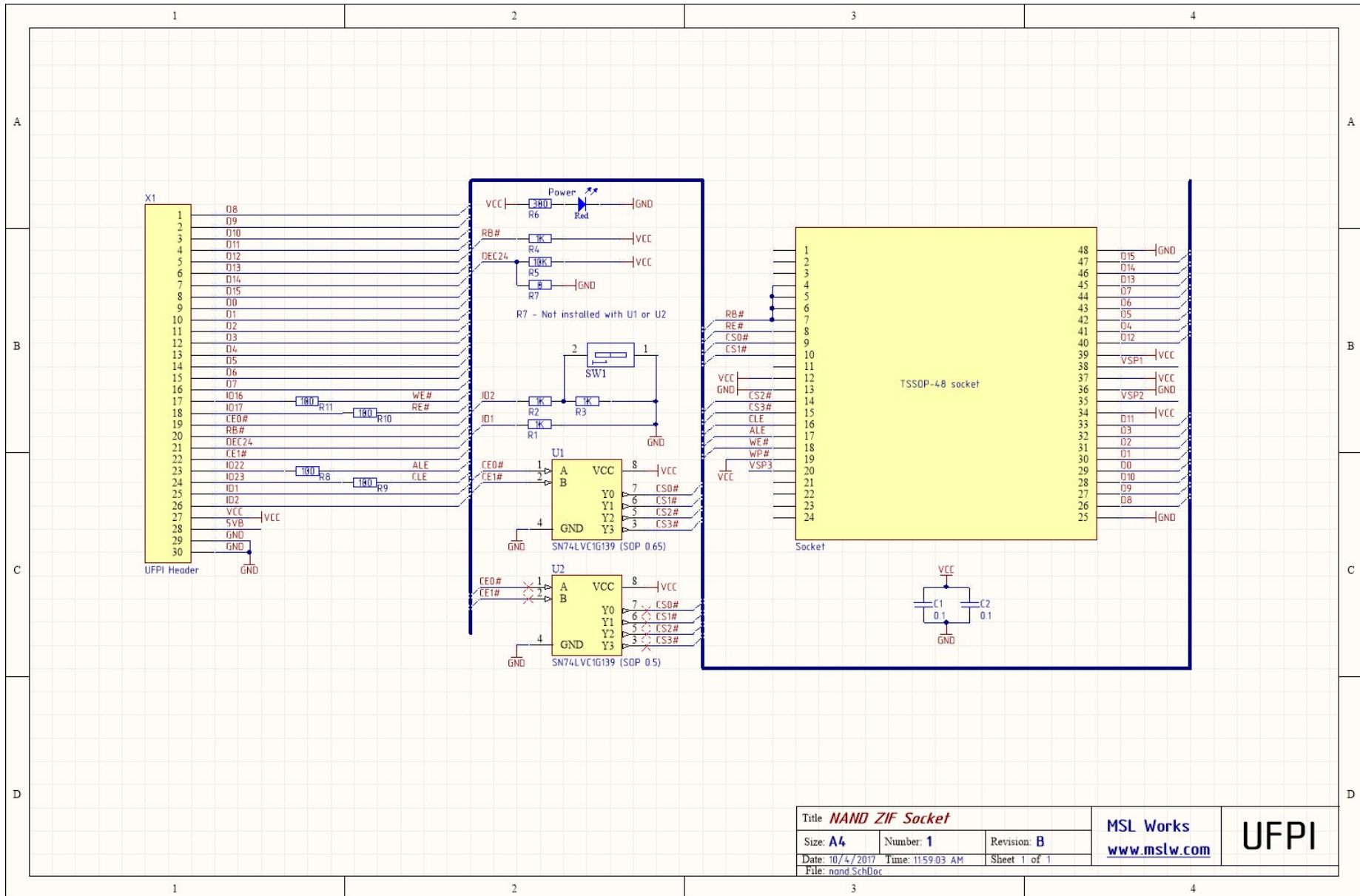


Title: LOGGER		
Size: A4	Number: 1	Revision: A
Date: 10/4/2017	Time: 11:46:06 AM	Sheet 1 of 1
File: logger_SchDoc		

MSL Works
www.mslw.com

UFPI

16.3.8 NAND Socket schematics

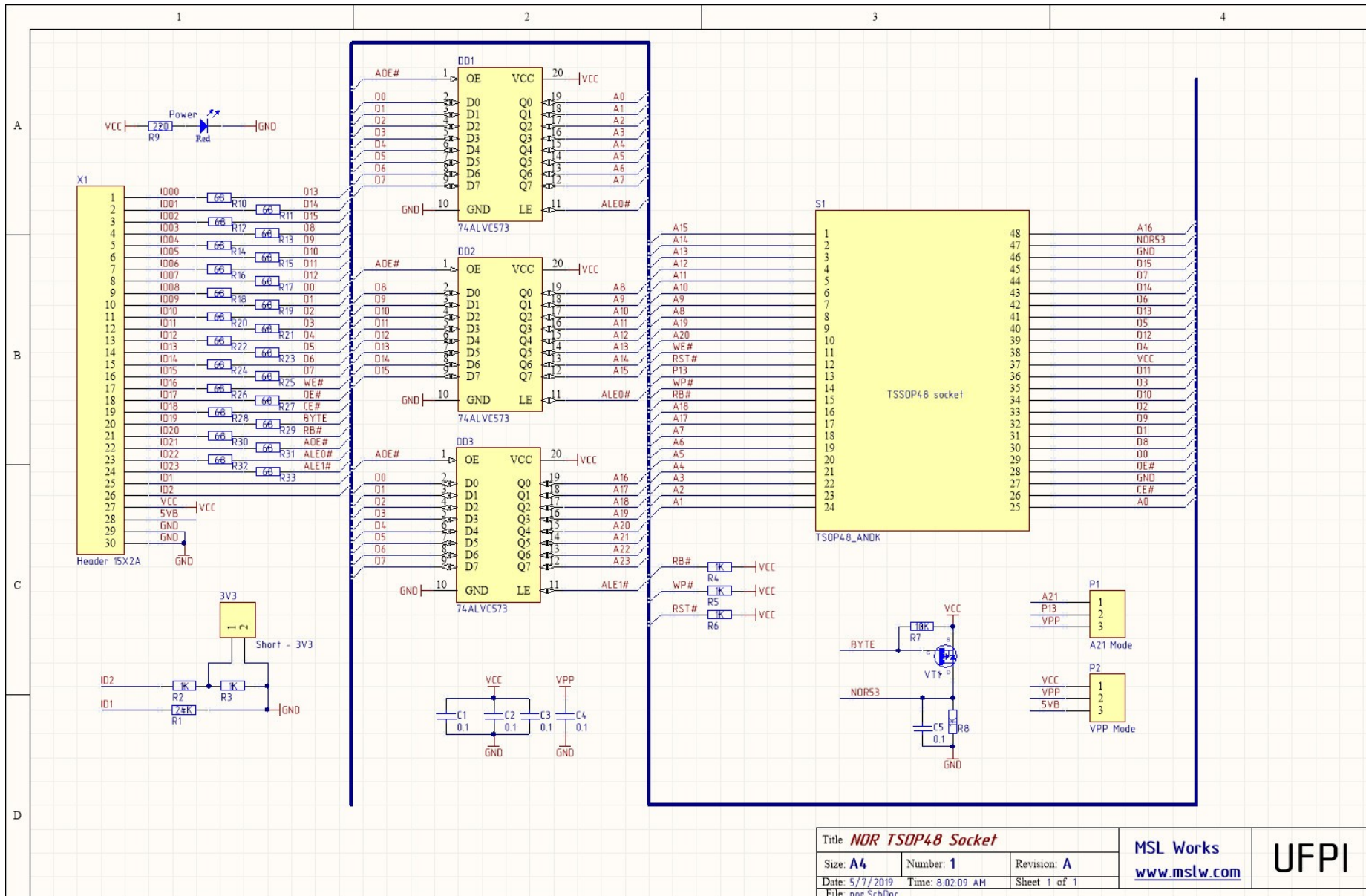


Title NAND ZIF Socket		
Size: A4	Number: 1	Revision: B
Date: 10/4/2017	Time: 11:59:03 AM	Sheet 1 of 1
File: nand_SchDoc		

MSL Works
www.mslw.com

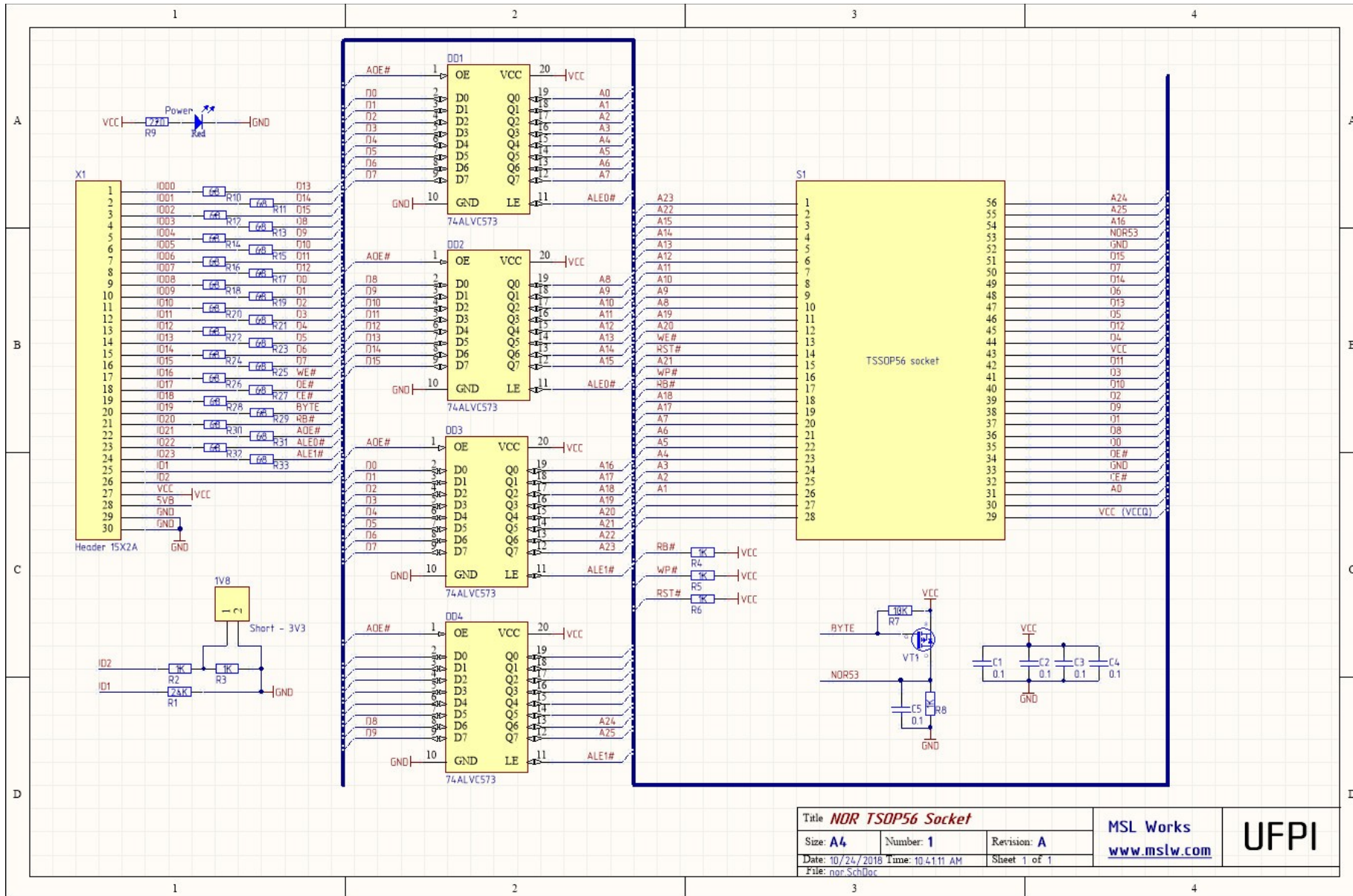
UFPI

16.3.9 NOR TSOP48 Socket schematics



Title NOR TSOP48 Socket			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: A		
Date: 5/7/2019	Time: 8:02:09 AM	Sheet 1 of 1		
File: nor_SchDoc				

16.3.10 NOR TSOP56 Socket schematics

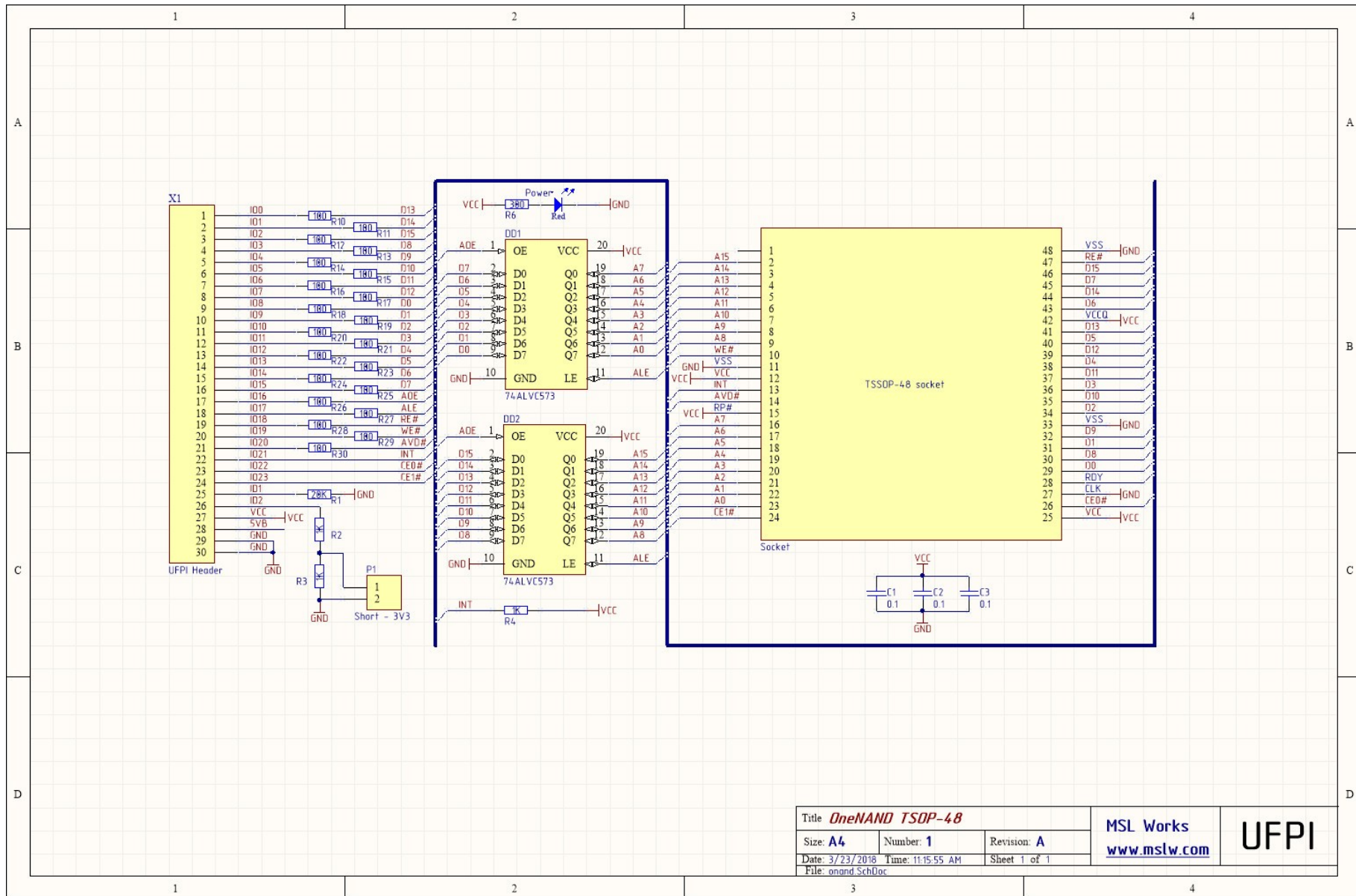


Title NOR TSOP56 Socket		
Size: A4	Number: 1	Revision: A
Date: 10/24/2018	Time: 10:41:11 AM	Sheet 1 of 1
File: nor_SchDoc		

MSL Works
www.mslw.com

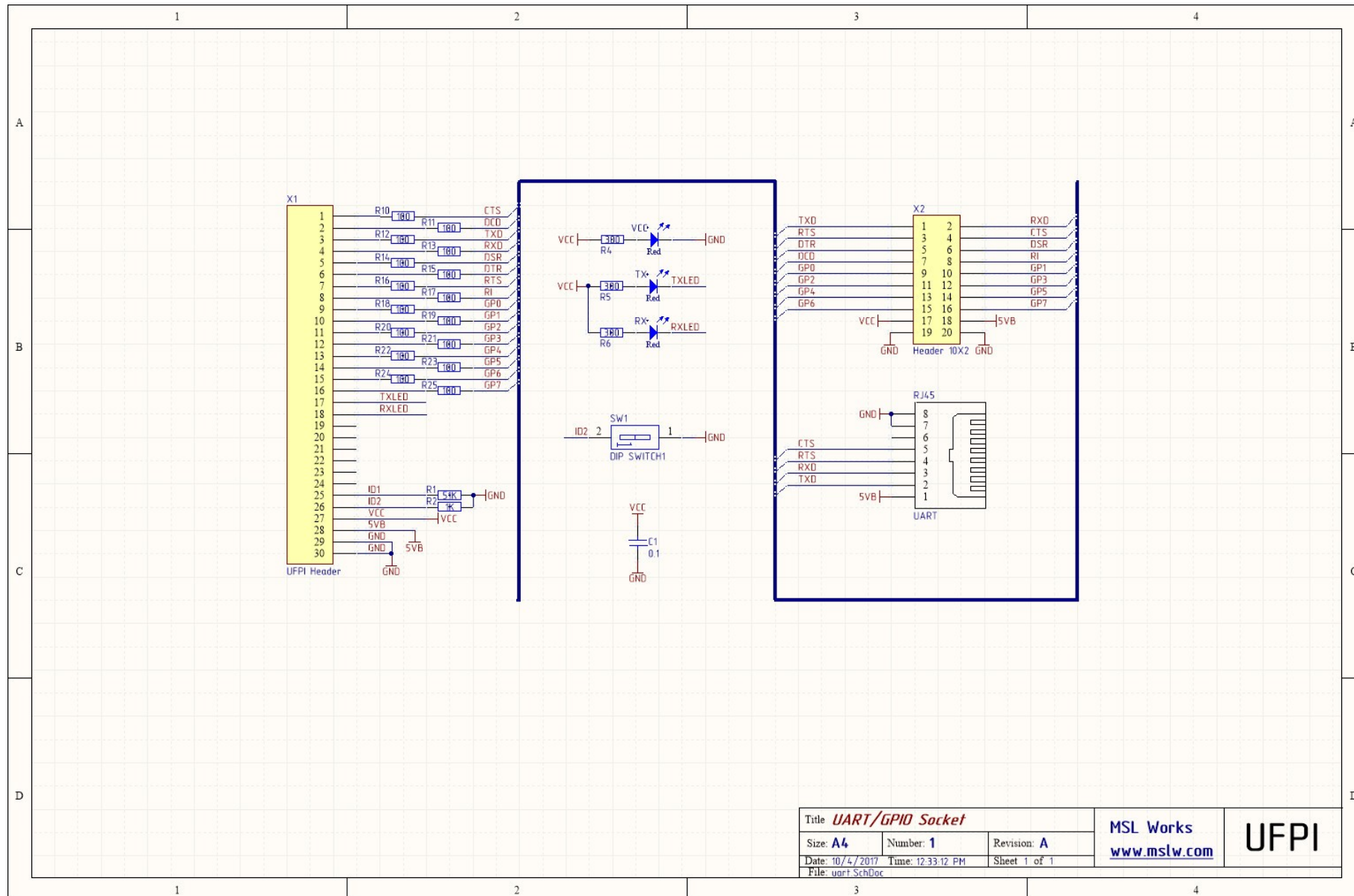
UFPI

16.3.11 OneNAND Socket schematics



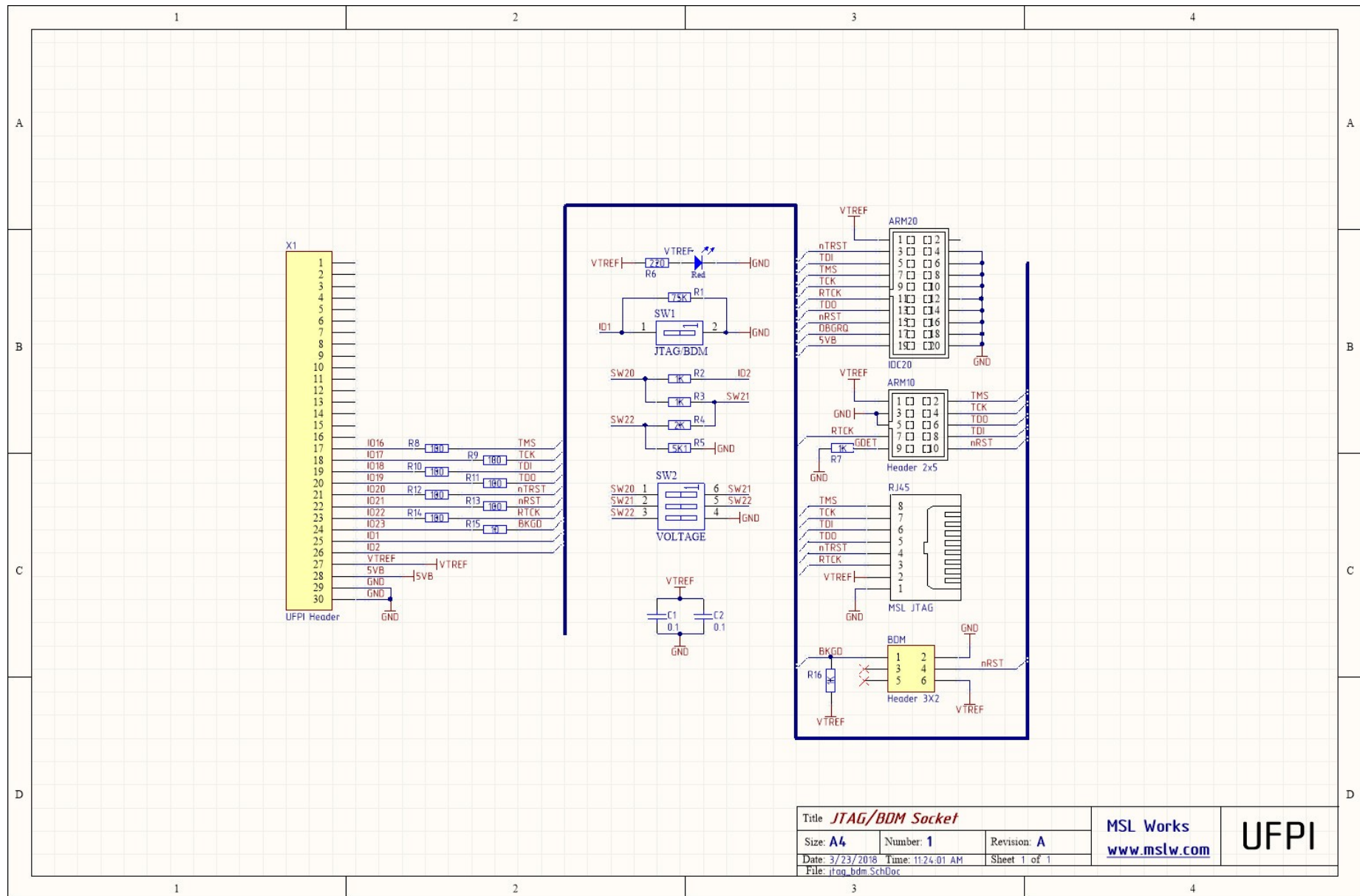
Title OneNAND TSSOP-48			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: A		
Date: 3/23/2018	Time: 11:15:55 AM	Sheet 1 of 1		
File: onand.SchDoc				

16.3.12 UART/GPIO Socket schematics



Title <i>UART/GPIO Socket</i>			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: A		
Date: 10/4/2017	Time: 12:33:12 PM	Sheet 1 of 1		
File: uart.SchDoc				

16.3.13 JTAG/BDM Socket schematics



Title <i>JTAG/BDM Socket</i>			MSL Works www.mslw.com	UFPI
Size: A4	Number: 1	Revision: A		
Date: 3/23/2018	Time: 11:24:01 AM	Sheet 1 of 1		
File: jtag_bdm SchDoc				