

Linux-драйвер forward

1.7.9

2026-01-27

Введение

Поддержка плат FDxxx в Linux осуществляется с помощью open-source драйвера **forward**.

Драйвер поддерживает следующие типы плат:

- FD722 - 2 входа, 2 выхода 3G-SDI
- FD722M2 - 1 входа, 1 выход, 1 настраиваемый вход/выход 3G-SDI
- FD722BP - 2 входа, 2 выхода 3G-SDI с аппаратным пропусканием сигнала
- FD788 - 8 настраиваемых входов/выходов 3G-SDI
- FD922 - 2 входа, 2 выхода 12G-SDI
- FD722 - 2 входа HDMI 2.0
- FD940 - 4 входа HDMI 2.0
- FD2110 - 2 настраиваемых входов/выходов 12G-SDI, 2 интерфейса 25G-Ethernet

Драйвер состоит из следующих компонент:

- Модуль ядра `forward` - обеспечивает базовый функционал плат - настройку входов-выходов, прошивку и т.д.
- Модуль ядра `forward-v4l2` - обеспечивает передачу видео через API Video4Linux2 - может использоваться как ПО компании SoftLab-NSK, так и сторонним ПО (ffmpeg, gstreamer, OBS и т.д.).
- Модуль ядра `forward-alsa` - обеспечивает передачу аудио через API ALSA - может использоваться сторонним ПО в качестве звуковой карты (pulseaudio, pipewire, gstreamer, OBS и т.д.).
- Модуль ядра `forward-ethernet` - предоставляет системный ethernet-адаптер для платы FD2110
- Приложение `forward-flash` - для обновления прошивки (firmware) устройств

Модули ядра собираются из исходников при установке с помощью системы DKMS. Эта система позволяет автоматически собирать драйвер под все ядра Linux, которые установлены в системе, а также автоматически пересобирает драйвер при обновлении ядра.

Минимальные требования к операционной системе

- Архитектура amd64 (x86_64):
- linux 5.4
- glibc 2.28
- dkms 2.8

Установка драйвера

Драйвер поставляется в трёх вариантах:

- Пакеты deb для систем на основе debian (debian, ubuntu, mint, Астра и т.д.)
- Пакеты rpm (CentOS, RHEL, Альт, РедОС и т.д.)
- Архив tar.bz2 для ручной установки ("тарбол") - для всех остальных систем Linux

Установка из deb (Ubuntu, debian, и т.д.)

Для систем с apt установка производится следующей командой

```
sudo apt install -y ./forward-driver-<version>_all.deb ./forward-
utils-<version>_amd64.deb
```

Для обновления достаточно установить новую версию пакета - старый драйвер удалится автоматически

Установка из архива tar

Для установки из архива драйвер требуется распаковать в корневую директорию, а затем добавить драйвер в систему dkms

```
sudo tar -C / -xhvf forward-driver-<version>.tar.bz2
sudo tar -C / -xhvf forward-utils-<version>.tar.bz2
sudo dkms install forward/<version>
```

Удаления драйвера в таком варианте не предусмотрено, однако это можно сделать вручную, убрав драйвер из dkms и удалив директорию:

```
sudo dkms remove forward/<version>
sudo rm -r /usr/src/forward-<version>
sudo rm -r /lib/firmware/forward
```

Проверка установки

После установки драйвера любым способом в системе должна появиться директория `/usr/src/forward-<version>` с исходниками драйвера. Также должна корректно отрабатываться команда `modprobe forward`, а команда `dkms status` должна показать наличие установленного драйвера в системе. Пример вывода `dkms status` :

```
forward/<version>, 6.17.9-arch1-1, x86_64: installed
```

Обновление прошивки

Вместе с драйвером в систему (директория `/lib/firmware/forward`) устанавливаются актуальные версии прошивок для плат, которые гарантированно работают с данной версией драйвера. Поэтому, после установки драйверов рекомендуется проверить версии прошивок плат, и, возможно, обновить их.

Исключением является плата FD2110 - она прошивается динамически при каждом старте драйвера, поэтому ей обновление прошивки не требуется

Обновление прошивки делается с помощью утилиты `forward-flash` из пакета `forward-utils`.

forward-flash

Утилита проверит актуальность прошивок и предложит обновить прошивки, если требуется. После обновления утилита предложит перезагрузить плату для запуска новой прошивки.

Этот механизм не работает корректно на всех системах, особенно в случае виртуальных машин. Если после обновления плата не работает, или работает некорректно - требуется холодный перезапуск компьютера - то есть полное выключение и включение питания.

Использование драйвера

После установки и перезагрузки или выполнения `modprobe forward` в системе должно появиться устройство `/dev/forward/<плата>-<номер>`, например `/dev/forward/fd722-70802`. Это сервисное устройство, предназначенное для настройки общих параметров платы, обновления прошивки и т.д. Кроме того, в `sysfs` должна появиться директория с параметрами (атрибутами) платы, некоторые из которых можно менять.

Атрибуты в sysfs

Находятся в директории `/sys/class/forward/<плата>-<номер>/`. С точки зрения пользователя и других программ эти файлы являются одно-строчными текстовыми файлами, но фактически, это специальные псевдо-файлы, при обращении к которым вызываются обслуживающие функции внутри драйвера. При ручной настройке эти атрибут можно читать с помощью `cat`, а записывать с помощью `echo`, например:

```
cat /sys/class/forward/fd722-70802/temperature
```

выведет текущую температуру платы, а команда

```
echo ---- > /sys/class/forward/fd722-70802/io_config
```

выключит все входа и выхода платы (об этом ниже).

Примечание: запись атрибутов в sysfs может происходить только от суперпользователя (root), и нет хорошего способа дать права на sysfs обычным пользователям

Из полезных атрибутов в `sysfs` можно отметить следующие:

- `software_id (r)` - логический (серийный) номер плат
- `hardware_id (r)` - физический (неизменяемый) номер платы
- `temperature (r)` - температура платы
- `mode (rw)` - режим работы - это список под-драйверов которые сейчас обслуживают эту плату, обычно это `v4l2, alsa`. Пользователь может менять этот атрибут, например, для отключения ALSA на плате.
- `available_modes (r)` - доступный набор под-драйверов для платы
- `version (r)` - версия платы и версия прошивки в ней, в формате `<версия платы>r<версия прошивки>`
- `io_config (rw)` - режим работы входов-выходов. Атрибут содержит кол-во символов, равное количеству входов-выходов, каждый символ соответствует одному входу-выходу и может принимать только значения `I` (вход), `O` (выход) или `-` (выключено)

Включение-выключение и направление интерфейсов

У некоторых (FD788, FD722M2, FD2110) плат есть двунаправленные интерфейсы - они могут быть как входами так и выходами (но не одновременно). Для настройки направления интерфейса в `sysfs` существует атрибут `io_config`. Он представляет собой строку, состоящую из символов `I`, `O`, `-`, где каждый символ соответствует одному из интерфейсов (разъему) платы. Интерфейсы идут в порядке их физического расположения на плате: сверху-вниз если корпус в стойке или справа-налево если корпус компьютера вертикальный. Например, у платы FD788 по-умолчанию строка `io_config` имеет вид `IIIIIIII`, т.е. все интерфейсы SDI - входа, а у FD722 - `II00`, т.е. идут сначала два входа потом два выхода.

Интерфейсы могут иметь следующие состояния:

- `I` - вход
- `O` - выход
- `-` - выключен - в этом режиме интерфейс электрически отключен, в системе устройства не появляются.

Некоторые платы имеют фиксированные входы и фиксированные выходы, в этом случае интерфейс может быть только отключен. Другие могут быть переконфигурированы:

- FD788 - 8 интерфейсов SDI, любой может быть как входом, так и выходом. По-умолчанию `IIIIIIII`
- FD722M2 - 1 фиксированный вход, затем 1 конфигурируемый интерфейс, затем 1 выход. Соответственно, `io_config` может быть `IIO` либо `IOO`. По-умолчанию `IIO`.
- FD2110 - 2 конфигурируемых интерфейса SDI, затем 16 виртуальных каналов-входов первого ethernet, потом 16 каналов-выходов первого ethernet, затем то же самое для второго ethernet. Итого строка `io_config` имеет 66 символов. По-умолчанию, все интерфейсы выключены (`-`) кроме первых двух SDI, которые настроены как входы (`II`).

Переконфигурирование интерфейсов возможно с помощью команды `echo`, при этом на плате должны быть остановлены потоки на тех интерфейсах, которые перенастраиваются. Например, для настройки платы FD788 в режим 4 входа + 4 выхода можно выполнить следующую команду:

```
echo IIIIO000 > /sys/class/forward/fd788-80002/io_config
```

То же самое можно делать из своих приложений через работу с файлами (например `open()`, `write()` для языка C).

Сохранение атрибутов между перезагрузками

Система драйверов в Linux является stateless - драйвера не могут сохранять никакое состояние между перезапусками. Это означает, что после перезагрузки все сделанные настройки платы сбрасываются в первоначальные. Настройки, сделанные в sysfs можно сохранить в виде правил udev. Эти правила позволяют автоматически записывать атрибуты при появлении платы в системе.

Например, для настройки FD788 в режим 4 входа + 4 выхода при старте, можно создать файл `/etc/udev/rules.d/90-fd788.rules` со следующим содержимым:

```
SUBSYSTEM=="forward", ATTR{software_id}=="80002", ATTR{io_config}:"IIIIIO00"
```

Данное правило буквально говорит: "при появлении устройства класса forward, у которого атрибут `software_id` имеет значение 80002 записать в его атрибут `io_config` значение `IIIIIO00`". [Более подробно о правилах udev](#)

V4L2

Драйвер `forward-v4l2` предоставляет доступ к видео через стандартное API Video4Linux2 (V4L2). Данное API поддерживается многим сторонним ПО, например, OBS Studio, gstreamer, ffmpeg и т.д. На каждый интерфейс платы создается устройство `/dev/videoX`, где X - порядковый номер в системе. Например, если в компьютере стоит одна FD788 будут созданы `/dev/video0` - `/dev/video7`. При двух платах будут созданы `/dev/video0` - `/dev/video7` для первой платы и `/dev/video8` - `/dev/video15` для второй. При этом нумерация плат зависит от порядка их появления в системе, что чаще всего соответствует определенному порядку расположения плат в слотах - т.е. не меняется при перезагрузке системы.

Настройка видео происходит через стандартные утилиты из пакета `v4l-utils` - консольную `v4l2-ctl` и GUI `qv4l2`.

Настройка формата видео

В API V4L2 формат ("режим", "мода") видео задается двумя разными сущностями:

- Тайминг (timing) - задает ширину, высоту, частоту сигнала на **физическом интерфейсе**. Может меняться пользователем.
- Формат (format) - задает именно формат обмена **между приложением и драйвером**. О формате договариваются приложение и драйвер.

Для работы с таймингами можно использовать `v4l2-ctl`.

Получение списка доступных таймингов:

```
v4l2-ctl -d /dev/videoX --list-dv-timings
```

Данная команда выведет подробный список доступных для платы таймингов, каждый описан, например для 1080i50 тайминг выглядит следующим образом:

```
Index: 0
Active width: 1920
Active height: 1080
Total width: 2640
Total height: 1125
Frame format: interlaced
Polarities: +vsync +hsync
Pixelclock: 74250000 Hz (50.00 fields per second)
Horizontal frontporch: 0
Horizontal sync: 720
Horizontal backporch: 0
Field 1:
Vertical frontporch: 2
Vertical sync: 20
Vertical backporch: 0
Field 2:
Vertical frontporch: 3
Vertical sync: 20
Vertical backporch: 0
Standards: SDI
Flags: CE-video
```

Всего в платах доступны следующие тайминги:

1. 1080i50
2. 1080i60
3. 1080i59.94
4. 576i50
5. 480i59.94
6. 720p50
7. 720p60
8. 720p59.94
9. 1080p25
10. 1080p30
11. 1080p29.97
12. 1080p24
13. 1080p23.98
14. 1080p50
15. 1080p60

16. 1080p59.94
17. 2160p25
18. 2160p30
19. 2160p29.97
20. 2160p24
21. 2160p23.98
22. 2160p50
23. 2160p60
24. 2160p59.94

В связи с тем, что некоторые тайминги недоступны на некоторых платах или на некоторых интерфейсах, этот список может отличаться

Для установки тайминга на выходе или принудительной установки тайминга на входе нужно использовать следующую команду:

```
v4l2-ctl -d /dev/videoX --set-dv-bt-timings index=<индекс>
```

Где "индекс" - это порядковый номер тайминга в списке, полученном с помощью `--list-dv-timings`, например, 1080p50 обычно имеет номер 13

Если корректный сигнал присутствует на входе, можно узнать его тайминг:

```
v4l2-ctl -d /dev/video2 --query-dv-timings
```

Или попросить у драйвера автоматически определить и установить корректный тайминг на входе:

```
v4l2-ctl -d /dev/video2 --set-dv-bt-timings query
```

После настройки таймингов плата может работать со сторонним ПО.

Настройка тайминга может производиться также с помощью GUI-приложения `qv4l2`.

Mute на выходе

По-умолчанию интерфейсы появляются с электрически отключенными выходами - сигнал передается на выход плат, но при этом сам выход выключен и сигнал не появляется на разъеме. Выход включается автоматически в начале передачи картинки через V4L2-устройство, по окончании работы приложения устройство выключается обратно. При этом можно вручную включить/выключить выход с помощью `v4l2-ctl`

```
v4l2-ctl -d /dev/videoX -c mute_output=0
```

Включит выход, до момента старта приложений плата будет передавать черную картинку.

После первой установки `mute_output` автоматическое включение/выключение перестает работать

Настройка генлока

В платах с SDI-выходами существует возможность плавно регулировать частоту и синхронизировать выхода. Во всех платах имеется **один общий** генератор частоты, от которого работают все выхода. Этот генератор может работать либо от своего внутреннего источника (мастер) либо синхронизироваться к аналоговому видео-сигналу на отдельном аналоговом входе. Это называется аналоговый генлок, black-burst sync или tri-level sync. Также, в режиме мастер возможна подстройка частоты генератора вручную в пределах $\pm 500\text{ppm}$ ($\pm 0.05\%$) - это позволяет реализовывать синхронизацию к внешним источникам,

например NTP.

Помимо общего для всей платы аналогового генлока каждый выход может быть **независимо** синхронизирован к любому входу. Это называется цифровой генлок, в этом режиме выход начинать работать на частоте входного SDI, и, соответственно асинхронно к остальным выходам. Такой режим позволяет организовать передачу видео со входа на выхода без потери/вставки кадров (без time-based correction).

Настройка генлока осуществляется через контролы выходного v4l2-устройства с помощью утилиты `v4l2-ctl` или V4L2 API

```
v4l2-ctl -d /dev/videoX -c genlock_source=1
```

Устанавливает источник синхронизации для выхода (или всей платы). Возможные варианты:

1. Мастер
2. Аналоговый генлок-вход
3. Первый SDI-вход
4. Второй SDI-вход
5. ...

После установки источники можно посмотреть текущее состояние генлока

```
v4l2-ctl -d /dev/videoX -C genlock_state
```

Возможные состояния:

1. Master - выход в режиме мастера, генлок не включен
2. No input signal - нет входного сигнала для генлока, либо он не того формата (кадровая частота не совпадает с выходом)
3. Locking - выход подстраивает свою частоту под источник
4. Locked - выход подстроился под источник и синхронен
5. Holdover - выход когда-то был синхронен, но потом входной сигнал пропал, выход работает на последней запомненной частоте.

В режиме генлока каждый выход можно сдвигать по фазе относительно источника синхронизации в пределах $\pm \text{кадр}/2$ с точностью в 6.734ns (1/148500000с). Делается это также через контролы:

```
v4l2-ctl -d /dev/videoX -C genlock_offset_6_734ns=1000000
```

Команда выше сдвинет картинку на 6.734ms в будущее, т.е. выходной кадр будет запаздывать на 6.734ms относительно входа.

Другие настройки V4L2

Полный список контролов, доступных плате можно посмотреть следующей командой:

```
v4l2-ctl -d /dev/videoX -l
```

Получить значение контрола:

```
v4l2-ctl -d /dev/videoX -C enable_vanc
```

Установить значение контрола:

```
v4l2-ctl -d /dev/videoX -c enable_vanc=1
```

Также, приложение `qv4l2` позволяет всё это делать из графического интерфейса.

Часть контролов является сервисными или предназначены для использования из приложений, но есть полезные пользовательские контролы:

- `analog_input_mode` - режим работы аналогового входа (генлок-входа), 0 - генлок/получение VITC, 1 - получение PPS, 2 - получение LTC
- `bypass_disable_forced` - для платы FD722BP принудительно выключить аппаратное пропускание сигнала
- `enable_vanc` - включить приём VBI-данных, после этого начинают корректно работать устройства `/dev/vbiX`
- `frequency_adjust_ppt` - изменить частоту несущей для передачи видео
- `genlock_source` - источник синхронизации для выхода. Может быть `Master (0)` , `Analog (1)` или конкретный `SDI (N + 1)`
- `genlock_state` - текущее состояние синхронизации. Могут быть `Master (0)` , `No input (1)` , `Locking (2)` , `Locked (3)` , `Holdover (4)`
- `genlock_offset_6_734ns` - сдвиг выхода относительно источника синхронизации, в пикселях 3G-SDI (1/148500000 с).
- `cloned_output` - включить клонирование сигнала. Четные (2, 4, 6, 8) выхода плат FD722, FD788, FD922 могут вставать в режим клонирования, когда на них выводится копия сигналов выходов 1, 3, 5, 7
- `mute_output` - электрически выключить ("замыютить") выход. При этом само V4L2 устройство и плата продолжают работать, но на интерфейсе остается электрическая тишина (Hi-Z).

Более подробно о контролах V4L2 рассказывается в разделе SDK.

ALSA

Драйвер `forward-alsa` предоставляет доступ к аудио через стандартное API ALSA. Данное API поддерживается многим сторонним ПО, например, `pulseaudio`, `OBS Studio`, `gststreamer`, `ffmpeg` и т.д. На каждый интерфейс платы создается ALSA-устройство `hw:Y,X`, где `Y` - порядковый номер платы (`card` в ALSA) в системе, `X` - номер выхода (`device` в ALSA). В отличие от V4L2, ALSA-устройства всегда имеют фиксированный номер `X`, зависящий только от физического расположения на плате. Например, если в компьютере стоит одна FD788 и нет других ALSA-карточек будут созданы `hw:0,0` - `hw:0,7`. При двух платах будут созданы `hw:0,0` - `hw:0,7` для первой платы и `hw:1,0` - `hw:1,7` для второй. При этом нумерация плат зависит от порядка их появления в системе, что чаще всего соответствует определённому порядку расположения плат в слотах - т.е. не меняется при перезагрузке системы.

Настройка аудио не нужна, тестирование звука возможно происходит через стандартные утилиты из пакета `alsa-utils`.

Чтобы получить список входных устройств, можно воспользоваться утилитой `arecord`:

```
arecord -l
```

Типичный вывод:

```
card 0: Intel [HDA Intel], device 0: Generic Analog [Generic Analog]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 0: IN [IN 1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

```
card 1: F80002 [FD788 80002], device 1: IN [IN 2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 2: IN [IN 3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 3: IN [IN 4]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 4: IN [IN 5]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 5: IN [IN 6]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 6: IN [IN 7]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: F80002 [FD788 80002], device 7: IN [IN 8]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Здесь `hw:0,0` - встроенная системная звуковая карточка, `hw:1,0` - `hw:1,7` - плата FD788 с номером 80002.

Чтобы получить список выходных устройств, можно воспользоваться утилитой `aplay` :

```
aplay -l
```

Pulseaudio/pipewire

Поскольку ALSA-устройства являются стандартным способом работы со звуком в Linux, стандартные компоненты системы начинают считать платы FD обычными звуковыми карточками и пытаются их использовать в качестве обычных динамиков для вывода звука и микрофонов для захвата. Вместе с драйверами устанавливается набор alsa-профилей (`/usr/share/alsa-card-profile`), которые препятствуют такому поведению, однако иногда (особенно если плата - единственное звуковое устройство) pulseaudio всё же захватывает устройство и не даёт работать остальному ПО. В этом случае необходимо в настройках системы найти раздел звук и выставить профиль "Отключено" для платы FD.

Примеры использования gstreamer

gstreamer - мультимедийный фреймворк, работающий в терминах конвейров (пайплайнов, графов), позволяющий производить обработку видео в реальном времени. Работа с драйвером forward возможна через стандартные компоненты `v4l2src` , `v4l2sink` , `alsasrc` , `alsasink` .

Единственное ограничение - драйвер в качестве синхронизации alsa использует v4l2-устройство, а gstreamer по-умолчанию при использовании alsasink/alsasrc делает их опорными часами. В некоторых графах это приводит к зависанию (deadlock), поэтому в этом случае у компонентов alsa нужно выставлять параметр `provide-clock=false`.

Захват (capture)

Захват видео и аудио и вывод на экран/системные динамики:

```
v4l2-ctl -d /dev/videoX --set-dv-bt-timings query
gst-launch-1.0 v4l2src device=/dev/videoX ! queue ! autovideosink alsasrc
device=hw:Y,X ! queue ! autoaudiosink
```

Захват видео и аудио и кодирование в файл:

```
v4l2-ctl -d /dev/videoX --set-dv-bt-timings query
gst-launch-1.0 -e v4l2src device=/dev/videoX ! queue ! videoconvert ! x264enc
! queue ! mux. \
                alsasrc device=hw:Y,X ! queue ! audioconvert ! opusenc ! queue
! mux. \
                mp4mux name=mux ! filesink location=/tmp/capture.mp4
```

Вывод (playback)

Вывод тестового видео на выход:

```
gst-launch-1.0 videotestsrc ! "video/x-raw,interlace-mode=interleaved" !
queue ! v4l2sink device=/dev/videoX audiotestsrc ! queue ! alsasink
device=hw:Y,X
```

"video/x-raw,interlace-mode=interleaved" здесь нужно только для форматов с чересстрочной разверткой. По каким-то причинам videotestsrc не может автоматически определить чересстрочность.

Обработка проходящего видео

Для примера возьмём видео с первого входа FD722, наложим на него текст с временем потока и отправим на первый выход. При этом включим генлок выхода к первому входу:

```
v4l2-ctl -d /dev/video2 -c genlock_source=2
gst-launch-1.0 v4l2src device=/dev/video0 ! queue ! timeoverlay ! queue !
v4l2sink device=/dev/video2 \
                alsasrc provide-clock=false device=hw:0,0 ! queue ! alsasink
provide-clock=false device=hw:0,2
```

На первом выходе FD722 появится сигнал со входа с наложенным временем

SDK

Драйвер forward реализует стандартное API V4L2 и ALSA с некоторым дополнительным функционалом. Поэтому SDK состоит из примеров использования этих API. Документация V4L2 API доступна на kernel.org Общение с подсистемой ALSA происходит через библиотеку libasound, документация доступна на alsa-project.org Все необходимое для нестандартного функционала определено в заголовочном файле `/usr/src/forward-<version>/include/forward-v4l2-ioctl.h`

Список примеров

Вместе с драйвером поставляется архив с примерами использования V4L2 API с драйвером `forward`. Примеры написаны на C++.

- `common` - небольшая библиотека с общими функциями, в т.ч. C++-обертка для V4L2-устройства
- `enumerate` - пример использования libudev для поиска устройств forward и сопоставления v4l2-устройств с ALSA-устройствами
- `io-switch` - пример переключения направления выходов (записи в sysfs) из кода
- `timings` - пример работы с таймингами
- `genlock` - пример включения генлока
- `capture-simple` - пример захвата видео
- `playback-simple` - пример проигрывания видео
- `capture-vbi-audio` - пример захвата видео, vbi и аудио
- `playback-vbi-audio` - пример проигрывания видео, vbi и аудио
- `capture-complex` - пример захвата видео, vbi и аудио через единый v4l2-интерфейс
- `playback-complex` - пример проигрывания видео, vbi и аудио через единый v4l2-интерфейс
- `capture-fd2110` - пример захвата видео по сети для платы FD2110
- `playback-fd2110` - пример проигрывания видео по сети для платы FD2110
- `capture-raw` - пример захвата "сырого" потока SDI в файл
- `playback-raw` - пример проигрывания "сырого" потока SDI из файла
- `capture-preview` - пример захвата видео и отображения его в Qt-окне
- `playback-watchdog-bypass` - пример воспроизведения видео с использованием платы FD722BP и сторожевого таймера
- `passthrough-simple` - пример пропуска видео со входа на выход с обработкой
- `av-delay-generator` - генератор тестового сигнала для проверки расхождения видео с аудио
- `av-delay-viewer` - просмотр тестового сигнала для проверки расхождения видео с аудио
- `hw-timestamps` - пример получения таймштампов на основе часов платы
- `analog-pps` - пример получения таймштампов на основе часов платы от сигнала PPS на аналоговом входе
- `analog-tc` - пример получения таймкода с аналоговым входа VITC или LTC
- `capture-asi-raw` - пример захвата "сырого" потока ASI, декодирование и запись в файл
- `playback-asi-raw` - пример проигрывания "сырого" потока ASI с кодированием из файла
- `passthrough-asi` - пример захвата "сырого" потока ASI, декодирование и передача на выход

Принципы работы с V4L2

Общение с драйвером идёт через файлы-устройства `/dev/videoX` посредством системного вызова `ioctl()`. В вызов `ioctl()` передается код операции (`VIDIOC_*`) и указатель на структуру, через которую происходит обмен информацией. Через `ioctl()` производится настройка тайминга, формата, изменение контролов, выделение буферов и т.д. Драйвер `forward` начиная с версии `1.7.0` использует мультипланарный API `Video4Linux2`.

Ввод и вывод видео происходит в следующем порядке:

1. Открытие устройства с помощью `open()`
2. Запрос, установка тайминга входа/выхода с помощью `ioctl VIDIOC_ENUM_DV_TIMINGS , VIDIOC_QUERY_DV_TIMINGS , VIDIOC_S_DV_TIMINGS`
3. Запрос, установка формата для общения с драйвером - `VIDIOC_ENUM_FMT , VIDIOC_G_FMT , VIDIOC_S_FMT`
4. Выделение буферов с помощью `VIDIOC_REQBUFS , VIDIOC_QUERYBUF`
5. Предварительное заполнение очереди буферов (pre-roll) - `VIDIOC_QBUF`
6. Включение проигрывания - `VIDIOC_STREAMON`
7. Цикл получения буферов из очереди, обработки, и передачи обратно в очередь - `VIDIOC_DQBUF , VIDIOC_QBUF`
8. Остановка проигрывания - `VIDIOC_STREAMOFF`
9. Освобождение буферов - `VIDIOC_REQBUFS`

Тайминги и форматы

В V4L2 есть две сущности, определяющие параметры видео:

- *Тайминг* - определяет параметры сигнала на входе/выходе платы - размеры, частоту, размер обратного хода.
- *Формат* - определяет параметры видео в буферах - размер, формат пикселей, stride и т.д.

Тайминг описывается структурой `v4l2_bt_timings`, в ней указывается геометрические размеры видимой части, размеры синхро-областей, несущая частота и т.д. Для получения списка всех возможных таймингов необходимо их перечислить с помощью `ioctl VIDIOC_ENUM_DV_TIMINGS`. В этот `ioctl` передается по-очереди индекс тайминга, начиная с 0, до тех пор пока `ioctl` не вернет код `-EINVAL`. Для входа также возможно запросить текущий тайминг на входе с помощью `VIDIOC_QUERY_DV_TIMINGS`.

Оба этих `ioctl` возвращают структуру с таймингом, которую затем можно передать в `VIDIOC_S_DV_TIMINGS` для установки параметров сигнала. Вызов `VIDIOC_G_DV_TIMINGS` получает текущий тайминг на устройстве (тот, который установили с помощью `VIDIOC_S_DV_TIMINGS`, а не тот, который на самом деле на входе). Входное устройство не обновляет тайминг при смене сигнала на входе, это нужно делать вручную. После запуска потока (`VIDIOC_STREAMON`) смена тайминга невозможна.

Формат описывается структурой `v4l2_pix_format_mplane` в ней описывается размер видео, формат пикселей, размеры плейнов, способ передачи полей и т.д. Формат пикселей является 4-байтной константой, более известной как FourCC, он же является уникальным идентификатором формата. В случае чересстрочных таймингов формат также позволяет выбрать способ хранения полей в буфере:

- `V4L2_FIELD_INTERLACED` - в буфере хранятся оба поля чересстрочно, первая строка является первой строчкой первого поля
- `V4L2_FIELD_ALTERNATE` - в буфере хранится только одно поле, кадр состоит из двух буферов

В текущей реализации `V4L2_FIELD_INTERLACED` требует дополнительного копирования данных, а `V4L2_FIELD_ALTERNATE` является zero-copy.

Вызов `VIDIOC_ENUM_FMT` работает по такому же принципу, как и `VIDIOC_ENUM_DV_TIMINGS` и возвращает список FourCC форматов, доступных для данного тайминга. Для установки формата необходимо его сначала получить (`VIDIOC_G_FMT`), затем поменять поля в структуре (обычно, только FourCC), затем установить (`VIDIOC_S_FMT`). API устроено таким образом, что `VIDIOC_S_FMT` считается рекомендацией драйверу, он имеет право подстроить формат под себя и вернуть в приложение измененный формат.

Драйвер `forward` реализует как стандартные форматы (YUYV, V210 и т.д.), так и имеет набор специализированных форматов (Raw, ASI, Complex) для расширенных функций.

Буфера и очереди

Само видео передается в терминах очередей и буферов. Буфер - сущность, содержащая один кадр (или поле), выделенная в ядре через вызов `ioctl()` и спроецированная (`mmap()`) в адресное пространство приложения. Буфер может иметь одну или несколько плоскостей (плейн, `plane`), каждая плоскость - непрерывная (виртуально) область в памяти, определенного (форматом) размера. В простых случаях плоскость одна и она содержит картинку в определенного формата, например при формате 8-bit YUYV (`V4L2_PIX_FMT_YUYV`) в единственном плейне будут лежать пиксели YUV в формате 4:2:2, по 16 бит на пиксель.

Помимо самого изображения буфера хранят некоторую мета-информацию:

- Порядковый номер (`sequence`) - счетчик кадров для отслеживания потерь
- Таймштамп - системное время прихода/отправки кадра
- Таймкод (VITC/LTC)
- В некоторых форматах также в каждом буфере указано, сколько реально данных использовано в каждом плейне (`bytesused`)

Выделение буферов происходит через `ioctl VIDIOC_REQBUFS`, в параметрах передается желаемое количество буферов (кадров). Каждый буфер имеет свой уникальный идентификатор - порядковый номер, и во всех вызовах в качестве параметра-буфера передается его номер. После выделения буферов нужно спроецировать буфера на адресное пространство приложения и получить указатели на данные в плейнах. Для этого нужно:

- Для каждого буфера вызвать `VIDIOC_QUERYBUF` чтобы получить подробную информацию о количестве плоскостей, их размер и оффсет для `mmap`
- Сделать `mmap()` для каждого плейна каждого буфера по аналогии с разделяемой памятью для получения доступа к данным

После этого буфера готовы к использованию в драйвере, при этом приложение также имеет доступ к видео-данным. Для освобождения буферов нужно выполнить `VIDIOC_REQBUFS` передав `0` в качестве количества буферов.

Обмен данными с драйвером и платой происходит в терминах очередей. Каждое устройство имеет очередь из буферов, приложение добавляет буфер в очередь с помощью `ioctl VIDIOC_QBUF`. После этого буфер переходит под контроль драйвера и по мере работы буфера передаются на вывод (ввод) платы. После того, как плата закончила работу с буфером, он может быть возвращен приложением. Для этого используется `ioctl VIDIOC_DQBUF` - это блокирующий вызов, ожидающий появления первого обработанного буфера от драйвера. При открытии устройства в неблокирующем режиме (параметр `O_NONBLOCK` для `open()`) `VIDIOC_DQBUF` не блокируется, а возвращает ошибку если буфер не готов. В этом случае необходимо использовать системный вызов `poll()`. После `VIDIOC_DQBUF` данные готовы к обработке приложением.

Итого, ввод/вывод данных с помощью очередей выглядит так:

1. Выделение буферов (`VIDIOC_REQBUFS`)
2. Подготовка буферов (`VIDIOC_QUERYBUF`)
3. Предварительное заполнение очереди (`preroll`) (`VIDIOC_QBUF`)
4. Старт потока (`VIDIOC_STREAMON`)
5. В цикле:
 1. Ожидание готовности буфера (`poll()`)
 2. Получение буфера из драйвера (`VIDIOC_DQBUF`)
 3. Обработка
 4. Возвращение буфера в очередь (`VIDIOC_QBUF`)
6. Остановка потока (`VIDIOC_STREAMOFF`)
7. Освобождение буферов (`VIDIOC_REQBUFS`)

Для корректной работы плат требуется, чтобы в очереди буферов всегда находилось как минимум 2 буфера - с одним плата работает, второй заранее подготовлен. Это означает, что требуется как минимум 3 буфера для работы

Если в момент конца кадра в очереди не находится буфера для передачи плате, происходит событие потери кадра. Для входа кадр пропускается, для выхода берется последний буфер (при наличии) и повторяется. Определить потерю кадров можно с помощью поля sequence у буферов - это счетчик кадров на плате, если после очередного VIDIOC_DQBUF sequence изменился больше чем на 1 - это означает, что некоторое количество кадров было потеряно.

VBI

Вместе с video-устройствами /dev/videoX драйвер создает стандартные устройства VBI /dev/vbiX . Через эти устройства происходит передача VBI (Vertical Blanking Interval) области. Работа с ними похожа на работу с /dev/videoX - установка формата, аллокация буферов, очереди. В качестве формата выступает структура v4l2_vbi_format , в которой передается кол-во линий и пиксельный формат для VBI. В текущих платах пиксельный формат для VBI - V4L2_PIX_FMT_Y10 , все пиксели 10-бит упакованы в 16-бит, с нулевой старшей частью. Для работы VBI требуется явное включение контроля V4L2_CID_FORWARD_ENABLE_VANC у устройства /dev/videoX .

Режим raw

В платах с интерфейсами SDI возможен прием и вывод всего 10-битного потока SDI потока целиком, со всеми областями синхронизации и пакетами. Это можно использовать для анализа SDI потока, его записи и дальнейшего проигрывания, а также для продвинутой обработки видео с минимальным изменением содержимого. Режим raw устанавливается с помощью установки формата с pixelformat = V4L2_PIX_FMT_SL_RAW на устройстве V4L2.

После установки формата, в структуре формата будут описаны параметры сырого потока - полные ширина, высота и размер в байтах. Для чересстрочных форматов, режим raw может работать только в V4L2_FIELD_ALTERNATE , т.е. каждое поле будет в отдельном буфере. Сами данные в буферах лежат в виде 10-битных слов, плотно упакованных по 4 штуки в 5 байт (40 бит). Порядок потоков SDI - сначала CbCr, потом Y. Порядок слов - little-endian, т.е. в первом байте лежат младшие 8 бит от первого принятого 10-битного слова:

Байт\Бит	7	0
0	Cb0[7:0]	
1	Y0[5:0]	Cb0[9:8]
2	Cr0[3:0]	Y0[9:6]
3	Y1[1:0]	Cr0[9:4]
4	Y1[9:2]	

Поскольку на выход поток передается "как есть" задача софта - правильно сгенерировать все требуемые маркеры SDI-потока - SAV, EAV и так далее. Любое нарушение в потоке (например, значение яркости/цвета за пределами диапазона 0x004 - 0x3FB) приведет к потере сигнала на приемнике. Примеры capture-raw и playback-raw показывают пример захвата потока в файл и проигрывание из файла, соответственно.

ASI

В платах с интерфейсами SDI возможен прием и вывод потока DVB ASI. На входе возможен прием потока в двух форматах:

- V4L2_PIX_FMT_MPEG - в каждом буфере будут декодированные пакеты MPEG-TS. Поле bytesused будет указывать на количество принятых байт.
- V4L2_PIX_FMT_SL_RAW_ASI - в каждом буфере будет полный не декодированный 8b10b поток, со всеми K-символами. Упаковка - такая же, как в режиме raw - 4 слова в 5 байтах. Поле

`bytesused` будет указывать на количество байт.

На выходе возможен только формат `V4L2_PIX_FMT_SL_RAW_ASI`, при этом 10-битные слова - это не кодировка 8b10b, а внутренний формат, в котором биты [7:0] - данные, а бит [8] указывает, является ли слово K-символом (1) или данными (0).

При обнаружении сигнала ASI на входе драйвер автоматически выставляет формат `V4L2_PIX_FMT_MPEG` - таким образом можно определить наличие сигнала ASI через `VIDIOC_G_FMT`. Если на входе ASI переключился на SDI, драйвер автоматически возвращает формат `V4L2_PIX_FMT_YUYV`.

При приеме/передаче ASI платы работают интервалами (пачками) в 20ms, то есть, каждый буфер соответствует времени потока в 20ms или 540000 символов. Для уменьшения задержек есть возможность уменьшить интервал вплоть до 2.5 через контрол `V4L2_CID_FORWARD_ASI_PERIOD`.

Режим complex

В связи с тем, что устройства ALSA и V4L2 являются фактически независимыми друг от друга, точная синхронизация аудио со звуком является нетривиальной задачей, особенно в случае проблем с сигналом на входе и пропаданием звука. Для упрощения синхронизации был реализован нестандартный режим для V4L2 - в виде форматов `V4L2_PIX_FMT_SL_COMPLEX_8BIT` и `V4L2_PIX_FMT_SL_COMPLEX_10BIT`. Кадр (поле) в этом формате состоит из трёх плоскостей (плейнов):

1. Само видео, 8 бит UYVY (`V4L2_PIX_FMT_SL_COMPLEX_8BIT`) или плотно-упакованные 10 бит UYVY (`V4L2_PIX_FMT_SL_COMPLEX_10BIT`).
2. Вход - все принятые пакеты SDI/HDMI/RTP за кадр, включая аудио. Выход - аудио, соответствующее кадру, в формате потока.
3. Структура `forward_v4l2_complex_info`, описывающая текущий кадр и статистику - заполняется драйвером.

В этом режиме драйвер не производит обработку данных - они попадают в приложение/плату в том формате, который использует плата. Для чересстрочных форматов, режим complex может работать только в `V4L2_FIELD_ALTERNATE`, т.е. каждое поле будет в отдельном буфере. Например, если включен VBI (через контрол `V4L2_CID_FORWARD_ENABLE_VANC`), то в первом плейне всегда в начале будет область VBI в 10 битном формате. Размер этой области можно подсчитать через тайминг (поля `.bt.vsync` + `.bt.vbackporch`), а на входном устройстве - в структуре `forward_v4l2_complex_info`.

Для понимания того, какой формат ожидать во втором плейне необходимо использовать контрол `V4L2_CID_FORWARD_COMPLEX_AUDIO_FORMAT`:

- `V4L_FORWARD_COMPLEX_AUDIO_RAW_32` (0) - 24 бита звука в 4 байтах, биты в младшей части (little-endian). Используется для выхода у SDI-плат.
- `V4L_FORWARD_COMPLEX_AUDIO_RAW_24BE` (1) - 24 бита в 3 байтах, Big-endian, используется на выходе FD2110
- `V4L_FORWARD_COMPLEX_AUDIO_ST272` (2) - Аудио в пакетах по стандарту ST-272 (SD-SDI аудио). Каждое слово в буфере - 16 бит, содержащие 10-битные данные пакетов по стандарту SMPTE ST-291.
- `V4L_FORWARD_COMPLEX_AUDIO_ST299` (3) - Аудио в пакетах по стандарту ST-299 (HD-SDI аудио). Каждое слово в буфере - 16 бит, содержащие 10-битные данные пакетов по стандарту SMPTE ST-291.
- `V4L_FORWARD_COMPLEX_AUDIO_HDMI` (4) - Аудио в пакетах по стандарту HDMI
- `V4L_FORWARD_COMPLEX_AUDIO_ST2110_30` (5) - Аудио в пакетах по стандарту ST2110-30
- `V4L_FORWARD_COMPLEX_AUDIO_AES3` (6) - 4-байтные aes-фреймы, используется в FD788 при приеме AES3id.

Для пакетных форматов в библиотеке `common` приложены соответствующие парсеры, вытаскивающие

готовые аудио сэмплы. Стоит отметить, что форматы `V4L2_FORWARD_COMPLEX_AUDIO_ST272` и `V4L2_FORWARD_COMPLEX_AUDIO_ST299` помимо аудио позволяют принимать любые пакеты SMPTE ST-291 из потока SDI.

События

Драйвер посылает события через стандартный интерфейс (API) V4L2. Драйвер посылает следующие события:

- `V4L2_EVENT_SOURCE_CHANGE` - стандартное событие смены формата
- `V4L2_EVENT_VSYNC` - стандартное событие синхронизации, посылается раз в поле, вне зависимости от того, включено проигрывание или нет.
- `V4L2_EVENT_FRAME_SYNC` - стандартное событие синхронизации, посылается раз в кадр перед тем, как отдать буфер в приложение при включенном проигрывании. Содержит текущий номер кадра.
- `V4L2_EVENT_FORWARD_TIMESTAMP` - специфичное для драйвера событие, содержит текущий таймштамп кадра, посылается раз в поле, вне зависимости от того, включено проигрывание или нет. В событии передается структура `struct v4l2_event_forward_timestamp` со следующими полями:
 - `buffer_sequence` - текущий номер кадра, совпадает с буфером и `V4L2_EVENT_FRAME_SYNC`
 - `field` - текущее поле, `V4L2_FIELD_*`
 - `eof_timestamp` - аппаратный таймштамп поля/кадра, формат тот же самый, что и в `V4L2_CID_FORWARD_HW_TIMESTAMP`
 - `irq_timestamp` - аппаратный таймштамп прерывания - время когда информация дошла до драйвера, формат тот же самый, что и в `V4L2_CID_FORWARD_HW_TIMESTAMP`

Список контролов V4L2

Данные контролы доступны через стандартный интерфейс (API) V4L2 и могут быть изменены с помощью стандартных `v4l2-ctl` и `qv4l2`. В заголовочном файле `/usr/src/forward-<version>/include/forward-v4l2-ioctl.h` имеются все необходимые объявления для использования этих контролов в своём коде. Большинство контролов доступно как для входов, так и для выходов. Также, большинство из них можно менять во время приема/передачи потока, без остановки. Некоторые контролы являются общими для всей платы - при изменении в одном устройстве они меняются для всех устройств.

- `V4L2_CID_FORWARD_ENABLE_VANC` (`bool`) - включить захват / проигрывание VANC-области **только при отключеном проигрывании**
- `V4L2_CID_FORWARD_WIDESCREEN` (`bool`) - входной сигнал имеет флаг widescreen в VPID / поставить флаг widescreen на выходе
- `V4L2_CID_FORWARD_HW_TIMESTAMP` (`u32`) - получить аппаратное время. На платах имеется счетчик, который работает на частоте 148.5MHz, синхронно с выходами - аппаратные часы. **только чтение**
- `V4L2_CID_FORWARD_FREQUENCY_ADJUST` (`u64`) - подстройка выходной частоты в prpb (point-per-trillion, 10^{-12}). Записывая значения в этот контрол можно плавно (без потери сигнала на выходе) поменять несущую частоту на выходе - $F_{out} = V4L2_CID_FORWARD_FREQUENCY_ADJUST / 1e-12 * F_{base}$. Не рекомендуется менять значение больше чем на 10ppm ($1e-6$) за один раз - это может привести к потере сигнала на входах и выходах на несколько секунд. Предел регулирования - $\pm 500ppm$ **общий для всей платы**
- `V4L2_CID_FORWARD_ASI_PERIOD` (`enum`) - переключить период приема/выдачи данных (буферов) из драйвера для ASI **только при отключеном проигрывании**
 - `20ms (0)` - по-умолчанию
 - `10ms (1)`

- 5ms (2)
- 2.5ms (3)
- V4L2_CID_FORWARD_TIMECODE (enum) - вид таймкода для приема на входе/вид таймкода который нужно выдать.
 - None (0) - не принимать/не выдавать таймкод - *по-умолчанию*
 - LTC (1) - принимать только LTC / выдавать LTC
 - VITC (2) - принимать только VITC / выдавать VITC
 - Any (3) - принимать любой таймкод / невалидно для выхода
- V4L2_CID_FORWARD_GENLOCK_SOURCE (enum) - источник генлока, привязка несущей частоты.
 - V4L_FORWARD_GENLOCK_SRC_MASTER (0) - мастер, несущая частота не зависит от входов, её можно изменить с помощью V4L2_CID_FORWARD_FREQUENCY_ADJUST , *по-умолчанию*
 - V4L_FORWARD_GENLOCK_SRC_ANALOG (1) - привязать несущую частоту к аналоговому входу (black-burst/tri-level-sync) **общий для всей платы**
 - V4L_FORWARD_GENLOCK_SRC_IN0 (2) - привязать несущую частоту к первому входу SDI
 - V4L_FORWARD_GENLOCK_SRC_IN1 (3) - привязать несущую частоту ко второму входу SDI
 - V4L_FORWARD_GENLOCK_SRC_INX (1 + X) - привязать несущую частоту ко X-му входу SDI
- V4L2_CID_FORWARD_GENLOCK_STATE (enum) - текущее состояние генлока **только чтение**
 - V4L_FORWARD_GENLOCK_MASTER (0) - в режиме мастер
 - V4L_FORWARD_GENLOCK_NO_INPUT_SIGNAL (1) - входной сигнал-источник генлока не обнаружен
 - V4L_FORWARD_GENLOCK_LOCKING (2) - идёт подстройка частоты и фазы
 - V4L_FORWARD_GENLOCK_LOCKED (3) - выхода в генлоке, всё хорошо
 - V4L_FORWARD_GENLOCK_HOLDOVER (3) - плата была в генлоке, а потом пропал сигнал, плата пытается удерживать частоту
- V4L2_CID_FORWARD_GENLOCK_OFFSET (u32) - сдвиг выходного кадра относительно источника генлока. Единица - $6.734\text{ns} = 1/148.5\text{MHz} = 1 \text{ пиксель формата } 1080\text{p}50/\text{p}60$
только для выхода
- V4L2_CID_FORWARD_ANALOG_RX_MODE (enum) - режим работы аналогового входа платы **общий для всей платы**
 - Genlock (0) - аналоговая видео-синхронизация - *по-умолчанию*
 - PPS (1) - прием сигнала PPS (1Hz)
 - LTC (1) - прием LTC-таймкода
- V4L2_CID_FORWARD_ANALOG_RX_TIMESTAMP (struct) - возвращает текущее состояние аналогового входа. Возвращается структура v4l2_forward_analog_rx_timestamp , которая содержит признак наличия сигнала на входе, таймштамп последнего события, текущее время платы и системы. **только чтение общий для всей платы**
- V4L2_CID_FORWARD_ANALOG_RX_TIMECODE (struct) - возвращает текущий тайм-код на аналоговом входе, возвращается структура struct v4l2_timecode . **только чтение общий для всей платы**
- V4L2_CID_FORWARD_BYPASS_DISABLE (bool) - выключить аппаратное пропускание сигнала для платы FD722BP. Этот контрол должен записываться не реже чем раз в V4L2_FORWARD_BYPASS_WATCHDOG_PERIOD_MS миллисекунд.
- V4L2_CID_FORWARD_BYPASS_DISABLE_FORCE (bool) - выключить аппаратное пропускание сигнала для платы FD722BP навсегда.
- V4L2_CID_FORWARD_WATCHDOG_ENABLE (bool) - включить аппаратное отключение выхода при отсутствии записи в V4L2_CID_FORWARD_WATCHDOG_KEEP_ALIVE .
- V4L2_CID_FORWARD_WATCHDOG_KEEP_ALIVE (u32) - отсрочить отключение выхода на X

миллисекунд.

- `V4L2_CID_FORWARD_CLONED_OUTPUT` (`bool`) - клонировать выход, работает только на четных выходах у плат с SDI.
- `V4L2_CID_FORWARD_MUTE_OUTPUT` (`bool`) - электрически выключить выход, сам выход продолжит работать, но на разъеме сигнала не будет.
- `V4L2_CID_FORWARD_COMPLEX_AUDIO_GROUPS` (`u32`) - количество аудио-групп (аудио каналов * 4) в режиме complex.
- `V4L2_CID_FORWARD_COMPLEX_AUDIO_FORMAT` (`enum`) - тип аудио для режима complex

только чтение

- `V4L_FORWARD_COMPLEX_AUDIO_RAW_32` (0) - 24 бита в 4 байтах, биты в младшей части
- `V4L_FORWARD_COMPLEX_AUDIO_RAW_24BE` (1) - 24 бита в 3 байтах, Big-endian
- `V4L_FORWARD_COMPLEX_AUDIO_ST272` (2) - Аудио в пакетах по стандарту ST-272 (SD-SDI аудио)
- `V4L_FORWARD_COMPLEX_AUDIO_ST299` (3) - Аудио в пакетах по стандарту ST-299 (HD-SDI аудио)
- `V4L_FORWARD_COMPLEX_AUDIO_HDMI` (4) - Аудио в пакетах по стандарту HDMI
- `V4L_FORWARD_COMPLEX_AUDIO_ST2110_30` (5) - Аудио в пакетах по стандарту ST2110-30
- `V4L_FORWARD_COMPLEX_AUDIO_AES3` (6) - 4-байтные aes-фреймы

Для платы FD2110 предусмотрены дополнительные контролы:

- `V4L2_CID_FORWARD_FD2110_STREAM_BPC` (`enum`) - количество бит на пиксель в потоке ST-2110 (не в формате v4l2!), может быть `V4L_FORWARD_FD2110_STREAM_BPC_8BIT` или `V4L_FORWARD_FD2110_STREAM_BPC_10BIT`
- `V4L2_CID_FORWARD_FD2110_STREAM_COLOR` (`enum`) - формат пикселя в потоке ST-2110 (не в формате v4l2!), пока может быть только `V4L_FORWARD_FD2110_STREAM_COLOR_YCBCR422`
- `V4L2_CID_FORWARD_FD2110_STREAM_ADDRESS_STRING` (`string`) - адрес потока в формате `ipv4:port` или `[ipv6]:port`, может быть multicast или unicast-адресом
- `V4L2_CID_FORWARD_FD2110_STREAM_ADDRESS` (`struct`) - тоже самое, что и `V4L2_CID_FORWARD_FD2110_STREAM_ADDRESS_STRING` но в виде бинарной структуры `v4l2_forward_stream_address`
- `V4L2_CID_FORWARD_FD2110_STREAM_SOURCE_PORT` (`u16`) - для выхода, исходящий порт
- `V4L2_CID_FORWARD_FD2110_STREAM_PT` (`u8`) - для выхода, поле Payload Type для RTP-пакетов
- `V4L2_CID_FORWARD_FD2110_STREAM_SSRC` (`u32`) - для выхода, поле SSRC для RTP-пакетов
- `V4L2_CID_FORWARD_FD2110_ST2022_7_TX_ENABLE` (`bool`) - для выхода, включить дублирование потоков по стандарту ST2022-7.
- `V4L2_CID_FORWARD_FD2110_STREAM_AUDIO_ADDRESS_STRING` (`string`) - тоже самое что и `V4L2_CID_FORWARD_FD2110_STREAM_ADDRESS_STRING` но для аудио в режиме complex
- `V4L2_CID_FORWARD_FD2110_STREAM_AUDIO_ADDRESS` (`struct`) - тоже самое что и `V4L2_CID_FORWARD_FD2110_STREAM_ADDRESS` но для аудио в режиме complex
- `V4L2_CID_FORWARD_FD2110_STREAM_AUDIO_SOURCE_PORT` (`u16`) - тоже самое что и `V4L2_CID_FORWARD_FD2110_STREAM_SOURCE_PORT` но для аудио в режиме complex
- `V4L2_CID_FORWARD_FD2110_STREAM_AUDIO_PT` (`u8`) - тоже самое что и `V4L2_CID_FORWARD_FD2110_STREAM_PT` но для аудио в режиме complex
- `V4L2_CID_FORWARD_FD2110_STREAM_AUDIO_CHANNELS` (`u8`) - количество аудио каналов в режиме complex

Прием AES3id

Плата FD788 позволяет принимать поток AES3 через коаксиальный кабель (AES3id). Для этого не требуется никакой настройки - плата автоматически определяет поток AES3id. При этом V4L2-устройство перестает работать, и плата может быть использована как обычная звуковая карта с 8 входами.

Работа с FD2110

Плата FD2110 значительно отличается от всех остальных - в ней помимо видео интерфейсов имеется сетевой интерфейс. Сами видео-интерфейсы требуют дополнительной настройки для работы через сеть. В системе плата FD2110 предстает в виде:

- Двух стандартных сетевых карт, которые можно использовать как обычные сетевые карты
- Двух V4L2 устройств и двух ALSA устройств, представляющих собой SDI-входа/выхода (как в FD788)
- До 64 V4L2 устройств и 64 ALSA устройств, представляющие собой потоки ST2110-20 и ST2110-30

Сетевая карта

Плата представляет два независимых ethernet-интерфейса (обычно `enp*s*d1` и `enp*s*d2`). Работают они как полноценные сетевые интерфейсы - на них задаются настройки сети, скорости, настройки `sfp` и т.д., можно передавать данные. Единственное ограничение - у каждого интерфейса только одна очередь пакетов, поэтому производительность ограничена несколькими гигабитами (до 5). **Потоки видео 2110 в норме не попадают на эти интерфейсы**

V4L2 и ALSA

Плата предоставляет до 66 устройств V4L2 и ALSA. Как и в остальных платах, включение потоков и изменение направления SDI настраивается через переменную в `sysfs`

```
/sys/class/forward/fd2110-<номер>/io_config
```

Например, по умолчанию, `cat /sys/class/forward/fd2110-*/io_config` вернет строку длиной 66

```
II-----
```

Здесь каждый символ - один из входов/выходов или потоков, `-` - поток выключен, `I` - вход, `O` - выход. Потоки идут в следующем порядке:

1. 2 потока SDI, могут быть `I`, `O`, `-`
2. 16 входных потоков первого (самого дальнего от PCIe) сетевого интерфейса, могут быть только `I`, или `-`
3. 16 выходных потоков первого сетевого интерфейса, могут быть только `O`, или `-`
4. 16 входных потоков второго сетевого интерфейса, могут быть только `I`, или `-`
5. 16 выходных потоков второго сетевого интерфейса, могут быть только `O`, или `-`

При записи строки в `io_config` появятся соответствующие V4L2 устройства `/dev/videoX` и ALSA `hw:X,Y`. Например

```
echo III-----O----- >
/sys/class/forward/fd2110-*/io_config
```

Включит все SDI на вход, один входной и один выходной поток на первом сетевом интерфейсе, соответственно, в системе появятся устройства

- `/dev/video0`, `/dev/video1` - SDI видео-входа
- `/dev/video2` - ST2110-20 видео-вход на первом сетевом интерфейсе
- `/dev/video3` - ST2110-20 видео-выход на первом сетевом интерфейсе
- `"hw:0,0"`, `"hw:0,1"` - SDI аудио входа

- "hw:0,2" - ST2110-30 аудио-вход на первом сетевом интерфейсе
- "hw:0,18" - ST2110-30 аудио-выход на первом сетевом интерфейсе

Дальнейшая работа идёт как с обычными V4L2 и ALSA устройствами

Настройка платы

Сетевые настройки (скорость, IP, домен, DNS и т.д.) настраиваются стандартными средствами системы - systemd-networkd, NetworkManager и т.д.

Включение потоков (запись в /sys/class/forward/fd2110-*/io_config) - можно настраивать скриптом, софтом (fopen(), fwrite()) или libudev), или правилом udev

```
SUBSYSTEM=="forward", ATTR{software_id}=="<номер>", ATTR{io_config}="III-----
-----0-----"
```

Пример настройки через libudev - forward-v4l2-samples/common/Board.cpp, функция configureIOMode()

Настройка PTP

Для PTP используется стандартный клиент Linux - linuxptp. Пример запуска:

```
ptp4l -i enp1s0d1 -H -s -m
```

После этого плата будет синхронна к PTP.

Настройка входа V4L2/ALSA

Настройка потока для приема происходит через контролы V4L2 и ALSA. Через контролы задаётся:

- Формат видео (тайминг) - 1080i50, 1080p50 и т.д.
- Адрес (IP + Port) по которому будет идти видео поток.
- Формат цвета потока (8/10-bit, YCbCr/RGB). Пока поддерживается только YCbCr, 8 или 10 бит.
- Адрес (IP + Port) по которому будет идти аудио поток.

Пример установки параметров через API V4L2 и захвата видео приведен в forward-v4l2-samples/capture-fd2110/

Всё это можно сделать с помощью shell-скрипта, v4l2-ctl или qv4l2 :

```
# Включаем 1-й входной и 1-й выходной поток
echo III-----0----- >
/sys/class/forward/fd2110-*/io_config

# Установка 1080p50 на входе
v4l2-ctl -d /dev/video2 --set-dv-bt-timings index=13

# Установка адреса и формата стрима на входе
v4l2-ctl -d /dev/video2 -c stream_address='239.21.10.1:10000'
# YCbCr 4:2:2
v4l2-ctl -d /dev/video2 -c stream_color_mode=0
# 10-bit
v4l2-ctl -d /dev/video2 -c stream_bits_per_component=1

# Настройка ALSA
amixer -c 0 cset "iface=PCM,name='PCM Capture Stream
Address',device=2,subdevice=1"
```

```
"0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0xef,0x15,0x0a,0x01
```

```
# Разрешение входного мультикаста на сетевом интерфейсе  
ip a add 239.21.10.1 dev enp1s0d1 autojoin
```

После этого устройства начинают работать как обычные видео и аудио-устройства - можно использовать `gststreamer` или `qv4l2` для тестов

```
gst-launch-1.0 v4l2src device=/dev/video2 ! autovideosink
```

Настройка выхода V4L2/ALSA

Настройка потока для передачи происходит также через контролы V4L2 и ALSA. Через контролы задаётся:

- Формат видео (тайминг) - 1080i50, 1080p50 и т.д.
- Адрес (IP + Port) по которому будет идти видео поток.
- Формат цвета потока (8/10-bit, YCbCr/RGB). Пока поддерживается только YCbCr, 8 или 10 бит.
- Номер исходящего порта
- Payload Type для RTP (обычно 96)
- Адрес (IP + Port) по которому будет идти аудио поток.

Всё это можно сделать с помощью shell-скрипта, `v4l2-ctl` или `qv4l2` :

```
# Включаем 1-й входной и 1-й выходной поток  
echo III-----0----- >  
/sys/class/forward/fd2110-*/io_config  
  
# Установка 1080p50 на выходе  
v4l2-ctl -d /dev/video3 --set-dv-bt-timings index=13  
  
# Установка адреса и формата стрима на выходе  
v4l2-ctl -d /dev/video3 -c stream_address='239.21.10.2:10000'  
# YCbCr 4:2:2  
v4l2-ctl -d /dev/video3 -c stream_color_mode=0  
# 10-bit  
v4l2-ctl -d /dev/video3 -c stream_bits_per_component=1  
# Порт источника  
v4l2-ctl -d /dev/video3 -c stream_source_port=10000  
# Payload type  
v4l2-ctl -d /dev/video3 -c stream_payload_type=96  
  
# Настройка ALSA  
amixer -c 0 cset "iface=PCM,name='PCM Playback Stream Source  
Port',device=18,subdevice=0" 10001  
amixer -c 0 cset "iface=PCM,name='PCM Playback Stream  
Address',device=18,subdevice=0"  
"0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0xef,0x15,0x0a,0x01
```

После этого устройства начинают работать как обычные видео и аудио-устройства - можно использовать `gststreamer` или `qv4l2` для тестов

```
gst-launch-1.0 videotestsrc ! v4l2sink device=/dev/video3
```

Настройка Seamless Protection Switching (SPS, ST2022-7) на выходе

Для дублирования потока на выходе нужно настроить адреса потоков на обоих портах, а затем выставить контрол `st2022_7_tx_enable`

```
v4l2-ctl -d /dev/video3 -c st2022_7_tx_enable=1
```

После этого, при проигрывании на первом порту данные будут дублироваться на втором, при этом адреса на втором порту будут подменяться. Устройства на втором порту работать не будут.

Полный пример:

```
# Включаем первые входы и выхода на обоих портах
echo III-----0-----I-----0----- >
/sys/class/forward/fd2110-*/io_config
# Первый выход вещает на 239.21.10.2:10000 и 239.21.10.2:10001
v4l2-ctl -d /dev/video3 -c stream_address="239.21.10.2:10000"
amixer -c 1 cset "iface=PCM,name='PCM Playback Stream Source
Port',device=18,subdevice=0" 10001
amixer -c 1 cset "iface=PCM,name='PCM Playback Stream
Address',device=18,subdevice=0"
"0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0xef,0x15,0x0a,0x02
# Второй выход вещает на 239.21.10.4:10000 и 239.21.10.4:10001
v4l2-ctl -d /dev/video5 -c stream_address="239.21.10.4:10000"
amixer -c 1 cset "iface=PCM,name='PCM Playback Stream Source
Port',device=50,subdevice=0" 10001
amixer -c 1 cset "iface=PCM,name='PCM Playback Stream
Address',device=50,subdevice=0"
"0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0xef,0x15,0x0a,0x04
# Включаем SPS
v4l2-ctl -d /dev/video3 -c st2022_7_tx_enable=1
```

История изменений

Дата	Версия драйвера	Изменения
2026.01.22	1.7.7	Исправления опечаток и английская версия
2025.12.12	1.7.7	Примеры Raw-ASI
2025.12.05	1.7.6	Документация режимов ASI, Raw, Complex
2025.12.03	1.7.6	Первая версия