

ДИСТРИБУТИВ ОС GNU/LINUX НА БАЗЕ BUILDROOT ДЛЯ 1892VM14Я. РУКОВОДСТВО ПРОГРАММИСТА

**Версия
11.03.2022**

ОГЛАВЛЕНИЕ

1	О документе	3
2	Общие сведения	4
3	Подготовка ПЭВМ	7
3.1	Аппаратные требования к ПЭВМ	7
3.2	Программные требования к ПЭВМ	7
4	Состав дистрибутива ОС	9
4.1	Buildroot	9
4.2	Загрузчик U-Boot	10
4.3	Скрипты сборки в Docker-контейнере	10
5	Состав образа SD-карты	11
6	Сборка дистрибутива	12
6.1	Артефакты сборки	12
6.2	Сборка в ОС ПЭВМ	13
6.3	Сборка в контейнере Docker	13
7	Подготовка носителя ОС	15
7.1	Прошивка SD-карты	15
7.2	Прошивка eMMC	15
8	Прошивка SPI флеш-памяти	17
8.1	Прошивка SPI флеш-памяти через BootROM	17
8.2	Прошивка SPI флеш-памяти из ОС модуля	17
9	Настройка ОС	18
9.1	Увеличение объема ОЗУ	18
9.2	Включение драйвера framebuffer voutfb	18
9.3	Настройка сети	18
9.4	Настройка 6LoWPAN	18
9.5	Параметры запуска Linux	19
9.6	Добавление программ в образ SD-карты	19
9.7	Управление выводами GPIO из пространства пользователя	19
9.8	Разметка свободной области SD-карты и монтирование в /data	20
10	Запуск модуля	21
11	Сообщения системному программисту	22

1. О ДОКУМЕНТЕ

Документ описывает дистрибутив операционной системы GNU/Linux на базе Buildroot для микросхемы 1892ВМ14Я (МСom-02), процедуру сборки и прошивки образа SD-карты с операционной системой и настройки операционной системы (далее — «ОС»).

Документ описывает дистрибутив версии 3.1.

2. ОБЩИЕ СВЕДЕНИЯ

Дистрибутив ОС GNU/Linux для MCom-02 предназначен для распространения исходных кодов ОС, инструментального ПО и скриптов сборки ОС.

Особенности дистрибутива:

- Поддержка модулей на базе микросхемы 1892ВМ14Я:
 - Салют-ЭЛ24Д1 r1.3;
 - Салют-ЭЛ24Д1 r1.4;
 - Салют-ЭЛ24Д1 r1.5;
 - Салют-ЭЛ24Д1 r1.5 с установленным навигационным радиомодулем RF2Chan v2;
 - Салют-ЭЛ24Д2 r1.1;
 - Салют-ЭЛ24ОМ1 r1.1 с установленным Салют-ЭЛ24ПМ1 r1.1 или Салют-ЭЛ24ПМ1 r1.2;
 - Салют-ЭЛ24ОМ1 r1.2 с установленным Салют-ЭЛ24ПМ1 r1.2, Салют-ЭЛ24ПМ2 r1.0 или Салют-ЭЛ24ПМ2 r1.1.
- Основные компоненты (подробнее см. *Состав дистрибутива ОС*):
 - система сборки Buildroot,
 - загрузчик U-Boot,
 - ядро Linux,
 - внешнее дерево Buildroot external-mcom02,
- Компоненты дистрибутива основаны на соответствующих версиях из публичных репозиториях с <https://github.com/elvees>.
- Архив дистрибутива содержит все компоненты — при сборке дистрибутива с поставляемыми файлами конфигурации поддержки MCom-02 не требуется доступ в интернет.

Компоненты дистрибутива предоставляются в исходных кодах. Исключения:

- библиотека пространства пользователя драйвера Mali GPU, демо-приложения GPU;
 - инструментальное ПО DSP DELcore-30M;
 - прошивки микросхем контроллеров установленных на модулях (например, Wi-Fi).
- Результатом сборки дистрибутива являются (подробнее см. *Сборка дистрибутива*):

- Образ SD-карты, содержащий ОС GNU/Linux. Образ SD-карты унифицирован и совместим со всеми поддерживаемыми модулями.
- Образы загрузчика U-Boot для всех поддерживаемых модулей.
- Поддержка сборки дистрибутива в CentOS 7 и в Docker-контейнере.
- Дистрибутив содержит следующие файлы конфигурации поддержки MCom-02:
 - `mcom_defconfig` — конфигурация дистрибутива по умолчанию.
 - `mcommin_defconfig` — минимальная конфигурация для загрузки ОС.
- Особенности файла конфигурации `mcom_defconfig`:
 - Назначение конфигурации — демонстрация и тестирование интерфейсов MCom-02 и поддерживаемых модулей.
 - Имя пользователя, пароль — `root`, `root`.
 - Корневая файловая система основана на BusyBox, оболочка — Bash, система инициализации — systemd.
 - Middleware: Python 3, GStreamer, ALSA, FFmpeg, Bluez, libdrm.
 - Стандартные бенчмарки и тесты: `coremark`, `ramspeed`, `memtester`, `fiio`, `dhystone`, `bonnie`, `hdparm`, `i2c-tools`, `iperf` и т.п.
 - Пакеты поддержки Mali GPU: модуль пространства ядра, библиотеки пространства пользователя, `mcom-mali-demos`. Аппаратное ускорение Qt на GPU.
 - Пакет поддержки SpaceWire — `mcom-libswic`.
 - Поддержка DSP DELcore-30M:
 - * Пакет утилит и демонстрационных приложений — `mcom-delcore30m-tests`.
 - * Инструментальное ПО для DSP (компилятор, линковщик).
 - Пакеты реализации OpenVX и демонстраций OpenVX: `openvx`, `openvx-demo`.
 - Пакеты реализации навигационного решения GNSS:
 - * `rf2chan-v2-gnss` — модуля ядра для поддержки радиомодуля RF2Chan v2.
 - * `mcom-talita` — драйвер пространства пользователя с реализацией навигационного решения.
 - Поддержка U-Boot из целевой ОС:
 - * `fw_setenv`, `fw_printenv` — утилиты установки и чтения переменных окружения U-Boot.
 - * `mcom02-fw-update` — утилита прошивки образов U-Boot.
 - Поддержка файла идентификации ОС `/etc/os-release`¹.

¹ <https://www.freedesktop.org/software/systemd/man/os-release.html>

- udev-правила и systemd-сервисы для инициализации 6LoWPAN, ALSA, CAN, SPI, Wi-Fi.
- Сетевой адрес получается по DHCP. Имя хоста по умолчанию — `msom02`.
- По умолчанию включен SSH.

3. ПОДГОТОВКА ПЭВМ

3.1 Аппаратные требования к ПЭВМ

ПЭВМ должна иметь конфигурацию:

- не менее 4 ГиБ ОЗУ, 20 ГиБ свободного места на НЖМД или твердотельном накопителе;
- на ПЭВМ должен быть установлен кард-ридер SD-карт;

Для работы с модулем требуется дополнительное оборудование:

- microUSB-USB кабель;
- патч-корд Cat5e;
- MicroSD-карта;
- опционально: USB-флеш-накопитель.

3.2 Программные требования к ПЭВМ

1. Операционная система ПЭВМ — CentOS 7.8 x86-64.
2. Для прошивки образов и работы с UART пользователь должен быть добавлен в группы `disk`, `dialout`.
3. Для сборки дистрибутива в ОС ПЭВМ должны быть установлены RPM-пакеты, перечисленные в файле `Dockerfile` (см. *Скрипты сборки в Docker-контейнере*):

```
sudo yum install <packages-from-dockerfile> -y
```

Примечание: Для установки пакетов на ПЭВМ должен быть настроен доступ в интернет.

Внимание: Начиная с версии Buildroot v3.2 требуется наличие GCC компилятора $\geq v5$.

Для GCC $\geq v5$ необходимо установить репозиторий `centos-release-scl-rh`, RPM-пакет `devtoolset-9` и настроить переменную окружения `PATH`. Подробнее см. соответствующие команды в файле `Dockerfile`.

Для работы с терминалом UART должны быть установлены RPM-пакеты `minicom` и `putty`.

4. В зависимости от модели используемого USB-UART переходника необходима установка драйверов.
5. При сборке дистрибутива в Docker-контейнере требуется дополнительная настройка ПЭВМ, подробнее см. *Сборка в контейнере Docker*.

4. СОСТАВ ДИСТРИБУТИВА ОС

Дерево исходных кодов дистрибутива представлено на диаграмме:

```
├─ buildroot/
│  └─ ...
│     └─ dl/
│        └─ output/
│           └─ ...
├─ Dockerfile
├─ external-common/
├─ external-mcom02/
├─ Makefile
├─ Makefile.docker
├─ Makefile.u-boot
└─ u-boot/
```

Компоненты исходного кода представлены в главах:

- *Buildroot*,
- *загрузчик U-Boot*,
- *скрипты сборки в Docker-контейнере*,

4.1 Buildroot

Пакет состоит из компонентов:

buildroot Исходные коды [системы сборки Buildroot](https://buildroot.org/)². Базовая версия Buildroot — 2018.02³. Некоторые рецепты пакетов Buildroot изменены.

Buildroot сконфигурирован файлом конфигурации `external-mcom02/configs/mcom_defconfig`. В директории `buildroot/dl` содержатся архивы исходных кодов всех пакетов данной конфигурации.

Архив `buildroot/dl/linux-mcom02*.tar.gz` — исходные коды ядра Linux 4.4.189.2⁴.

external-common Внешнее дерево пакетов Buildroot независимых от архитектуры. Дерево оформлено в соответствии с описанием [Buildroot br2-external tree](#)⁵.

external-mcom02 Внешнее дерево пакетов Buildroot для поддержки MCom-02.

Makefile Скрипт сборки Buildroot. Скрипт устанавливает переменную `BR2_EXTERNAL` с указанием до директорий `external-*` и вызывает `make` в директории `buildroot`. Т.о.

² <https://buildroot.org/>

³ <https://git.buildroot.net/buildroot/commit/?h=2018.02>

⁴ <https://github.com/elvees/linux/tree/mcom02/v4.4.189.2>

⁵ <https://buildroot.org/downloads/manual/manual.html#outside-br-custom>

при вызове `make` в корневой директории дистрибутива доступны все стандартные цели Buildroot⁶, например: `make help` — вывод справки по целям Buildroot.

Результатом сборки Buildroot являются образ SD-карты и инструментальные средства Buildroot SDK для ПЭВМ (подробнее см. *Артефакты сборки*).

4.2 Загрузчик U-Boot

Загрузчик состоит из компонентов:

u-boot Исходные коды загрузчика U-Boot версии 2019.01.0.9. Описание загрузчика см. документ «Загрузчик U-Boot для 1892BM14Я. Руководство программиста».

Makefile.u-boot Скрипт сборки образов U-Boot всех поддерживаемых модулей. Для сборки U-Boot скрипт использует инструментальные средства Buildroot SDK.

Результатом сборки U-Boot являются образы перечисленные в главе *Артефакты сборки*.

4.3 Скрипты сборки в Docker-контейнере

Состав скриптов:

Dockerfile Файл конфигурации Docker-образа⁷ для среды сборки дистрибутива.

Makefile.docker Скрипт сборки Docker-образа и сборки дистрибутива в контейнере Docker.

⁶ <https://buildroot.org/downloads/manual/manual.html#make-tips>

⁷ <https://docs.docker.com/>

5. СОСТАВ ОБРАЗА SD-КАРТЫ

Схема разбиения образа SD-карты представлена в таблице 5.

Таблица 5.1. Схема разбиения образа SD-карты на области

Область	Начало (байт)	Размер (байт)	Примечание
MBR	0	512	
Раздел <i>root</i>	1 МиБ	1 ГиБ	Раздел с файловой системой EXT4 с корневой файловой системой <i>rootfs</i>

На разделе *root* содержится скомпилированное ядро Linux — файл `/boot/zImage`.

6. СБОРКА ДИСТРИБУТИВА

Перед сборкой дистрибутива необходимо:

1. Разархивировать архив дистрибутива (<package-name> — имя архива без расширения tar.bz2)

```
tar xf <package-name>.tar.bz2
```

Предупреждение: Требования к полному пути к архиву и имени архива:

- должны использовать только ASCII-символы (не должны содержать символов кириллицы),
- не должны содержать пробелов.

2. Сменить текущую рабочую директорию в распакованную директорию:

```
cd <package-name>
```

Сборка дистрибутива выполняется в *ОС ПЭВМ* или в *контейнере Docker*.

6.1 Артефакты сборки

Результатом сборки дистрибутива являются:

1. Образ SD-карты `buildroot/output/images/mcom02-buildroot-sdcard.img`, состав образа описан в главе *Состав образа SD-карты*.
2. Инструментальные средства Buildroot SDK для ПЭВМ — в директории `buildroot/output/host`.
3. Образы SPI флеш-памяти `uboot-images/*.img`. Соответствие модулей и образов загрузчика:
 - `mcom02-salute-el24d1-r1.3-uboot.img` — Салют-ЭЛ24Д1 r1.3;
 - `mcom02-salute-el24d1-r1.4-uboot.img` — Салют-ЭЛ24Д1 r1.4;
 - `mcom02-salute-el24d1-r1.5-uboot.img` — Салют-ЭЛ24Д1 r1.5;
 - `mcom02-salute-el24d1-r1.5-rf2chan-uboot.img` — Салют-ЭЛ24Д1 r1.5 с навигационным радиомодулем RF2Chan v2;
 - `mcom02-salute-el24d2-r1.1-uboot.img` — Салют-ЭЛ24Д2 r1.1;
 - `mcom02-salute-el24pm1-r1.1-1.2-om1-r1.1-1.2-uboot.img` — Салют-ЭЛ24ОМ1 с Салют-ЭЛ24ПМ1;

- `mcom02-salute-el24pm2-r1.2-om1-r1.1-1.2-uboot.img` — Салют-ЭЛ24ОМ1 с Салют-ЭЛ24ПМ2.

Совместимость ревизий модулей ОМ и ПМ приведена в [Таблице совместимости](#)⁸.

6.2 Сборка в ОС ПЭВМ

1. Выполнить команду для сборки Buildroot и образа SD-карты:

```
make
```

Длительность сборки составляет около 150 минут и зависит от производительности CPU ПЭВМ.

Результатом сборки являются образ SD-карты и Buildroot SDK.

2. Собрать образ U-Boot:

1. Выполнить команду для вывода справки по целям скрипта, запомнить цель соответствующую используемому модулю:

```
make -f Makefile.u-boot help
```

2. Выполнить команду для сборки U-Boot. Пример команды сборки загрузчика модуля Салют-ЭЛ24Д1 r1.5:

```
make -f Makefile.u-boot saluted1-r15
```

Результатом сборки данного примера является файл `uboot-images/mcom02-salute-el24d1-r1.5-uboot.img`.

6.3 Сборка в контейнере Docker

Сборка в контейнере Docker предназначена для организации воспроизводимой сборки (например, непрерывной интеграции) и не рекомендуется для разработки и отладки скриптов сборки.

Предупреждение: В случае если сборка Buildroot была выполнена в контейнере, то повторные запуски сборки (после реконфигурации, изменения исходных кодов пакетов) также должны выполняться в контейнере. Повторный запуск сборки Buildroot в ОС ПЭВМ будет завершаться ошибкой. Для очистки генерируемых промежуточных файлов Buildroot необходимо выполнить `make clean`.

Причины:

- Рабочая директория Buildroot на файловой системе ОС ПЭВМ и на файловой системе контейнера имеет различные пути.

⁸ <http://multicore.ru/index.php?id=1389>

- При сборке скрипты Buildroot устанавливают абсолютные пути в генерируемых скриптах сборки.

Для сборки в контейнере Docker необходимо:

1. Установить и настроить сервис Docker:

1. Установить Docker версии 17.07 или выше на ПЭВМ согласно инструкции [Get Docker CE for CentOS⁹](#).
2. Добавить текущего пользователя в группу `docker` согласно инструкции [Post-installation steps for Linux¹⁰](#).
3. Настроить прокси для сервиса Docker (при необходимости) согласно инструкции [Control Docker with systemd¹¹](#).
4. Настроить прокси для клиента Docker (при необходимости) согласно инструкции [Configure the Docker client¹²](#).

2. Запустить сборку образа Docker:

```
make -f Makefile.docker checkenv image
```

Проверить, что образ создан командой `docker images | grep buildroot`. Пример вывода:

```
elvees mcom02-buildroot-centos-v1.0 ad261c2c728c 26 hours ago 533MB
```

3. Запустить сборку Buildroot и образа SD-карты в контейнере:

```
make -f Makefile.docker build CONTAINER_CMD="make"
```

4. Запустить сборку образов U-Boot всех поддерживаемых модулей в контейнере:

```
make -f Makefile.docker build CONTAINER_CMD="make -f Makefile.u-boot"
```

Результатом сборки являются образы, перечисленные в главе *Артефакты сборки*.

⁹ <https://docs.docker.com/install/linux/docker-ce/centos/>

¹⁰ <https://docs.docker.com/install/linux/linux-postinstall/>

¹¹ <https://docs.docker.com/config/daemon/systemd/>

¹² <https://docs.docker.com/network/proxy/#configure-the-docker-client>

7. ПОДГОТОВКА НОСИТЕЛЯ ОС

7.1 Прошивка SD-карты

Для записи образа на SD-карту необходимо:

1. Извлечь SD-карту из считывателя карт ПЭВМ и считать список устройств командой:

```
ls -la /dev/sd*
```

2. Установить SD-карту в кард-ридер ПЭВМ и повторно считать список устройств командой `ls -la /dev/sd*`. Вычесть из списка устройств после установки SD-карты список устройств до установки карты и получить устройство `/dev/sdX` и/или список устройств `/dev/sdX1`, `/dev/sdX2`... (где 1, 2, ... номера разделов SD-карты). В случае, если получен список устройств, то получить устройство `/dev/sdX` отбрасыванием последней цифры из устройства соответствующего первому разделу SD-карты `/dev/sdX1`.
3. Записать образ на SD-карту:

```
sudo dd if=buildroot/output/images/mcom02-buildroot-sdcard.img of=/dev/sdX  
↵ bs=4M  
sudo sync
```

4. Извлечь SD-карту из считывателя карт ПЭВМ.

7.2 Прошивка eMMC

Для прошивки образа в память eMMC на модулях Салют-ЭЛ24ПМ1 и Салют-ЭЛ24ПМ2 выполнить:

1. *Запустить модуль.* Загрузка системы должна быть произведена с SD-карты.
2. Подключиться к модулю по терминалу UART и считать его IP-адрес командой `ifconfig`, значение IP-адреса сохранить на ПЭВМ в переменной `TARGET_IP_ADDR`:

```
TARGET_IP_ADDR=<IP-адрес-модуля>
```

3. В терминале на ПЭВМ установить переменную с путём до файла образа:

```
IMG_PATH=<путь-до-файла-образа-SD-карты>
```

4. Выполнить на ПЭВМ команду:

```
ssh root@$TARGET_IP_ADDR dd of=/dev/mmcblk0 bs=4M conv=fsync < $IMG_PATH
```

5. При необходимости проверить корректность записи образа, для этого на ПЭВМ выполнить команду:

```
IMG_SIZE=$(stat -c%s "$IMG_PATH")  
  
ssh root@$TARGET_IP_ADDR "dd if=/dev/mmcblk0 bs=4M count=$((($IMG_SIZE / 1024  
↪ / 1024 / 4)) | head -c $IMG_SIZE | md5sum"
```

Полученную MD5-сумму сравнить с MD5-суммой записанного файла образа.

8. ПРОШИВКА SPI ФЛЕШ-ПАМЯТИ

8.1 Прошивка SPI флеш-памяти через BootROM

Прошивка SPI флеш-памяти модуля *образом загрузчика U-Boot* через BootROM выполняется в случае, если загрузчик повреждён или не прошит. Операция трудозатратна: требуются ручные циклы переключения питания и изменение состояния переключателей *BOOT* на модуле.

Прошивка выполняется утилитами из пакета MCom-02 flash tools согласно [инструкции](#)¹³ на GitHub.

8.2 Прошивка SPI флеш-памяти из ОС модуля

Для прошивки SPI флеш-памяти из ОС модуля используется утилита `mcom02-fw-update`: необходимо скопировать образ с ПЭВМ на ФС устройства (например, командой `scp`) и выполнить на устройстве:

```
mcom02-fw-update <image-file>
```

Предупреждение: Во время исполнения процесс прошивки не должен прерываться.

Для прошивки с восстановлением переменных окружения U-Boot используется опция `-r`:

```
mcom02-fw-update -r <image-file>
```

Подсказка: Для печати и установки переменных окружения загрузчика U-Boot в ОС модуля используются утилиты `fw_printenv` и `fw_setenv` соответственно.

¹³ <https://github.com/elvees/mcom02-flash-tools>

9. НАСТРОЙКА ОС

9.1 Увеличение объема ОЗУ

Доступный объем ОЗУ по умолчанию равен 1 ГиБ для модулей Салют-ЭЛ24Д1 и Салют-ЭЛ24Д2, 2 ГиБ — для модулей Салют-ЭЛ24ОМ1. Для увеличения объема ОЗУ до 2 ГиБ необходимо активировать второй контроллер DDR (если применимо для данного модуля).

Включение контроллера DDR выполняется установкой значения `enable` для переменной окружения `ddrctl_cmd` в режиме монитора загрузчика (подробнее см. «Загрузчик U-Boot для 1892ВМ14Я. Руководство программиста»):

```
setenv ddrctl_cmd enable
saveenv
```

9.2 Включение драйвера framebuffer voutfb

По умолчанию образ SD-карты собирается с выключенным модулем ядра `voutfb` для модулей Салют-ЭЛ24Д1 и Салют-ЭЛ24Д2. При необходимости включение произвести путем удаления строки `modprobe.blacklist=voutfb` из переменной окружения `cmdline` в режиме монитора загрузчика (подробнее см. «Загрузчик U-Boot для 1892ВМ14Я. Руководство программиста»)

9.3 Настройка сети

По умолчанию ОС настроена на получение сетевого адреса по DHCP. Настройка параметров сети задаётся в конфигурационных файлах `/etc/systemd/network/*.network` и `/usr/lib/systemd/network/*.network` на корневой файловой системе. Полная документация по настройке сети доступна в [документации systemd](#)¹⁴.

Имя хоста по умолчанию — `msom02`. Для изменения имени хоста необходимо отредактировать конфигурационные файлы `/etc/hostname` и `/etc/hosts` на корневой файловой системе.

9.4 Настройка 6LoWPAN

Сетевой интерфейс 6LoWPAN обеспечивает взаимодействие по протоколу IPv6 через сеть стандарта IEEE 802.15.4.

Параметры сетевого интерфейса задают командой `ip`. Для автоматической настройки интерфейса используется сервис `systemd lowpan.service` (выключен по умолчанию).

Для изменения IP, выставяемого сервисом, необходимо изменить файл:

¹⁴ <https://www.freedesktop.org/software/systemd/man/systemd.network>

- `external-mcom02/overlay/elvees/usr/lib/systemd/system/lowpan.service` в директории исходных кодов на ПЭВМ;
- или `/usr/lib/systemd/system/lowpan.service` на файловой системе на целевой платформе и перезапустить сервис:

```
systemctl daemon-reload
systemctl restart lowpan
```

9.5 Параметры запуска Linux

Параметры запуска Linux задаются с помощью переменных окружения загрузчика (подробнее см. «Загрузчик U-Boot для 1892BM14Я. Руководство программиста»).

Совет: При работе с Салют-ЭЛ24ОМ1 для изменения загрузочного устройства на карту памяти microSD необходимо установить переменную окружения загрузчика `boot_targets` в значение `legacy_mmc1 mmc1`.

Дополнительные параметры запуска Linux необходимо передавать через переменную окружения `cmdline` загрузчика.

Параметры запуска Linux описаны в [Kernel Parameters](#)¹⁵.

9.6 Добавление программ в образ SD-карты

Система сборки Buildroot поддерживает добавление в сборку программ и библиотек пользователя. Подробная документация находится в директории `buildroot/docs`.

Возможно добавление программ на базе стандарта компьютерного зрения OpenVX (см. «ELVEES OpenVX SDK для 1892BM14Я. Руководство программиста»).

9.7 Управление выводами GPIO из пространства пользователя

Управление выводами GPIO СнК осуществляется с помощью служебных файлов в `/sys/class/gpio` (подробнее см. [GPIO Sysfs Interface for Userspace](#)¹⁶).

Управление выводом GPIO может быть недоступно, если он используется драйвером.

В соответствии с руководством пользователя на СнК выводы GPIO делятся на 4 группы: GPIOA, GPIOB, GPIOC, GPIOD. В Linux выводы GPIO обозначаются номерами. Соответствие базовых номеров для каждой группы приведено в таблице 9.7. Внутри групп номера идут по порядку. Например, вывод GPIOA5 соответствует номеру $480 + 5 = 485$ в Linux.

¹⁵ <https://github.com/elvees/linux/blob/mcom02/Documentation/kernel-parameters.txt>

¹⁶ <https://www.kernel.org/doc/Documentation/gpio/sysfs.txt>

Таблица 9.1. Соответствие обозначений выводов GPIO номерам в Linux

Группа	Базовый номер в Linux
GPIOA	480
GPIOB	448
GPIOC	416
GPIOD	384

9.8 Разметка свободной области SD-карты и монтирование в /data

Для создания раздела из свободной области SD-карты и монтирования в директорию /data необходимо:

1. Запустить модуль.
2. Выполнить команду:

```
create-data-partition && echo "Partition successfully created"
```

Разметку считать завершённой успешно в случае вывода в терминал скриптом сообщения:

```
Partition successfully created
```

10. ЗАПУСК МОДУЛЯ

Для запуска модуля необходимо выполнить следующие действия:

1. Собрать образ SD-карты и образ загрузчика для модуля.
2. Записать образ SD-карты.
3. Прошить SPI флеш-память модуля образом загрузчика.
4. Настроить ОС.
5. Установить SD-карту в слот MicroSD модуля.
6. Установить переключатель *BOOT* модуля в положение, соответствующее загрузке из SPI флеш-памяти (подробнее см. Руководство пользователя на модуль).
7. Подключить модуль к источнику питания (подробнее см. Руководство пользователя на модуль).
8. Открыть терминал UART модуля, или установить соединение по протоколу SSH (логин: `goot`, пароль: `goot`).
9. Выполнить команду `uname -a`. Считать модуль готовым к использованию при выводе в терминал сообщения:

```
Linux mcom02 4.1.41.3 #1 SMP Fri Sep 1 17:08:44 MSK 2017 armv7l GNU/Linux
```

Примечание: `systemd` инициализирует пользовательское пространство параллельно процедуре инициализации устройств (к примеру, терминал инициализируется до загрузки всех модулей ядра). Это может привести к проблемам запуска, на ранних этапах, приложений, взаимодействующих с устройствами. Рекомендации по ожиданию устройств описаны в `systemd-udev-settle.service`¹⁷.

¹⁷ <https://www.freedesktop.org/software/systemd/man/systemd-udev-settle.service.html>

11. СООБЩЕНИЯ СИСТЕМНОМУ ПРОГРАММИСТУ

Отладочный модуль выводит в терминал UART сообщения о ходе загрузки. Пример вывода в терминал при успешной загрузке:

```
DDR retention disabled

U-Boot SPL 2017.07.0.3 (Sep 01 2017 - 17:12:01)
DDR controllers init started
DDR controller #0 init done
DDR controller #1 init done
Trying to boot from SPI

U-Boot 2017.07.0.3 (Sep 01 2017 - 17:12:01 +0300), Build: v2.5-2017-09-01

CPU:   MCom-compatible
Model: Salute-EL240M1 r1.1
I2C:   ready
DRAM:  2 GiB
MMC:   sdhci0@3800b000: 0, sdhci1@3800d000: 1
SF: Detected m25p32 with page size 256 Bytes, erase size 64 KiB, total 4 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
DDR controller #1 disabled
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc0(part 0) is current device
reading zImage
3310424 bytes read in 211 ms (15 MiB/s)
## Flattened Device Tree blob at 7f768140
   Booting using the fdt blob at 0x7f768140
   Loading Device Tree to 4fff8000, end 4ffffbcd ... OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Linux version 4.1.41.3 (jenkins_drap@leo-pc.elvees.com)
(gcc version 5.2.0 (Buildroot 2015.08.1) ) #1 SMP Fri Sep 1 17:08:44 MSK 2017
...

Welcome!
mcom02 login:
```

АЛФАВИТНЫЙ УКАЗАТЕЛЬ

В

boot_targets, 19

С

cmdline, 18, 19

Д

ddrctl_cmd, 18



переменная окружения

boot_targets, 19

cmdline, 18, 19

ddrctl_cmd, 18