

# **ДРАЙВЕР КОНТРОЛЛЕРА SWIC ДЛЯ МИКРОСХЕМЫ 1892ВМ14Я. РУКОВОДСТВО ПРОГРАММИСТА**

**Версия v3.1**

**01.11.2019**

## ОГЛАВЛЕНИЕ

<b>1</b>	<b>О документе</b>	<b>3</b>
<b>2</b>	<b>Общее описание</b>	<b>4</b>
<b>3</b>	<b>Стандартные системные вызовы</b>	<b>5</b>
3.1	Функция open() . . . . .	5
3.2	Функция write() . . . . .	5
3.3	Функция read() . . . . .	5
<b>4</b>	<b>Описание IOCTL</b>	<b>6</b>
4.1	SWCIOС_SET_LINK . . . . .	6
4.2	SWCIOС_GET_LINK_STATE . . . . .	6
4.3	SWCIOС_GET_SPEED . . . . .	7
4.4	SWCIOС_SET_TX_SPEED . . . . .	7
4.5	SWCIOС_SET_MTU . . . . .	8

## **1. О ДОКУМЕНТЕ**

Документ содержит описание Linux-драйвера для контроллера SWIC микросхемы 1892ВМ14Я.

## 2. ОБЩЕЕ ОПИСАНИЕ

Драйвер *elvees-swic* предоставляет поддержку работы на канальном уровне SpaceWire:

1. Установка/разрыв соединения.
2. Получение статуса соединения.
3. Приём/передача пакетов SpaceWire.
4. Получение скорости приёма.
5. Установка/получение скорости передачи.
6. Установка размера пакета.

Драйвер реализует интерфейс символьного устройства Linux.

Ограничения драйвера:

1. Максимальный размер пакета (MTU) равен 1 МиБ (спецификация SpaceWire не регламентирует максимальный размер пакета).
2. Не поддерживается передача данных из/в пространство пользователя без копирования (zero-copy).
3. Не поддерживается передача коммуникационного пакета.
4. Не поддерживается работа с управляющими кодами.
5. Не поддерживается работа в дуплексном режиме.
6. Из-за ограничения `gf#7224` возможна некорректная работа драйвера если к контроллеру DDRMC1 подключено более 1 ГиБ памяти.

## 3. СТАНДАРТНЫЕ СИСТЕМНЫЕ ВЫЗОВЫ

### 3.1 Функция `open()`

Данная функция принимает в качестве аргумента имя устройства `/dev/spacewireX`, которое необходимо открыть и возвращает в пространство пользователя файловый дескриптор устройства.

### 3.2 Функция `write()`

При вызове функции `write()` драйвер выполняет:

1. Проверку на наличие соединения. Если соединение неактивно, вызов функции завершается с ошибкой `ENOLINK`.
2. Передачу входных пользовательских данных в канал SpaceWire. Если размер пользовательских данных больше установленного размера пакета (MTU), драйвер автоматически разбивает данные на порции размером не больше MTU. Каждая порция данных отправляется отдельным пакетом.

Если во время передачи данных произошел разрыв соединения, `write()` прекращает работу и возвращает количество байт, которое было передано на момент разрыва соединения.

Функция `write()` потокобезопасна.

**Предупреждение:** Из-за ограничения контроллера SWIC rf#11325 при завершении функции `write()` драйвер не гарантирует, что все пользовательские данные были отправлены. В результате при разрыве соединения после завершения `write()` возможна потеря данных.

### 3.3 Функция `read()`

При вызове функции `read()` драйвер выполняет:

1. Проверку на наличие соединения. Если соединение неактивно, вызов функции завершается с ошибкой `ENOLINK`.
2. Проверку размера пользовательского буфера. Если он меньше максимального размера пакета, функция завершается с ошибкой `ENOBUFS`.
3. Приём данных одного пакета из канала SpaceWire в пользовательский буфер.

Если во время приема данных произошел разрыв соединения, `read()` прекращает работу и возвращает количество байт, которое было принято на момент разрыва соединения.

Функция `read()` потокобезопасна.

## 4. ОПИСАНИЕ IOCTL

Перед вызовом функций `ioctl()` необходимо получить файловый дескриптор устройства, см. *Функция `open()`*.

### 4.1 SWCIOC\_SET\_LINK

Установка/сброс соединения:

```
int ioctl(int fd, SWCIOC_SET_LINK, unsigned int link_status);
```

Данный вызов применим к файловому дескриптору устройства.

Аргументы:

- `link_status` — тип выполняемой операции: 0 — сброс соединения, 1 — установка соединения.

При сбросе соединения драйвер выполняет:

1. Сброс link-интерфейса контроллера.
2. Остановку RX DATA DMA.
3. Очистку всех FIFO.

При установке соединения драйвер выполняет:

1. Сброс соединения.
2. Включение PLL TX SWIC, LVDS SWIC.
3. Установку коэффициента для подсчёта таймаутов установки соединения.
4. Запуск TX DATA DMA в режиме самоинициализации.

Данный вызов возвращает управление, не дожидаясь установки соединения. Для чтения статуса соединения необходимо вызвать `SWCIOC_GET_LINK_STATE`.

### 4.2 SWCIOC\_GET\_LINK\_STATE

Получение статуса соединения:

```
enum swic_link_state {  
    LINK_ERROR_RESET,  
    LINK_ERROR_WAIT,  
    LINK_READY,  
    LINK_STARTED,  
    LINK_CONNECTING,  
    LINK_RUN  
};
```

```
int ioctl(fd, SWCIOCG_GET_LINK_STATE, enum swic_link_state *link_status);
```

Данный вызов применим к файловому дескриптору устройства.

Статусы соединения:

- LINK\_ERROR\_RESET — соединение отсутствует.
- LINK\_ERROR\_WAIT — ошибка ожидания соединения.
- LINK\_READY — узлы готовы к соединению.
- LINK\_STARTED — установка соединения началась.
- LINK\_CONNECTING — соединение устанавливается.
- LINK\_RUN — соединение установлено.

При вызове драйвер возвращает в пространство пользователя один из статусов соединения.

### 4.3 SWCIOCG\_GET\_SPEED

Получение скорости приёма и передачи:

```
struct elvees_swic_speed {
    unsigned int rx;
    unsigned int tx;
};

int ioctl(fd, SWCIOCG_GET_SPEED, struct elvees_swic_speed *speed);
```

Данный вызов применим к файловому дескриптору устройства.

При вызове драйвер возвращает в пространство пользователя значения скоростей (в Кбит/с) приёма и передачи.

### 4.4 SWCIOCG\_SET\_TX\_SPEED

Установка скорости соединения:

```
enum swic_tx_speed {
    TX_SPEED_2P4,
    TX_SPEED_4P8,
    TX_SPEED_72,
    TX_SPEED_120,
    TX_SPEED_168,
    TX_SPEED_216,
    TX_SPEED_264,
    TX_SPEED_312,
    TX_SPEED_360,
    TX_SPEED_408
};
```

```
};  
  
int ioctl(fd, SWCIOС_SET_TX_SPEED, enum swic_tx_speed tx_speed);
```

Данный вызов применим к файловому дескриптору устройства.

Возможные скорости передачи:

- TX\_SPEED\_2P4 — 2.4 Мбит/с,
- TX\_SPEED\_4P8 — 4.8 Мбит/с,
- TX\_SPEED\_72 — 72 Мбит/с,
- TX\_SPEED\_120 — 120 Мбит/с,
- TX\_SPEED\_168 — 168 Мбит/с,
- TX\_SPEED\_216 — 216 Мбит/с,
- TX\_SPEED\_264 — 264 Мбит/с,
- TX\_SPEED\_312 — 312 Мбит/с,
- TX\_SPEED\_360 — 360 Мбит/с,
- TX\_SPEED\_408 — 408 Мбит/с.

При вызове драйвер устанавливает скорость передачи, равную значению входного аргумента `tx_speed`.

Если значение входного аргумента `ioctl()` не входит в вышеперечисленный набор возможных скоростей, драйвер возвращает ошибку `EINVAL`.

## 4.5 SWCIOС\_SET\_MTU

Установка размера пакета:

```
int ioctl(fd, SWCIOС_SET_MTU, unsigned long mtu);
```

Данный вызов применим к файловому дескриптору устройства.

При вызове драйвер устанавливает размер пакета, равный значению входного аргумента `mtu`.

Если значение входного аргумента `ioctl()` превышает определённый в драйвере максимальный размер пакета, драйвер возвращает ошибку `EINVAL`.