СВД
Встраиваемые
Системы

Разработка модулей подсистемы управления общими блоками микропроцессора (platform-control)

Данила Петров

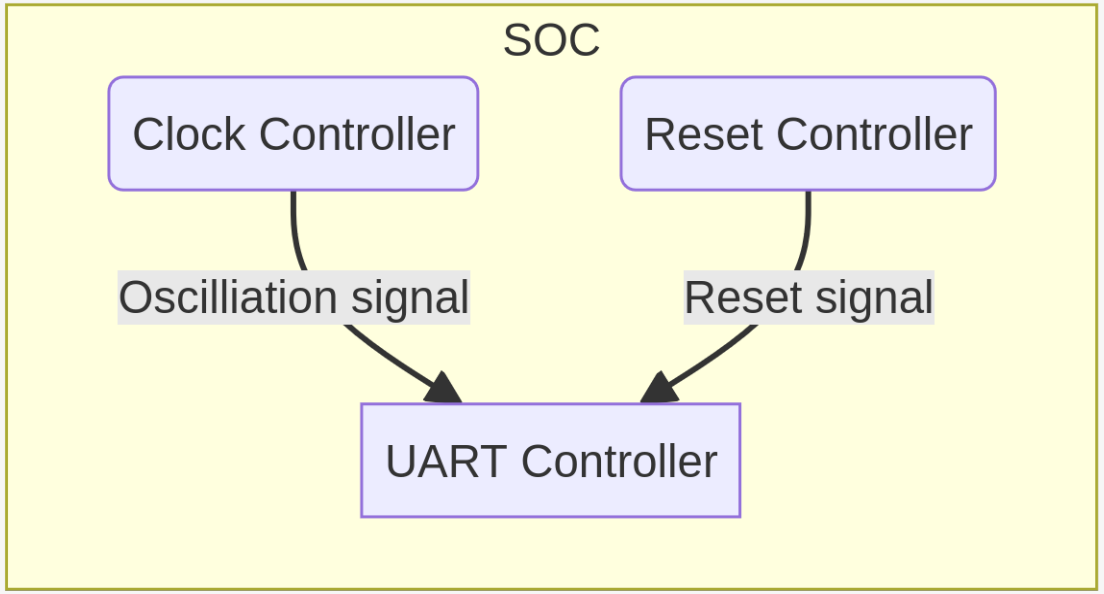
Инженер-программист
Отдел операционных систем

Повестка дня

- Общие блоки микропроцессоров (SoC)
- Проблема HAL и общих ресурсов
- Подсистема platform-control
- Разработка загружаемых драйверов devp-✿
- Применение на актуальных платформах

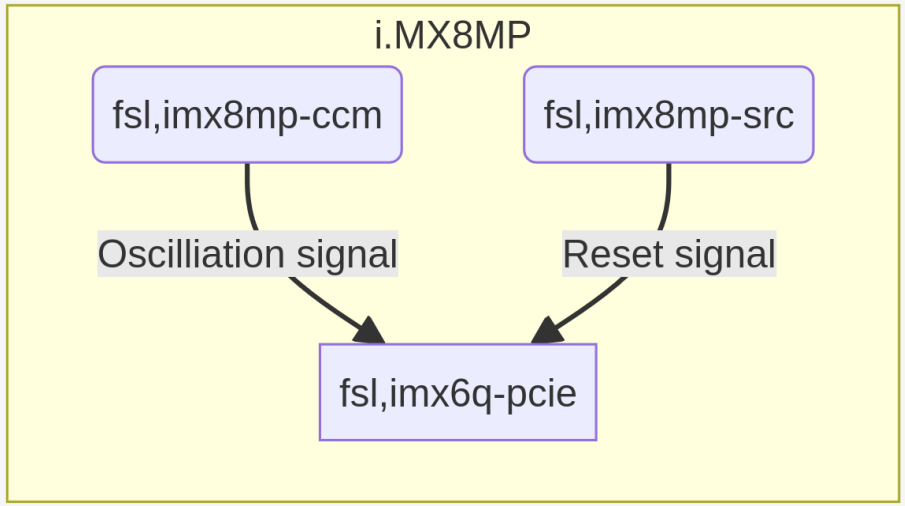
Общие блоки микропроцессоров (SoC)

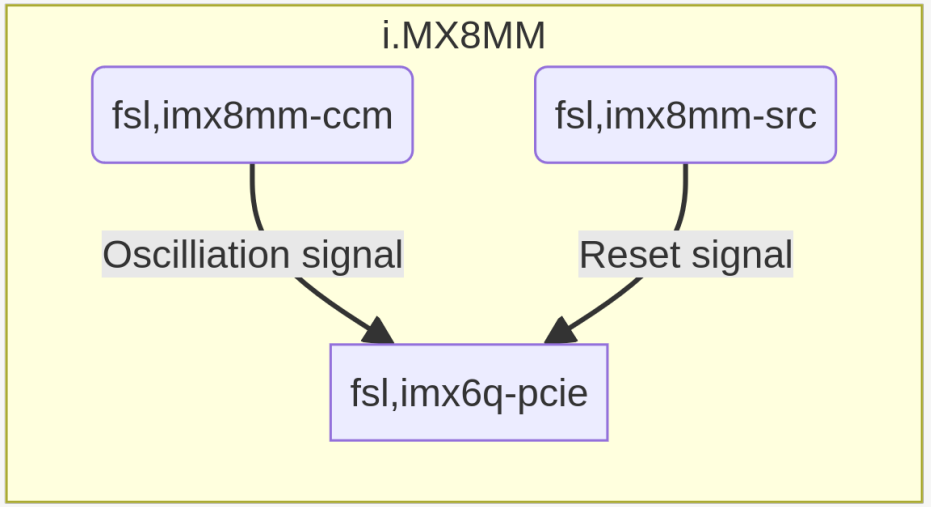
- **Ряд IP-блоков разделяют ресурсы другим IP-блокам**
  - Clock Controller — разделяет сигналы системного тактирования
  - Reset Controller — разделяет сигналы сброса
  - Контроллеры интерфейсов (UART, I2C, SPI …) - зависимые блоки



Проблема HAL и общих ресурсов

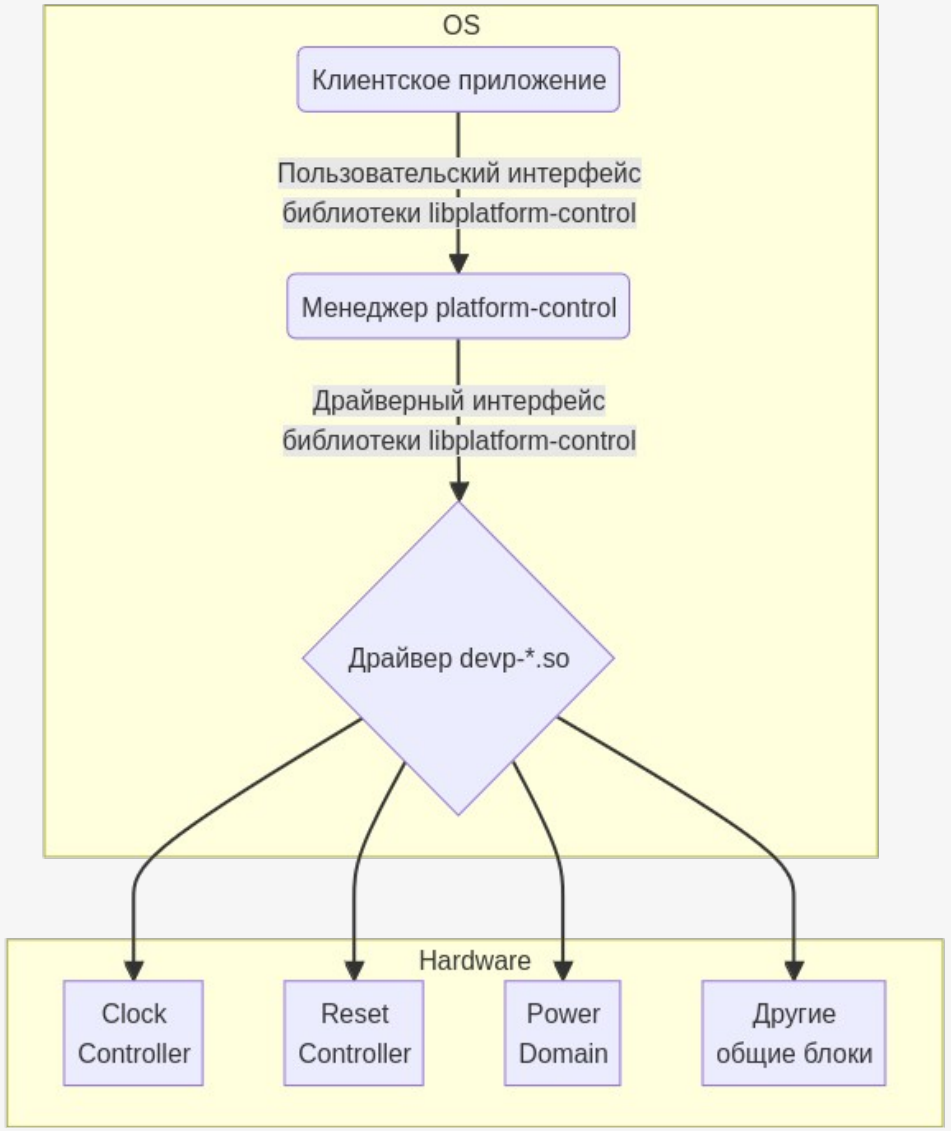
- Типовые контроллеры интерфейсов используются в разных SoC
- Общесистемные IP-блоки для каждого семейства SoC — специфичны
- Интерфейс не типизирован — напрашивается HAL (Hardware Abstraction Library)
- Одновременное обращение к ресурсам (регистрам) общих блоков со стороны клиентского приложения приводит к неопределенному состоянию





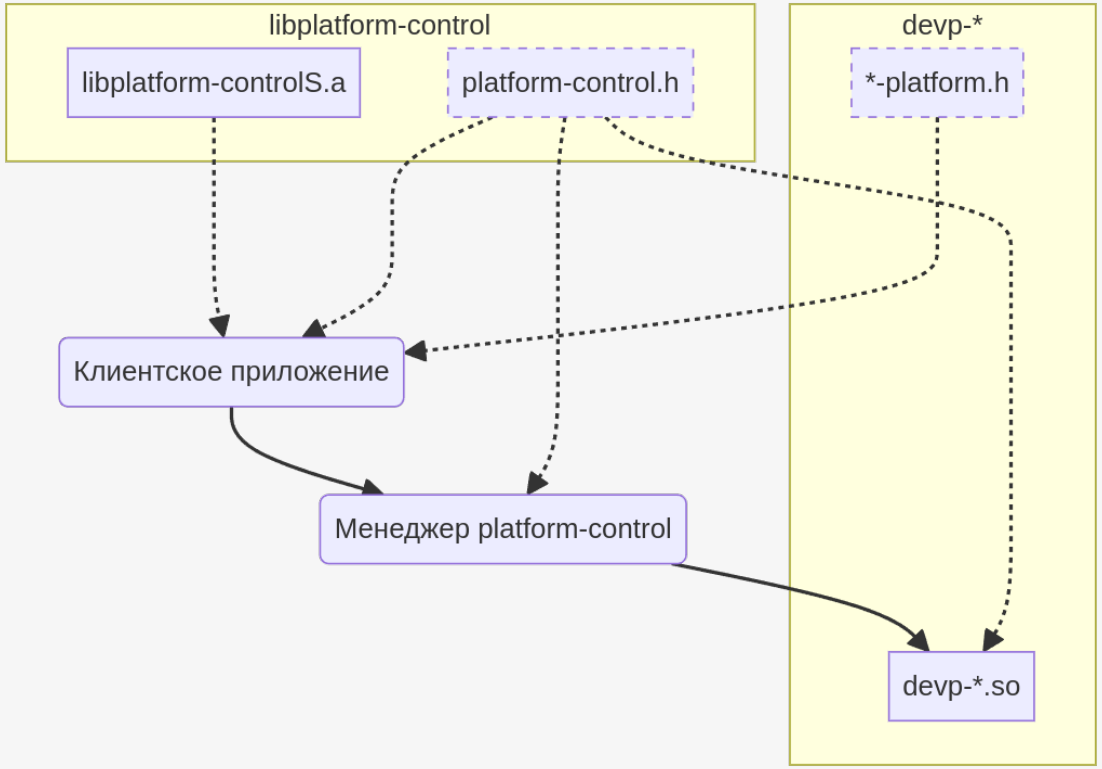
Подсистема platform-control: Архитектура

Имя компонента	Описание
Библиотека libplatform-control	Содержит в себе прикладные интерфейсы подсистемы, типы данных и макроопределения
Менеджер ресурсов platform-control	Менеджер, обрабатывающий запросы клиентских приложений и управляющий загружаемым драйвером
Драйверный интерфейс	Внутренний драйверный интерфейс динамически подгружаемых драйверов
Драйвер devp-*.so	Загружаемые драйвера, специфичные для конкретной платформы (SoC)



Подсистема platform-control: Компоненты

Имя компонента	Описание
libplatform-control(S).a	Скомпилированный архив библиотеки подсистемы, содержащий в себе реализацию клиентского API
platform-control.h	Заголовочный файл библиотеки, содержащий базовые макросы и объявления для работы с библиотекой подсистемы
devp-*.so	Скомпилированный платформозависимый драйвер, содержит низкоуровневую реализацию работы с регистрами управляемых аппаратных блоков, пресущую конкретной платформе (SoC)
***-platform.h**	Заголовочный файл, содержащий платформозависимые псевдоопределения управляемых аппаратных блоков



Подсистема platform-control: Пользовательское API

platform-control.h

```
int plat_ctrl_close(int fd);
int plat_ctrl_open(void);
int plat_ctrl_getdrvinfo(int fd, plat_ctrl_drvinfo_t *drvinfo);
int plat_ctrl_pd_attach(int fd, int id);
int plat_ctrl_pd_deattach(int fd, int id);
int plat_ctrl_reset_assert(int fd, int id);
int plat_ctrl_reset_deassert(int fd, int id);
int plat_ctrl_clk_enable(int fd, int id);
int plat_ctrl_clk_disable(int fd, int id);
int plat_ctrl_clk_set_rate(int fd, int id, uint32_t rate);
int plat_ctrl_pll_set_rate(int fd, int id, uint32_t rate);
int plat_ctrl_clk_set_parent(int fd, int id, uint32_t parent);
uint32_t plat_ctrl_clk_get_rate(int fd, int id);
uint32_t plat_ctrl_pll_get_rate(int fd, int id);
```

- Реализация вызовов определена в подключаемой статической библиотеке подсистемы **libplatform-control.a**
- Необходимость в использовании только со стороны клиентского приложения

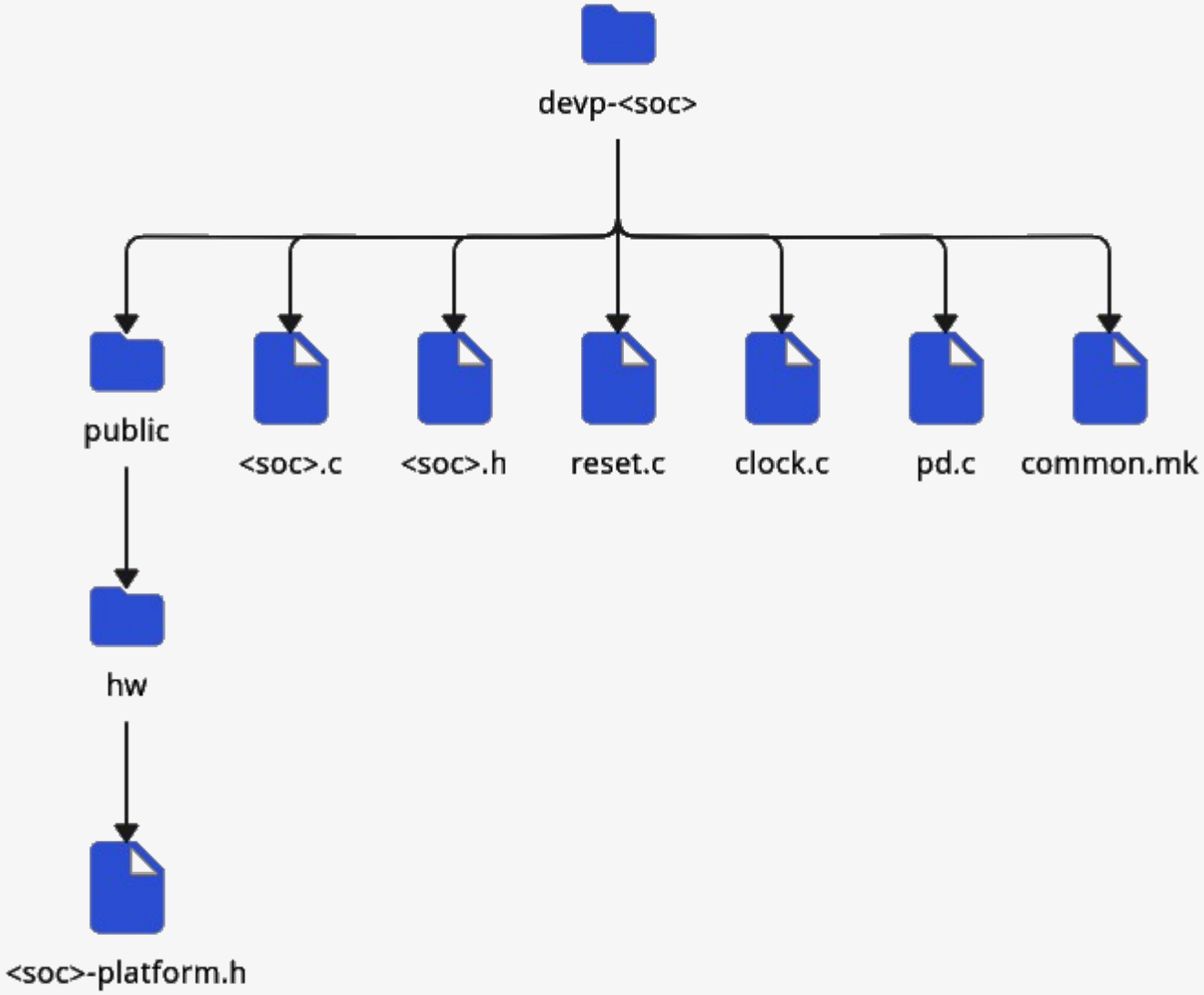
Подсистема platform-control: Драйверное API

platform-control.h

```
typedef  struct  {
  size_t  size;
  void*  (*init)(void *hdl, char *options);
  void   (*fini)(void *hdl);
  int    (*drvinfo)(void *hdl, plat_ctrl_drvinfo_t *info);
  int    (*set_pd)(void *hdl, plat_ctrl_pd_cfg_t *cfg);
  int    (*set_reset)(void *hdl, plat_ctrl_reset_cfg_t *cfg);
  int    (*set_clk)(void *hdl, plat_ctrl_clk_cfg_t *cfg);
  int    (*get_clk)(void *hdl, plat_ctrl_clk_cfg_t *cfg);
}  plat_ctrl_funcs_t;
```

- Поля структуры представлены в виде **callback**-функций
- Структура **plat_ctrl_funcs_t** определяет драйверный интерфейс
- Реализация вызовов пределена в драйвере **devp-*.so**

Разработка загружаемых драйверов devp-*: Дерево исходников



- Разбиение функционала управления на отдельные единицы трансляции сугубо опционально
- Линковка статической библиотеки подсистемы **libplatform-control.a** для сборки **не требуется**

Разработка загружаемых драйверов devp-*: Драйверное API

soc.c

```
#include <hw/platform-control.h>

plat_ctrl_funcs_t plat_ctrl_drv_entry = {
  sizeof(plat_ctrl_funcs_t),
  soc_init,          /* init() */
  soc_fini,          /* fini() */
  NULL,              /* drvinfo() */
  NULL,              /* set_pd() */
  soc_set_reset,     /* set_reset() */
  NULL,              /* set_clk() */
  NULL               /* get_clk() */
};
```

- Определение структуры интерфейса драйвера с именем **plat_ctrl_drv_entry** — обязательно
- Для получения доступа к функциям драйверного API менеджер **platform-control** использует системную функцию **dlsym()**

Разработка загружаемых драйверов devp-*: Инициализация

soc.c

```
void * soc_init(void *hdl, char *options)
{
    /* Выделение необходимых ресурсов */
    ...
    return (hdl);
}

void soc_fini(void *hdl)
{
    /* Освобождение выделенных в init() ресурсов */
    ...
    free(hdl);
}
```

- Функции init() и fini() вызываются автоматически, при запуске/завершении работы менеджера **platform-control**
- Вызов функции init() предусматривает выделение необходимых драйверу ресурсов (маппирование физических областей памяти управляемых контроллеров, с помощью функции mmap_device_io())
- Вызов функции fini() предусматривает освобождение выделенных ранее в init() ресурсов (размаппирование областей памяти управляемых контроллеров, используя munmap_device_io())

Разработка загружаемых драйверов devp-*: Управление

platform-control.h

```
typedef struct {
  uint32_t    id;
  uint32_t    state;
#define PLAT_CTRL_RST_STATE_DISABLED    (0 << 0)
#define PLAT_CTRL_RST_STATE_ENABLED     (1 << 0)
} plat_ctrl_reset_cfg_t;
```

reset.c

```
int soc_set_reset(void *hdl, plat_ctrl_reset_cfg_t *cfg)
{
    /* Взаимодействие с регистрами Reset контроллера */
    ...
    in32()/out32();
}
```

- Функция set_reset() вызывается менеджером при обработке пользовательских вызовов plat_ctrl_reset_assert() / plat_ctrl_reset_deassert()
- Настройка регистров контроллера осуществляется на основе конфигурации, переданной с помощью структуры **plat_ctrl_reset_cfg_t**
- Структуры конфигураций управляемых блоков определяются внутри функций пользовательского интерфейса

Разработка загружаемых драйверов devp-*: Клиент

common.mk

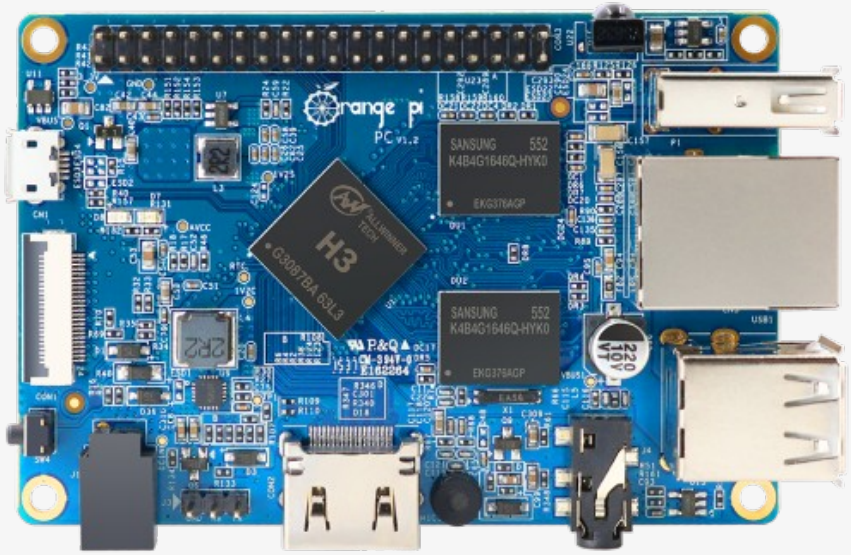
```
LIBS += platform-controlS
```

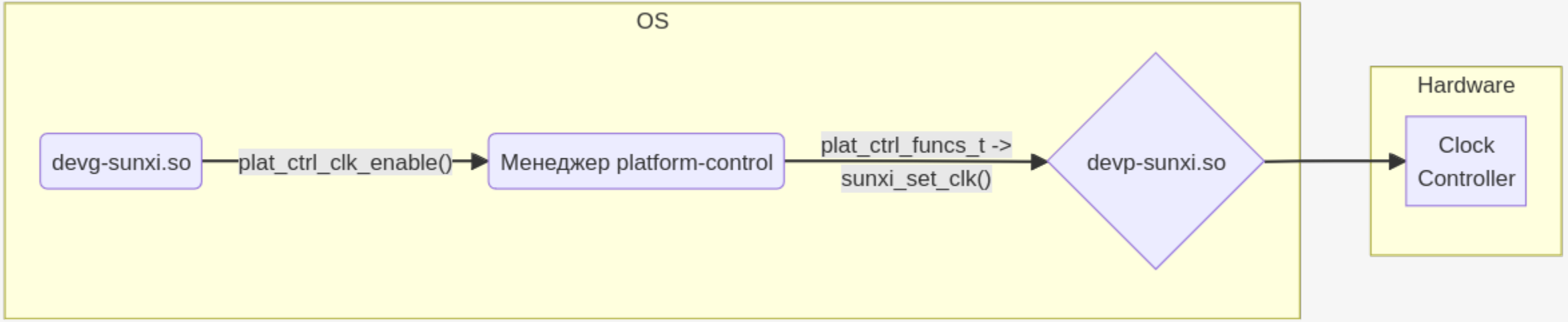
file.c

```
#include  <hw/platform-control.h>
#include  <hw/soc-platform.h>

/* Регистрируем файловый дескриптор `/dev/platform` */
int fd = plat_ctrl_open();

/* Включаем сигнал сброса необходимой линии согласно параметру `id` */
if(plat_ctrl_reset_assert(fd, id) != EOK)
{
    /* Сигнализирование о некорректной работе вызова */
}

/* Закрываем файловый дескриптор */
plat_ctrl_close(fd);
```

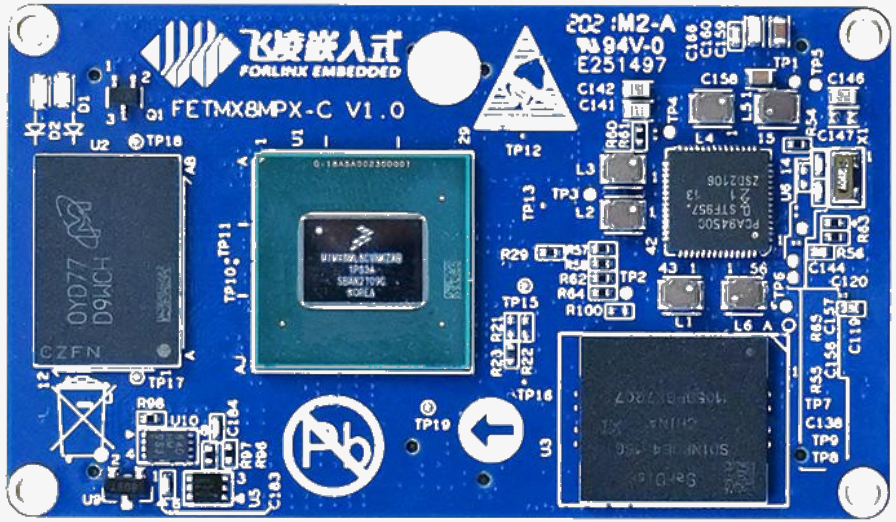
- Для доступа к функциям пользовательского интерфейса линкуем библиотеку **libplatform-control.a**
- Подключаем необходимые заголовочные файлы с объявлениями структур, функций подсистемы и псевдоопределениями управляемых блоков
- Взаимодействие с подсистемой происходит через файловый дескриптор **/dev/platform**

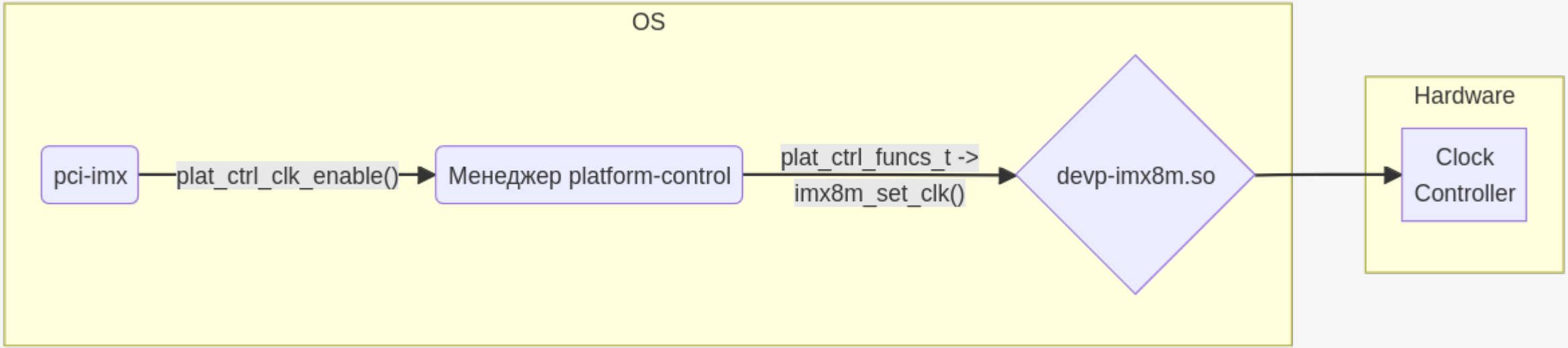
Применение на платформах: Allwinner H3



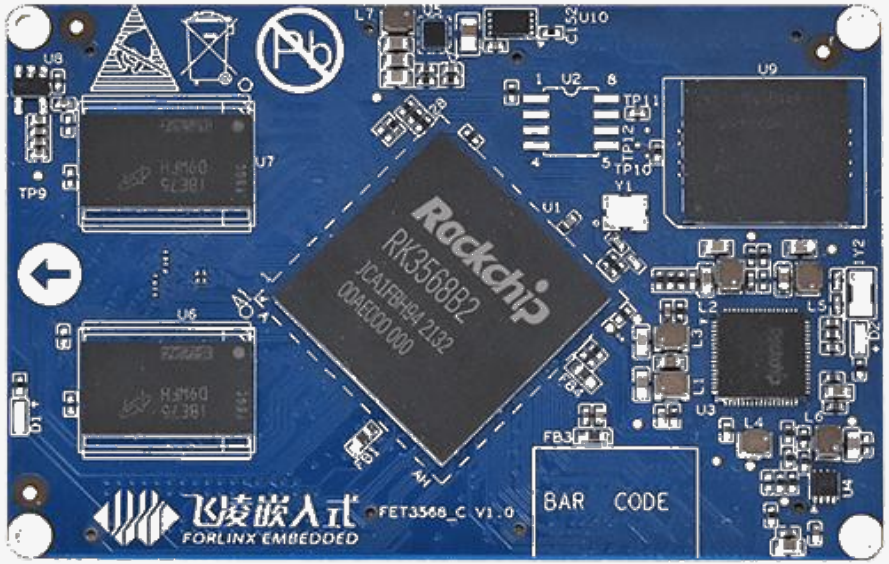


Применение на платформах: NXP i.MX8M



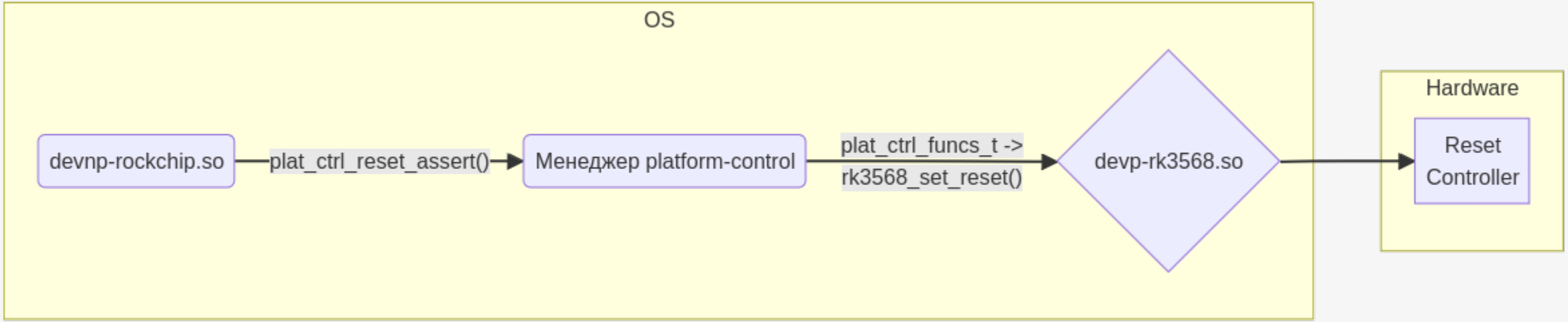


Применение на платформах: Rockchip RK3568



Rockchip

Rockchip RK3568 Quad Core



Материалы

- Публичный BSP для OrangePi PC
- Исходный код драйвера devp-sunxi

Ознакомительная статья на Хабр<e>

СВД
Встраиваемые
Системы

Спасибо за внимание!

Данила Петров

Инженер-программист
Отдел операционных систем

ул. Кузнецовская, д. 19,
г. Санкт-Петербург
+7 (812) 346-89-56
www.kpda.ru
support@kpda.ru