ЯДРО LINUX ДЛЯ 1892ВМ14Я. РУКОВОДСТВО ПРОГРАММИСТА

Версия
11.03.2022

ОГЛАВЛЕНИЕ

1. О ДОКУМЕНТЕ

Документ содержит описание основных подсистем и драйверов ядра Linux, реализованных для поддержки аппаратуры СнК 1892ВМ14Я и модулей на базе СнК.

Ядро Linux поддерживает модули следующих ревизий:

- 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.

Файлы DTS `*.dtsi`, `*.dts` расположены в дереве исходных кодов U-Boot `arch/arm/dts/*.dts*`. Пути до прочих файлов приведены относительно корня дерева исходных кодов Linux.

2. ПОДСИСТЕМА УПРАВЛЕНИЯ ТАКТОВЫМИ СИГНАЛАМИ

Управление тактовыми сигналами и частотами в ядре Linux реализовано с использованием Common Clock Framework.[1] Тактовые сигналы микросхемы описаны в виде дерева в файле `mcom02.dtsi`. Для управления тактовыми сигналами и частотами используются следующие драйверы, описанные в `drivers/clk/elvees/clk-mcom.c`:

- `mcom-clk-gate`;
- `mcom-clk-divider`;
- `mcom-clk-mux`;
- `mcom-clk-pll`;
- `mcom-cmctr`.

Для корректного управления тактовыми сигналами каждый драйвер устройства, входящий в состав ядра Linux, должен реализовывать:

1. При инициализации драйвера:
    1. Захват необходимого для устройства тактового сигнала, используя функцию `clk_get()`.
    2. Включение тактового сигнала, используя функцию `clk_enable()`.
2. При удалении драйвера:
    1. Выключение тактового сигнала, используя функцию `clk_disable()`.

При инициализации подсистемы управления тактовыми сигналами происходит начальная настройка всех PLL и делителей частот микросхемы. Устанавливаемые при инициализации значения множителей PLL и делителей частот описаны в файле `mcom02.dtsi`.

[1] https://www.kernel.org/doc/Documentation/clk.txt

3. ДРАЙВЕР КОНТРОЛЛЕРА SPI *DW_SPI_MMIO*

Драйвер *dw_spi_mmio* управляет контроллерами SPI0 и SPI1 СнК 1892ВМ14Я.

Драйвер поддерживает следующие возможности:

1. Работа в режиме ведущего устройства.
2. Использование GPIO в качестве сигналов CS.
3. Настройка работы контроллера: фазы (CPHA) и полярности (CPOL) сигнала SCK, полярности сигнала CS (при использовании GPIO в качестве CS).
4. Поддержка слов длиной 8 и 16 бит.
5. Управление скоростью передачи в зависимости от ведомого устройства.

Ограничения драйвера:

1. Не поддерживается передача данных с помощью DMA.
2. Не поддерживается передача слов данных младшим битом вперед.

4. ДРАЙВЕР КОНТРОЛЛЕРА PWM *PWM-MCOM*

Драйвер *pwm-mcom* управляет контроллером PWM 1892ВМ14Я. Драйвер реализует стандартный интерфейс PWM[2]

Файл с исходным кодом драйвера — `drivers/pwm/pwm-mcom.c`. Описание DTS bindings представлено в файле `Documentation/devicetree/bindings/pwm/elvees,mcom-pwm.txt`.

Ограничения драйвера:

1. Не реализовано управление каналами OUTB.
2. Не поддерживается режим счёта PWM-контроллера (PWM API не поддерживает данный режим).
3. Не реализовано управление предделителем.

[2] https://www.kernel.org/doc/Documentation/pwm.txt

5. ДРАЙВЕР КОНТРОЛЛЕРА ДИСПЛЕЯ *VPOUT-DRM*

Данный раздел применим к драйверу контроллера дисплея VPOUT СнК 1892ВМ14Я для подсистемы DRM — *vpout-drm*.

Документация, описывающая текущую версию подсистемы DRM, доступна по ссылке Linux GPU Driver Developer's Guide[3].

Исходный код драйвера содержится в директории `drivers/gpu/drm/vpout`.

На рисунке 5.1 представлена диаграмма зависимостей компонентов драйвера Direct Rendering Manager (DRM).

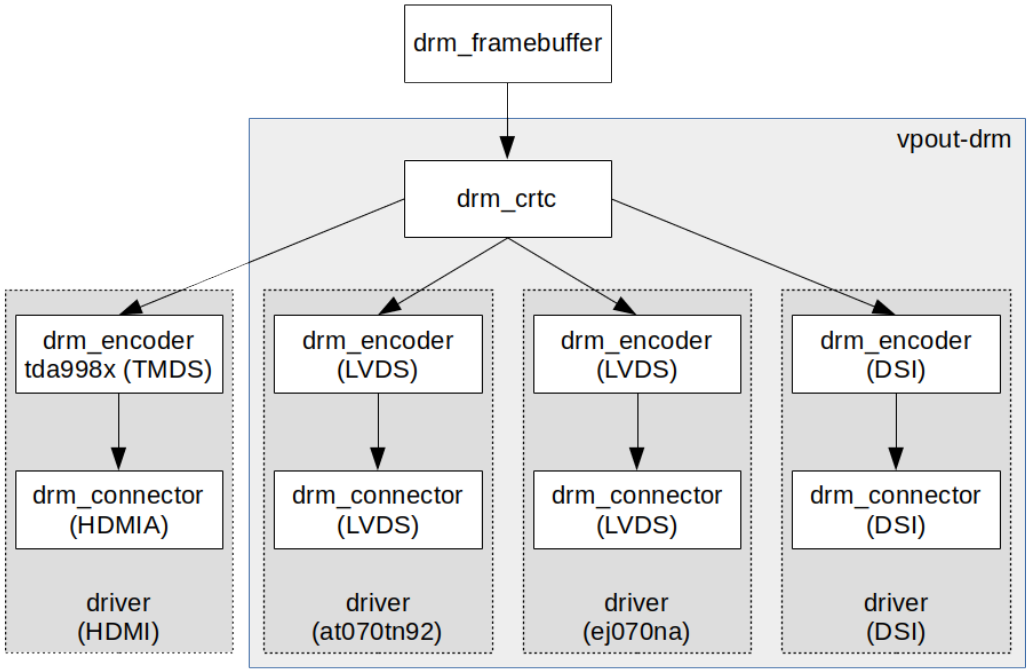

Рисунок 5.1. Диаграмма зависимостей компонентов драйвера DRM

Драйвер обеспечивает следующие возможности:

1. Разрешение экрана до 1920x1080 пикселей;
2. Поддержка внешнего HDMI передатчика NXP TDA998x;
3. Поддержка внешних панелей с заданием параметров дисплея через DTS;
4. Чтение Extended Display Identification Data (EDID);
5. Эмуляция фреймбуфера через устройство `/dev/fb0`.

Ограничения драйвера:

[3] https://www.kernel.org/doc/html/latest/gpu/index.html

1. Не поддерживаются чересстрочные видеорежимы (не поддерживаются контроллером дисплея VPOUT);
2. Не поддерживаются HDMI передатчики отличные от NXP TDA998x;
3. Не реализована поддержка абстракции плоскостей (plane abstraction);
4. Не поддерживается атомарная установка видеорежима.

Экспериментальные возможности драйвера, функционал не проверялся на аппаратуре:

1. Поддержка DSI-дисплеев:
    1. Не поддерживается MIPI Display Command Set (MIPI DCS) и Extended Display Identification Data (EDID) DSI-дисплеев, т.к. MIPI DCS не поддерживается в аппаратуре;
    2. Фиксированные параметры дисплея (разрешение, FPS) считываются из DTS. Дополнительно в командной строке ядра должно быть указано разрешение и тип DSI-дисплея;
    3. В коде драйвера фиксировано количество DSI-линий — 4.

При использовании в качестве устройства вывода HDMI монитора драйвер устанавливает оптимальный для подключенного монитора видеорежим, определяемый по EDID. С помощью параметров ядра (kernel parameters) возможно установить фиксированный видеорежим. Например, следующая строка задает разрешение экрана в 1280×720 пикселей:

```
video=HDMI-A-1:1280x720
```

При использовании в качестве устройства вывода DSI-дисплея драйвер устанавливает видеорежим заданный с помощью параметров ядра (kernel parameters). При этом параметры дисплея считываются из DTS. Например, следующая строка задает разрешение экрана в 1280×720 пикселей:

```
video=DSI:1280x720
```

Подробное описание параметров ядра, задающих видеорежим, содержится в документе `Documentation/fb/modedb.txt`.

6. ДРАЙВЕР ФРЕЙМБУФЕРА *VPOUTFB*

Для вывода на экран графического окружения на СнК используется подсистема FBDev[4] и драйвер *vpoutfb*. Директория с исходным кодом драйвера — `drivers/video/fbdev/vpoutfb`. Драйвер управляет контроллером VPOUT и HDMI-адаптером ITE IT66121.

Алгоритм работы драйвера:

1. Если в DTS в узле output присутствует свойство compatible=»ite,it66121», то выполнить настройку контроллера ITE IT66121, подключенного по I2C.
2. Считать из DTS видеорежим и настроить VPOUT для вывода в заданном видеорежиме.
3. Если в DTS отсутствует видеорежим или тайминги некорректны, или свойство output отсутствует, то настроить VPOUT для вывода в режиме 720p 60 FPS.

Вызов ioctl `FBIOPUT_VSCREENINFO` с заданием неподдерживаемого режима завершается с `-EINVAL`. (Следовательно, вызов `fbset` завершится с ненулевым кодом возврата).

Поддерживаются следующие ioctl:

- `FBIOGET_VSCREENINFO`;
- `FBIOPUT_VSCREENINFO`;
- `FBIOGET_FSCREENINFO`;
- `FBIOGETCMAP`;
- `FBIOPUTCMAP`;
- `FBIOBLANK`;
- `VPOUTFB_GET_MEMORY_ID`.

При появлении прерывания OUT_FIFO_INT блока VPOUT драйвер останавливает и переинициализирует VPOUT. При этом в dmesg печатается сообщение «Caught OUT_FIFO_INT, reinitializing VPOUT».

В драйвере не реализовано:

1. Чтение EDID HDMI-монитора и ограничение возможных разрешений согласно данным из EDID.
2. Остановка/запуск VPOUT при отключении/подключении HDMI-монитора.

Примечание: Т.к. автоматическое определение подключения HDMI-монитора отсутствует, драйвер может быть выключен по умолчанию. Загрузка драйвера выполняется командой `modprobe vpoutfb`.

---

[4] https://www.kernel.org/doc/Documentation/fb/api.txt

Примечание: Для управления видеорежимами может использоваться утилита `fbset` и файл `fb.modes`.

Драйвер считывает видеорежим из DTS в соответствии с описанием в `Documentation/devicetree/bindings/video/display-timing.txt`. В DTS-файле `mcom02.dtsi` описан формат цветовых компонентов изображения, устанавливаемые при инициализации драйвера. Подробное описание полей узла устройства VPOUT представлено в файле `Documentation/devicetree/bindings/fb/vpoutfb.txt`.

Примечание: Модуль *vpoutfb* используется консолью – перед выгрузкой модуля необходимо отключить консоль от драйвера:

```
echo 0 > /sys/class/vtconsole/vtcon1/bind
modprobe -r vpoutfb
```

7. ДРАЙВЕР VPU *AVICO*

7.1 Общие сведения

Драйвер *avico* управляет VPU VELcore-01 и реализует аппаратное кодирование и декодирование видео по стандарту H.264. Драйвер реализован с использованием подсистемы V4L2[5] и предоставляет стандартный программный интерфейс для кодирования/декодирования и управления.

7.1.1 Кодирование видео

1. Поддерживается входное несжатое видео с форматом пикселей M420[6].
2. Максимальная ширина кадра — 1920 пикселей.
3. Максимальная высота кадра — 4096 пикселей.
4. Поддержка кодирования видео с разрешением кратным 2 по ширине и высоте.
5. Возможность установки FPS видеопотока.
6. Возможность установки параметра QP с помощью контролов[7]:
   - V4L2_CID_MPEG_VIDEO_H264_I_FRAME_QP;
   - V4L2_CID_MPEG_VIDEO_H264_P_FRAME_QP;
   - V4L2_CID_MPEG_VIDEO_H264_CHROMA_QP_INDEX_OFFSET.
7. Возможность установки IDR-кадра с помощью контрола V4L2_CID_MPEG_VIDEO_FORCE_KEY_FRAME.
8. Возможность установки размера GOP с помощью контрола V4L2_CID_MPEG_VIDEO_GOP_SIZE. Новый размер GOP применяется со следующего IDR-кадра после завершения текущего GOP. Чтобы применить новый размер GOP на следующем кадре, нужно запросить IDR-кадр с помощью контрола V4L2_CID_MPEG_VIDEO_FORCE_KEY_FRAME.
9. Поддержка более одного потока кодирования видео. Максимальное число поддерживаемых потоков кодирования зависит от ширины и высоты кадра, количества запрошенных V4L2-буферов и объема памяти, которая может быть выделена с помощью Contiguous Memory Allocator (CMA). Теоретическое максимальное число поддерживаемых потоков кодирования можно вычислить с помощью формулы:

$$n = M/(W * (128 + (2 * B_c + 3/2 * (B_o + 1)) * H))$$

[5] https://linuxtv.org/downloads/v4l-dvb-apis-new/userspace-api/v4l/v4l2.html
[6] https://www.kernel.org/doc/html/v5.4/media/uapi/v4l/pixfmt-m420.html
[7] https://linuxtv.org/downloads/v4l-dvb-apis-new/userspace-api/v4l/ext-ctrls-codec.html

где M — размер памяти, которая может быть выделена с помощью CMA, W — ширина кадра, H — высота кадра, B_o — количество V4L2-буферов output-интерфейса[8], B_c — количество V4L2-буферов capture-интерфейса[9]. Дополнительно количество потоков n ограничивается фрагментацией CMA. Производительность кодирования каждого видео понижается с увеличением числа потоков, т.к. используется один аппаратный поток кодирования. Пример реального максимального числа потоков и достигаемой при этом производительности кодирования при QP 23, четырех V4L2-буферов output-интерфейса, четырех V4L2-буферов capture-интерфейса, размере CMA-памяти 128 МБ и частоте VPU 312 МГц:

- 1920x1072 — 4 потока, ~15 FPS;
- 1280x720 — 6 потоков, ~22 FPS;
- 640x480 — 19 потоков, ~18 FPS.

Пример ограничения числа потоков кодирования для обеспечения производительности кодирования ~30 FPS при QP 23, четырех V4L2-буферов output-интерфейса, четырех V4L2-буферов capture-интерфейса, размере CMA-памяти 128 МБ и частоте VPU 312 МГц:

- 1920x1072 — 2 потока, ~30 FPS;
- 1280x720 — 4 потока, ~30 FPS;
- 640x480 — 12 потоков, ~30 FPS.

10. Поддержка сжатия с постоянным битрейтом с помощью контролов:
    - V4L2_CID_MPEG_VIDEO_FRAME_RC_ENABLE — включение/выключение контроля битрейта;
    - V4L2_CID_MPEG_VIDEO_BITRATE — битрейт, бит/с.

    Контроль битрейта будет выключен, если число кадров в GOP меньше трёх.
11. Особенности очереди capture:
    - Поддерживаются буферы типов DMABUF и MMAP.
    - Буферы должны быть физически непрерывными.
    - Буферы типа MMAP выделяются драйвером с помощью CMA и являются физически непрерывными и некешируемыми.
12. Особенности очереди output:
    - Поддерживаются буферы типов DMABUF, MMAP и USERPTR.
    - Буферы типов DMABUF и MMAP должны быть физически непрерывными.
    - Буферы типа MMAP выделяются драйвером с помощью CMA и являются физически непрерывными и некешируемыми.

[8] https://linuxtv.org/downloads/v4l-dvb-apis-new/userspace-api/v4l/dev-output.html
[9] https://linuxtv.org/downloads/v4l-dvb-apis-new/userspace-api/v4l/dev-capture.html

- Если буферы типа USERPTR не являются физически непрерывными, выполняется копирование данных из них в непрерывную память с помощью DMA. Это снижает производительность кодирования.

7.1.2 Декодирование видео

1. Поддерживается выходное несжатое видео с форматом пикселей M420[10].
2. Поддержка декодирования видео с разрешением до 1920x1072 кратным 16 по ширине и высоте.
3. Поддержка контрола V4L2_CID_MPEG_VIDEO_GOP_SIZE.
4. Поддержка события V4L2_EVENT_SRC_CH_RESOLUTION[11] при смене разрешения видео (в т.ч. в начальный момент момент времени, когда разрешение видео неизвестно, см. Memory-to-Memory Stateful Video Decoder Interface[12]).
5. Особенности очередей capture и output:
   - Поддерживаются буферы типов DMABUF и MMAP.
   - Буферы должны быть физически непрерывными.
   - Буферы типа MMAP выделяются драйвером с помощью CMA и являются физически непрерывными и некешируемыми.

7.1.3 Ограничения драйвера

1. Поддерживается только один аппаратный поток кодирования и один аппаратный поток декодирования.
2. Несжатое видео имеет нестандартный формат пикселей (M420).
3. Шаг между яркостными и/или цветовыми строками должен быть кратен 16 байтам.
4. Для кодирования требуется 180 КиБ памяти XYRAM.
5. Использование с драйвером *delcore30m* невозможно, т.к. драйвер *avico* использует SDMA через API DMA-engine, драйвер *delcore30m* — непосредственное управление SDMA.
6. Поддерживается только контроль битрейта в режиме CBR. Контроль битрейта реализован на базе алгоритма, описанного в JVT-G012 и JVT-K049, только частично: при вычислении целевого параметра квантования для P-кадров не учитывается средняя абсолютная разница (MAD) кадра и его прогноза, т.к. VPU не предоставляет доступ к значению MAD, вычисляемому им на этапе оптимизации R-D. MAD может быть вычислен программно, но предполагается, что это повлечет значительное снижение производительности, поэтому вместо модели R-D, описанной в JVT-G012 и JVT-K049, используется модель R-D, описанная в *Контроль битрейта при кодировании*. Контроль битрейта возможен только при количестве кадров в GOP, равном 3 и более.

[10] https://www.kernel.org/doc/html/v5.4/media/uapi/v4l/pixfmt-m420.html
[11] https://www.kernel.org/doc/html/v5.4/media/uapi/v4l/vidioc-dqevent.html
[12] https://www.kernel.org/doc/html/v5.4/media/uapi/v4l/dev-decoder.html#memory-to-memory-stateful-video-decoder-interfac

7. Нет возможности менять FPS в процессе кодирования.
8. Для декодирования требуется выделенная память в двух контроллерах DDR (по 8 МиБ в каждом DDR для поддержки разрешения 1920x1072).
9. При декодировании сжатый кадр должен быть целиком расположен в одном output-буфере.

При кодировании для обхода проблемы rf#1382 драйвер использует промежуточные буферы в XYRAM для восстановленных и сжатых данных. Всего используется 4 буфера по 45 КиБ (строка макроблоков для кадра шириной 1920 пикселей в формате M420) — 2 буфера для восстановленных данных и 2 для сжатых. В результате реализации обхода проблемы, максимальная ширина кадров ограничилась 1920 пикселями.

При кодировании после каждой строки макроблоков VPU останавливается и драйвер выполняет следующие действия:

1. Настраивает VPU на другой промежуточный буфер.
2. Запускает SDMA для копирования данных из промежуточного буфера в DDR.
3. Запускает VPU на обработку следующей строки макроблоков.

При кодировании для обхода проблемы rf#2003 в обработчике прерывания используется задержка, состоящая из следующих действий:

1. Ожидание завершения чтения очередных данных исходного и референсного кадров.
2. Ожидание завершения 80-кратного чтения регистра EVENTS.
3. Ожидание снятия всех флагов регистра EVENTS, указывающих на текущую работу VDMA.

При декодировании для обхода проблемы зависания rf#1382 драйвер в начале декодирования выделяет буферы DDRx и DDRy в разных DDR, затем каждый кадр выполняет следующие действия:

1. Копирование входных сжатых данных из буфера пользователя в буфер DDRx с помощью SDMA. DDRx — буфер, в котором находится референсный кадр.
2. Копирование в VRAM первых двух строк макроблоков из буфера референсного кадра. Данное копирование стало необходимым после перемещения восстановленного кадра в отдельный буфер из общего буфера для референсного/восстановленного кадра. Восстановленный кадр теперь записывается в буфер DDRy.
3. Запуск декодирования, при этом референсный кадр и сжатые данные считываются из DDRx, а восстановленный кадр записывается в DDRy.
4. Выгрузка последней строки макроблоков из VRAM в буфер восстановленного кадра.
5. Копирование полученного в результате декодирования восстановленного кадра из DDRy в буфер пользователя с помощью SDMA.
6. Буферы DDRx и DDRy меняются местами, так что восстановленный кадр в DDRy используется как референсный кадр в DDRx для следующего кадра.

7.2 Использование драйвера

При инициализации аппаратного блока драйвер регистрирует два устройства V4L2:

- index = 0 — для кодирования,
- index = 1 — для декодирования.

В дистрибутиве Buildroot для устройств создаются символические ссылки в директории `/dev/v4l/by-path`:

```
# ls -l /dev/v4l/by-path/*codec*
lrwxrwxrwx ... /dev/v4l/by-path/platform-37100000.codec-video-index0 -> ../../
↪video0
lrwxrwxrwx ... /dev/v4l/by-path/platform-37100000.codec-video-index1 -> ../../
↪video1
```

7.3 Контроль битрейта при кодировании

Контроль битрейта основан на алгоритме, описанном в JVT-G012[13] и JVT-K049[14].

Битрейт контролируется на уровне GOP и на уровне кадра. В задачи контроля битрейта входит вычисление параметра квантования, так чтобы средний битрейт видео соответствовал заданному значению.

В последующем описании используются следующие обозначения:

- f — частота кадров (предполагается, что не меняется в процессе кодирования);
- w — ширина кадра в пикселях;
- h — высота кадра в пикселях;
- $i \in \mathbb{N}$ — номер GOP;
- $j \in \mathbb{N}$ — номер кадра в пределах GOP;
- N_i — количество кадров в i-ом GOP;
- $b_i(j)$ — размер j-го сжатого кадра в i-ом GOP;
- $B_i(j)$ — число бит, оставщихся в GOP, на момент кодирования j-го кадра в i-ом GOP;
- $QP_i(j)$ — параметр квантования j-го кадра в i-ом GOP;
- $\overline{QP}_i^p$ — усредненный параметр квантования по всем P-кадрам в i-ом GOP;
- $R_i(j)$ — целевой битрейт на момент кодирования j-го кадра в i-ом GOP;
- $S_i(j)$ — целевой уровень виртуального буфера (число оставшихся бит) после кодирования j-го карда в i-ом GOP;

[13] https://www.itu.int/wftp3/av-arch/jvt-site/2003_03_Pattaya/JVT-G012.zip
[14] https://www.itu.int/wftp3/av-arch/jvt-site/2004_03_Munich/JVT-K049.doc

- $V_i(j)$ — уровень заполнения виртуального буфера сжатых кадров (число бит в буфере сжатых кадров гипотетического референсного декодера) на момент кодирования j-го кадра в i-ом GOP;
- $bpp = \frac{R_1(1)}{fwh}$ — целевое число бит на пиксель.

7.3.1 Описание алгоритма

Общее число бит для оставшихся кадров в GOP в случае сжатия с константным битрейтом вычисляется по формуле:

$$B_i(j) = \begin{cases} \frac{R_i(j)}{f} \times N_i - V_i(j) & j = 1 \\ B_i(j-1) - b_i(j-1) & j = 2, 3, \ldots, N_i \end{cases}$$

Уровень заполнения виртуального буфера вычисляется по формуле:

$$V_i(1) = \begin{cases} 0 & i = 1 \\ V_{i-1}(Ni - 1) & \text{other} \end{cases}$$

$$V_i(j) = V_i(j-1) + b_i(j-1) - \frac{R_i(j-1)}{f} \quad j = 2, 3, \ldots, N_i$$

Для первого карда первого GOP значение параметра квантования выбирается на основе целевого числа бит на пиксель по формуле:

$$QP_1(1) = \begin{cases} 40 & bpp \leq l1 \\ 30 & l1 < bpp \leq l2 \\ 20 & l2 < bpp \leq l3 \\ 10 & bpp > l3 \end{cases}$$

Рекомендуемые значения:

- $l1 = 0.15, l2 = 0.45, l3 = 0.9$ — для кадров размером QCIF/CIF;
- $l1 = 0.6, l2 = 1.4, l3 = 2.4$ — для кадров больше QCIF/CIF.

Для последующих GOP:

$$QP_i(1) = max(QP_{i-1}(1) - 2, min(QP_{i-1}(1) + 2, \overline{QP}^p_{i-1} - min(2, \frac{N_{i-1}}{15})))$$

$$\overline{QP}^p_i = \sum_{j=2}^{N_i} QP_i(j)/(N_i - 1)$$

Полученное значение параметра квантования корректируется по формуле:

$$QP_i(1) = QP_i(1) - 1 \quad \text{if} \quad QP_i(1) > QP_{i-1}(N_{i-1}) - 2$$

Перед вычислением параметра квантования P-кадров вычисляется целевой размер P-кадра.

Целевой размер P-кадра зависит от целевого уровня виртуального буфера. Целевой уровень виртуального буфера определяется для каждого P-кадра согласно размерам первого IDR-кадра и первого P-кадра.

После кодирования первого P-кадра в i-ом GOP начальное значение целевого уровня виртуального буфера устанавливается по формуле:

$$S_i(2) = V_i(2)$$

Целевой уровень виртуального буфера для последующих P-кадров вычисляется по формуле:

$$S_i(j+1) = S_i(j) - \frac{S_i(2)}{N_i - 2}$$

Размер, выделяемый для j-го P-кадра в i-ом GOP, вычисляется на основе целевого уровня виртуального буфера, частоты кадров, целевого битрейта и реального уровня заполнения виртуального буфера по формуле:

$$\tilde{T}_i(j) = \frac{R_i(j)}{f} + \gamma(S_i(j) - V_i(j))$$

где γ — константа, равная 0.5.

При вычислении целевого размера кадра также следует учесть число оставшихся бит в GOP для кадра, которое вычисляется по формуле:

$$\hat{T}_i(j) = \frac{B_i(j)}{N_i - max(j-1, 1)}$$

Целевой размер P-кадра — взвешенная комбинация $\tilde{T}$ и $\hat{T}$, которая вычисляется по формуле:

$$T_i(j) = \beta\hat{T}_i(j) + (1-\beta)\tilde{T}_i(j)$$

где β — константа, равная 0.5.

Для совместимости с гипотетическим референсным декодером целевой размер P-кадра должен также ограничиваться согласно *Совместимость с гипотетическим референсным декодером*.

В JVT-K049 и JVT-G012 целевой параметр квантования вычисляется на основе квадратичной модели R-D, в которой используется средняя абсолютная разница[15] (MAD) текущего базового элемента (кадр, макроблок или слайс) и его прогноза, полученного на этапе оптимизации R-D. VPU не предоставляет значение MAD для кадра, поэтому вычисление целевого параметра квантования выполняется другим методом.

Согласно Rate-Distortion Analysis for H.264/AVC Video Coding and its Application to Rate Control[16] увеличение параметра квантования на единицу приводит к уменьшению битрейта на ~12.5%, таким образом целевой параметр квантования QP может быть вычислен из формулы:

$$T = c_1 \times 0.875^{(QP - QP_c)} T_c + c_2$$

[15] MAD для текущего базового элемента вычисляется на основе линейной модели прогнозирования, в которой используется действительный MAD предыдущего базового элемента.

[16] https://ieeexplore.ieee.org/document/1546001

где T — целевой размер кадра, T_c — размер кадра, полученный при значении QP_c параметра квантования, c_1, c_2 — коэффициенты, начальные значения которых соответственно равны 1 и 0. Коэффициенты обновляются после кодирования каждого кадра согласно методу наименьших квадратов.

Данный метод требует двух проходов кодирования каждого кадра, т.к. сначала требуется вычислить размер кадра для заданного QP и в случае необходимости скорректировать QP и повторно выполнить кодирование. Для выполнения одного прохода кодирования предлагается прогнозировать размер текущего кадра для заданного QP на основе размера предыдущего P-кадра по формуле[17]:

$$\tilde{T_c} = a_1 T_c + a_2$$

где a_1 и a_2 — коэффициенты модели прогнозирования размера кадра, T_c — размер предыдущего кадра. Начальные значения коэффициентов соответственно равны 1 и 0. Коэффициенты обновляются после кодирования каждого кадра согласно методу наименьших квадратов.

Для предотвращения резкого изменения качества видео вычисленный параметр квантования корректируется по формуле:

$$QP_i(j) = min(QP_i(j-1) + 2, max(QP_i(j-1) - 2, QP_i(j)))$$

Полученное значение параметра квантования ограничивается диапазоном [0; 51] и передается в VPU, который выполняет оптимизацию R-D согласно значению QP.

После завершения кодирования кадра значения коэффициентов a_1, a_2, c_1, c_2 обновляются.

Примечание - В текущей реализации алгоритма контроля битрейта коэффициенты a_1, a_2, c_1, c_2 всегда равны начальным значениям.

7.3.2 Совместимость с гипотетическим референсным декодером

Чтобы удовлетворять требованиям гипотетического референсного декодера целевой размер P-кадра также ограничивается нижней границей ($Z_i(j)$) и верхней границей ($U_i(j)$) по формуле:

$$T_i(j) = min(U_i(j), max(Z_i(j), T_i(j)))$$

Верхняя и нижняя границы инициализируются по формулам:

$$Z_i(1) = B_{i-1}(N_{i-1}) + \frac{R_i(1)}{f}$$
$$U_i(1) = B_{i-1}(N_{i-1}) + be(t_{r,1}(1)) \times \omega$$

где $B_{i-1}(N_{i-1})$ — число бит для оставшихся кадров в GOP с номером (i-1), $B_0(N_0) = 0$, ω — константа, равная 0.9, $t_{r,1}(1)$ — длительность удаления 1-го кадра из виртуального буфера, $be(t)$ — размер данных (в битах), эквивалентный длительности *t* с коэффициентом преобразования, равным битрейту.

[17] Формула аналогична модели прогнозирования MAD, описанной в JVT-G012 и JVT-K049.

Последующие значения $Z_i(j)$ и $U_i(j)$ вычисляются по формулам:

$$Z_i(j) = Z_i(j-1) + \frac{R_i(j)}{f} - b_i(j-1)$$
$$U_i(j) = U_i(j-1) + (\frac{R_i(j)}{f} - b_i(j-1)) \times \omega$$

Примечание - Формулы для $Z_i(j)$ и $U_i(j)$, описанные выше, представлены в Adaptive rate control for H.264[18]. Предполагается, что в JVT-K049 даны некорректные формулы $Z_i(j)$ и $U_i(j)$ (в формулах вместо размера предыдущего кадра используется размер текущего кадра, который не может быть известен на данном этапе). В документе JVT-G012 целевой размер P-кадра не ограничивается нижней и верхней границами, поэтому формулы $Z_i(j)$ и $U_i(j)$ там отсутствуют.

[18] https://www.sciencedirect.com/science/article/abs/pii/S104732030500060X

8. ДРАЙВЕР КОНТРОЛЛЕРА ETHERNET *ARASAN-GEMAC*

Драйвер *arasan-gemac* управляет контроллером Ethernet Arasan GEMAC. Драйвер реализует стандартный интерфейс network devices, описанный в `Documentation/networking/netdevices.txt`. Обработка RX-прерываний реализована с использованием интерфейса NAPI[19].

Директория с исходным кодом драйвера — `drivers/net/ethernet/arasan`.

Драйвер поддерживает выполнение следующих операций из пространства пользователя:

1. Установка скорости (10/100/1000 Мб/с);
2. Установка дуплекса (full/half);
3. Установка уровня сообщений драйвера;
4. Установка MAC-адреса;
5. Чтение регистров контроллера Ethernet Arasan GEMAC утилитой ethtool;
6. Перезапуск автосогласования;
7. Проверка физического подключения;
8. Установка MTU кадра в диапазоне 68 – 3500 байт;
9. Отключение фильтрации пакетов (Promiscuous mode);
10. Включение приема всех multicast-пакетов (IFF_ALLMULTI);
11. Поддерживается фильтрация unicast-пакетов по MAC-адресу и multicast-пакетов по hash-таблице.

Примечание: Включение promiscuous mode повышает нагрузку на CPU.

Драйвер не поддерживает:

1. Управление паузой;
2. Чтение и запись EEPROM;
3. Wake-on-Lan;
4. Управление объединением прерываний.

[19] https://wiki.linuxfoundation.org/networking/napi

9. ПОДСИСТЕМА УПРАВЛЕНИЯ ЭНЕРГОПОТРЕБЛЕНИЕМ

Подсистема управления энергопотреблением Linux определяет модели управления энергопотреблением (подробнее см. Device Power Management Basics[20]):

- System Sleep;
- Runtime Power Management.

9.1 Модель System Sleep

В модели System Sleep определены состояния сна (подробнее см. System Power Management Sleep States[21]):

- Suspend-To-Idle (s2idle, freeze);
- Standby, Power-On Suspend (shallow, standby);
- Suspend-to-RAM (deep);
- Suspend-to-disk (disk).

Поддерживаемые состояния сна:

- Suspend-To-Idle;
- Power-On Suspend.

Для энергосбережения в состоянии Power-On Suspend используются свойства драйверов:

- поддержка приостановки (suspend) контроллера СнК в драйвере;
- поддержка приостановки (suspend) контроллера внешнего интерфейса (приёмопередатчик CAN, Ethernet PHY, и т.д.) в драйвере;
- поддержка CPU Hotplug (подробнее см. главу *CPU Hotplug*).

Поддержка приостановки реализована в драйверах контроллеров СнК:

- *avico* (невозможен переход в Power-On Suspend во время сжатия);
- *delcore-30m*;
- *designware-i2c*;
- *designware-i2s*;
- *dw-apb-gpio*;
- *dw-apb-uart*;

[20] https://www.kernel.org/doc/html/latest/driver-api/pm/devices.html
[21] https://www.kernel.org/doc/Documentation/power/states.txt

- *dw-wdt*;
- *dwc2*;
- *sdhci-mcom02*.

Поддержка приостановки реализована в драйверах контроллеров внешних интерфейсов модулей на базе СнК:

- *bcm4329-fmac*;
- *mcp2515*;
- *ft313h*.

Поддержка приостановки не реализована в драйверах контроллеров СнК:

- *arasan-gemac*;
- *dw-apb-ssi*;
- *dw-apb-timer*;
- *mcom-pwm*;
- *mfbsp-i2s*;
- *nfc-v2p99*;
- *pl330*;
- *vinc*;
- *vpout-drm*;
- *vpoutfb*.

Поддержка пробуждения (wakeup) реализована в драйверах:

- *dw-apb-uart* (подробнее см. *Пример пробуждения по событию от UART*);
- *dw-apb-gpio*;
- *rtc-ds1307* (подробнее см. *Пример пробуждения по событию от RTC*).

9.1.1 Пример пробуждения по событию от UART

1. Установить UART0 в качестве источника пробуждения:

   ```
   echo enabled > /sys/devices/platform/38028000.serial/tty/ttyS0/power/wakeup
   ```

2. Перевести ОС в состояние сна:

   ```
   echo freeze > /sys/power/state # enter Suspend-To-Idle state
   ```

   или:

   ```
   echo standby > /sys/power/state # enter Power-On Suspend state
   ```

3. В терминале на ПЭВМ отправить любой символ в приёмник контроллера UART0.

9.1.2 Пример пробуждения по событию от RTC

1. Перевести ОС в состояние сна до указанного времени пробуждения:

```
rtcwake -s 3 -m freeze # enter Suspend-To-Idle state
```

или:

```
rtcwake -s 3 -m standby # enter Power-On Suspend state
```

9.2 Модель Runtime Power Management

Для поддержки динамического управления энергопотреблением реализованы:

- механизм CPU hotplug;
- драйвер управления частотой ядер CPU *cpufreq-dt*.

9.2.1 Механизм CPU hotplug

Механизм CPU hotplug[22] позволяет включать и выключать процессорные ядра, не перезагружая систему, что может использоваться:

- для отключения CPU1;
- для перехода системы в состояния сна.

Для выключения и включения процессорных ядер используются функции `cpu_down()` и `cpu_up()`, описанные в файле `kernel/cpu.c`.

Использование через sysfs:

1. Для отключения питания ядра CPU1 необходимо выполнить:

```
echo 0 > /sys/devices/system/cpu/cpu1/online
```

2. Для включения питания ядра CPU1 необходимо выполнить:

```
echo 1 > /sys/devices/system/cpu/cpu1/online
```

9.2.2 Драйвер управления частотой ядер CPU *cpufreq-dt*

Штатный драйвер *cpufreq-dt*, позволяет управлять тактовой частотой ядер CPU0 и CPU1 через подсистему CPUfreq[23].

Директория с исходным кодом драйвера — `drivers/cpufreq`. Список частот ядер CPU описан в DTS-файле `mcom02.dtsi`. Описание DTS bindings представлено в файле `Documentation/devicetree/bindings/cpufreq/cpufreq-dt.txt`.

Возможности драйвера:

[22] https://www.kernel.org/doc/Documentation/core-api/cpu_hotplug.rst
[23] https://www.kernel.org/doc/Documentation/cpu-freq/user-guide.txt

1. Регуляторы масштабирования тактовой частоты ядер CPU (CPUfreq governors):
    - `ondemand` (по-умолчанию) — устанавливает тактовую частоту в зависимости от нагрузки на ядрах CPU;
    - `conservative` — похож на `ondemand`, но более экономный (предпочтение отдаётся меньшим тактовым частотам);
    - `performance` — устанавливает тактовую частоту в максимальное значение;
    - `userspace` — позволяет устанавливать частоту из пространства пользователя.
2. Управление регуляторами и частотами через `sysfs`.

Ограничения драйвера:

1. Не поддерживается управление напряжением питания ядер CPU, т.к отсутствует поддержка в СнК.
2. Не поддерживается независимое управление частотой ядер CPU, т.к отсутствует поддержка в СнК.

Для установки тактовой частоты ядер из пространства пользователя необходимо:

1. Выбрать регулятор `userspace`:

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

2. Выбрать частоту из поддерживаемых. Список частот доступен в файле `/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies`.
3. Установить частоту. Значение частоты передаётся в кГц, например:

```
echo 312000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

10. ПОДСИСТЕМА UART В РЕЖИМЕ RS-485

Для управления полудуплексными приёмопередатчиками RS-485 используются ioctl TIOCMBIS/TIOCMBIC:

```
int rts_flag = TIOCM_RTS;
ioctl(fd, TIOCMBIS, &rts_flag); // set send mode
ioctl(fd, TIOCMBIC, &rts_flag); // set receive mode
```

Примечание: модули Салют-ЭЛ24ОМ1 имеют полудуплексный приёмопередатчик RS-485.

11. МОДУЛЬ *DMATESTCONTIG* ДЛЯ ТЕСТИРОВАНИЯ SDMA

Модуль *dmatestcontig* не входит в состав исходных кодов Linux. Описание относится к модулю *dmatestcontig* из коммита `4896f90a4cd335d91f5b9f2c1457f24baa7f98ce` в репозитории модуля. Модуль *dmatestcontig* основан на модуле dmatest[24] из коммита `9da2b1641a098eb9b00aacbfd7715efb6a503f7b` в репозитории ядра Linux.

Модуль *dmatestcontig* позволяет провести тестирование SDMA с передачей данных, измерением производительности и проверкой правильности переданных данных. Управление SDMA осуществляется через DMA Engine API[25]. Поддерживаемые типы передач: DMA_MEMCPY, DMA_XOR, DMA_PQ. Модуль поддерживает ряд параметров, которые позволяют задать размер буферов, количество повторений передач, максимальное число используемых каналов SDMA, количество потоков ядра, использующих канал SDMA, и др. Особенностью модуля *dmatestcontig* является то, что он позволяет выбрать типы передач, указывать специфические регионы памяти и включать частоты, которые, например, могут быть необходимы для обеспечения доступа к устройству памяти.

Модуль поддерживает все параметры, которые имеет модуль *dmatest*, а также добавляет параметры:

- `tests` — типы выполняемых передач данных (по умолчанию выбраны все поддерживаемые типы передач);
- `memregs` — базовые адреса зарезервированных регионов памяти в DTS, в которые каналы SDMA должны адресовать запросы чтения/записи (по умолчанию регион выбирается автоматически);
- `src` — регионы памяти из `memregs`, в которые каналы SDMA будут адресовать запросы чтения (по умолчанию первый регион, указанный в `memregs`);
- `dst` — регионы памяти из `memregs`, в которые каналы SDMA будут адресовать запросы записи (по умолчанию первый регион, указанный в `memregs`);
- `clocks` — наименования узлов устройств в DTS, описывающих частоты, которых нужно включить, например, для обеспечения доступа SDMA к устройству памяти.

Параметр `tests` содержит битовую маску типов передач. Значение 1 бита включает тип передачи и наоборот. Соответствие типов передач и номеров бит в параметре:

- бит 0 — DMA_MEMCPY;
- бит 1 — DMA_XOR;
- бит 2 — DMA_PQ.

[24] https://www.kernel.org/doc/Documentation/dmaengine/dmatest.txt
[25] https://www.kernel.org/doc/html/latest/driver-api/dmaengine/client.html

Резервирование регионов памяти в DTS выполняется согласно Reserved memory regions[26]. Пример резервирования регионов памяти для модулей Салют-ЭЛ24ПМ:

```
reserved-memory {
  #address-cells = <1>;
  #size-cells = <1>;
  ranges;

  dmatestc_ddr0_reserved: dmatestc_ddr0@0x7f800000 {
    no-map;
    reg = <0x7f800000 0x800000>;
  };
  dmatestc_ddr1_reserved: dmatestc_ddr1@0xdf800000 {
    no-map;
    reg = <0xdf800000 0x800000>;
  };
  dmatestc_xyram_reserved: dmatestc_xyram@0x3a400000 {
    no-map;
    reg = <0x3a400000 0x40000>;
  };
};
```

Элементы массивов `src` и `dst` ссылаются на регионы памяти по индексу региона в массиве `memregs`. Номер канала SDMA, с которым ассоциируется регион, определяется индексом элемента в `src`/`dst`.

Пример загрузки модуля с параметрами `tests`, `src` и `dst`:

```
modprobe dmatestcontig wait=1 run=1 noverify=1 tests=0x01 \
  max_channels=5 memregs=0x7f800000,0xdf800000,0x3a400000 \
  src=2,2,0,1,1 dst=0,0,2,2,1
modprobe -r dmatestcontig
```

В примере выше пять каналов SDMA адресуют запросы чтения/записи в регионы памяти следующим образом:

- Канал 0: чтение из региона с базовым адресом 0x3a400000, запись в регион с адресом 0x7f800000.
- Канал 1: чтение из региона с базовым адресом 0x3a400000, запись в регион с адресом 0x7f800000.
- Канал 2: чтение из региона с базовым адресом 0x7f800000, запись в регион с адресом 0x3a400000.
- Канал 3: чтение из региона с базовым адресом 0xdf800000, запись в регион с адресом 0x3a400000.
- Канал 4: чтение из региона с базовым адресом 0xdf800000, запись в регион с адресом 0xdf800000.

Пример загрузки модуля с параметром `clocks`:

[26] https://www.kernel.org/doc/Documentation/devicetree/bindings/reserved-memory/reserved-memory.txt

```
modprobe dmatestcontig run=1 noverify=1 memregs=0x7f800000,0x3a400000 \
  src=0 dst=1 clocks="dsp_aclk"
modprobe -r dmatestcontig
```

В примере выше включается частота, описанная в узле устройства «dsp_aclk» в DTS, необходимая для обеспечения доступа к XYRAM.