



ЗАО НПЦ
МИКРОПРОЦЕССОРНЫЕ
ТЕХНОЛОГИИ



Мост Ethernet-SpaceWire

Руководство пользователя

Редакция 2.0.2

Санкт-Петербург
2019

Содержание

Содержание.....	2
Список изменений.....	4
1 Введение	5
2 Назначение и функциональные возможности моста Ethernet-SpaceWire	5
2.1 Функциональные параметры.....	6
2.2 Структурная схема моста.....	7
2.3 Функции моста	8
3 Требования к компьютеру	14
4 Требования к устройствам SpaceWire	14
5 Требования к питанию	14
6 Подключение и работа с мостом	14
6.1 Общие правила подключения моста к ПК и SpaceWire-устройствам.....	14
6.2 Установка SDK моста Ethernet-SpaceWire	15
6.2.1 Установка SDK на компьютер с Linux	16
6.2.2 Установка SDK на компьютер с Windows.....	17
6.3 Архитектура SDK в ОС.....	19
6.3.1 Архитектура библиотеки SDK в Linux	19
6.3.2 Архитектура библиотеки SDK в Windows.....	19
6.4 Инициализация моста	20
6.5 Подключение к мосту устройств с интерфейсом SpaceWire.....	23
6.6 Настройка моста	24
6.6.1 Настройки моста по умолчанию.....	25
6.6.2 Приложение configure_bridge.....	25
6.6.3 Настройка моста через функции API.....	26
6.6.3.1 Настройка всех параметров моста функцией SpW_Send_Conf_Packet.....	26
6.6.3.2 Настройка каждого параметров моста функциями API.....	29
6.7 Основная работа с мостом.....	32
6.7.1 Приложение bridge_app	32
6.7.2 Приложение example_receive_packets	34
6.7.3 Отправка данных через функции API	35
6.7.4 Прием данных через функции API.....	36
6.7.5 Типы Ethernet-фреймов и функции-обработчики	39
6.7.6 Взаимодействие функций-обработчиков и функций приема данных	42
6.7.7 Фрейм с кодами ошибок	42

6.7.8	Прием и передача управляющих кодов.....	45
6.8	Работа с несколькими мостами Ethernet-SpaceWire через коммутатор Ethernet.....	45
7	Завершение и перезагрузка работы с мостом	46
8	Приложения для работы с мостом	46
8.1	Сборка приложений для работы с мостом	47
8.2	Отображение принимаемых пакетов в приложениях	48
9	Создание пользовательских приложений для работы с мостом.....	52

Список изменений

Версия	Изменение	Дата
2.0.2	1. Разделы 6.2. и 6.7.8. Добавлена работа с управляющими кодами неопределенными стандартом.	29.11.2019
	2. Раздел 6.6.3.1. Изменен пример использования функции SpW_Send_Conf_Packet для настройки моста.	

1 Введение

Данный документ описывает мост Ethernet-SpaceWire, процесс его подключения, настройки и эксплуатации.

2 Назначение и функциональные возможности моста Ethernet-SpaceWire

Мост Ethernet-SpaceWire предназначен для подключения к сети SpaceWire (SpW) вычислительных машин и комплексов, имеющих интерфейс сети Gigabit Ethernet. Позволяет передавать и принимать данные и управляющие коды между сетью Ethernet и сетью SpaceWire.

Функции моста:

- Передача и прием данных и управляющих кодов между сетью Ethernet и сетью SpaceWire
- Транзитная передача данных из порта Spacewire в порт SpaceWire моста
- Фильтрация пакетов, приходящих из сети Spacewire
- Управление режимами работы, установка скоростей канала SpaceWire, настройка логических адресов абонента в сети SpaceWire
- Настройки локального и удаленного MAC-адреса сети Ethernet

Типовые применения:

- Управление, конфигурация и отладка сети SpaceWire путем подключения персонального компьютера к сети SpaceWire по интерфейсу Ethernet
- Организация каналов передачи и приема информационных потоков / пакетов SpaceWire между сетью SpaceWire и ЭВМ / ВК в наземных комплексах тестирования и испытаний

Пользователям предоставляется программное обеспечение для настройки моста, чтения состояния моста и приема/передачи пакетов данных и управляющих кодов между компьютером и мостом.

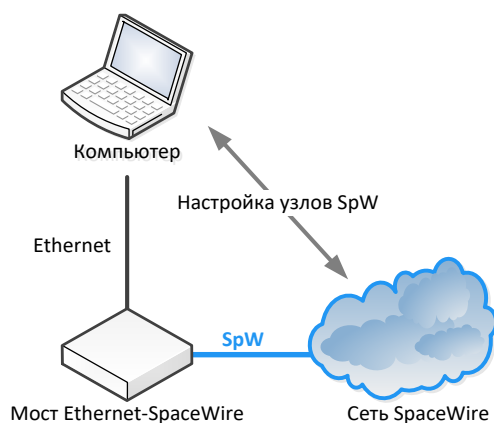


Рисунок 1 – Схема применения моста Ethernet-SpaceWire для настройки сети SpW

2.1 Функциональные параметры

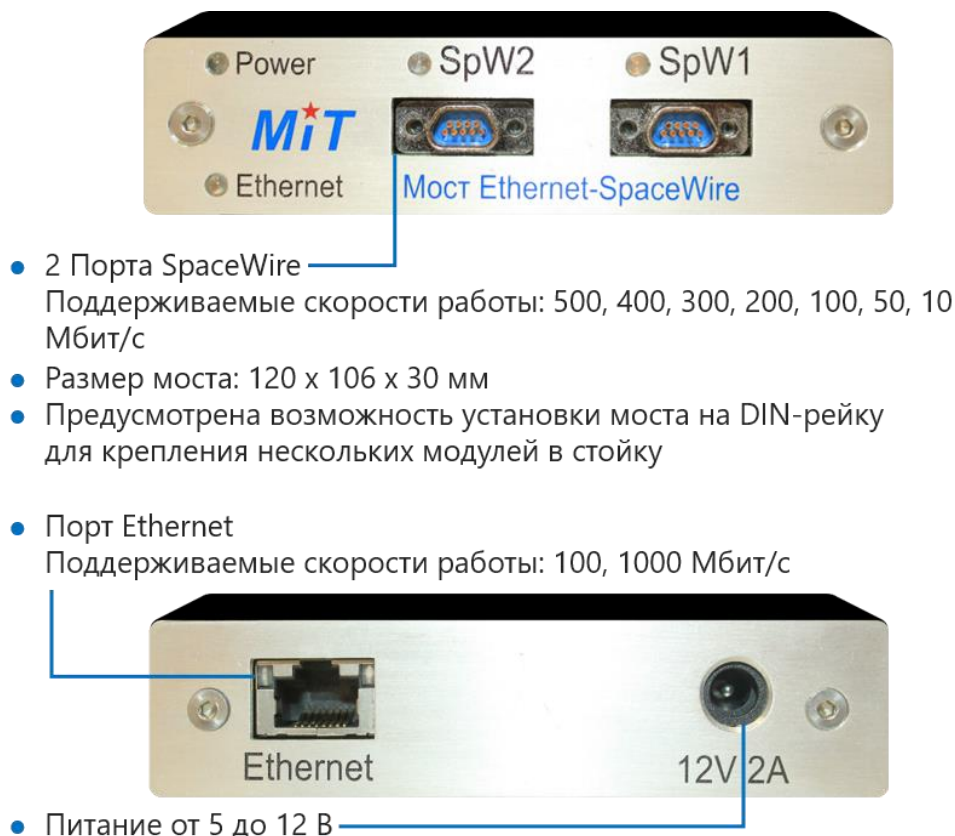


Рисунок 2 – Описание функциональных параметров моста Ethernet-SpaceWire

Мост оснащен четырьмя светодиодами, которые отображают статус питания, Ethernet-соединения и двух SpaceWire-соединений.



Рисунок 3 – Описание световой индикации моста Ethernet-SpaceWire

2.2 Структурная схема моста

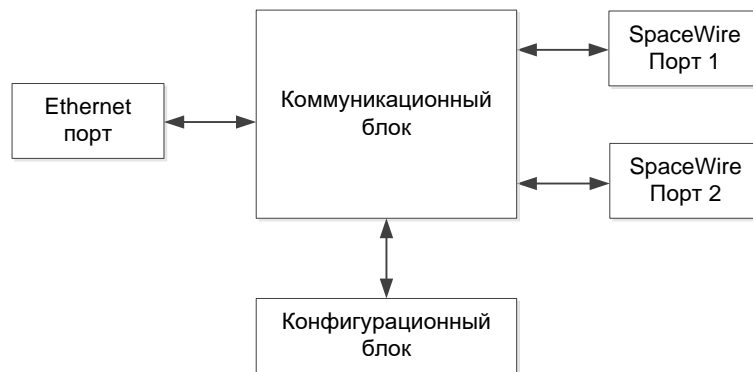


Рисунок 4 – Структурная схема моста Ethernet-SpaceWire

Мост имеет два порта **SpW**, полностью совместимых со стандартом SpaceWire ECSS-E-50-12C и его расширением – механизмом распределенных прерываний – в соответствии с проектами российского стандарта SpaceWire-RUS и европейского стандарта SpaceWire ECSS-E-50-12C rev.1

Порты **SpW** поддерживают скорость работы до 500 Мбит/с. Канал SpW является дуплексным. Предусмотрена возможность программной настройки скорости передачи данных.

Ethernet порт поддерживает скорость работы 100, 1000 Мбит/с. Предусмотрена система кредитования, информирование о потере фреймов Ethernet, изменение MAC-адреса и фильтрация фреймов по MAC-адресу.

SpW-порты соединены с Ethernet-портом через коммуникационный блок.

Коммуникационный блок настраивается через конфигурационный блок путем передачи конфигурационного фрейма в Ethernet порт: коммуникационный блок идентифицирует по заголовку конфигурационный фрейм, передает его в конфигурационный блок, который настраивает коммуникационный блок согласно заданным параметрам (см. [раздел 2.3](#)).

Исходящий трафик Ethernet-порта моста может быть направлен в: порт SpW1, порт SpW2, конфигурационный блок, а также в оба порта SpW одновременно.

Исходящий трафик может содержать: конфигурационные фреймы для настройки моста, SpW-пакеты с управляющими кодами, SpW-пакеты с данными.

Входящий трафик поступает в мост через порты SpW (управляющие коды и данные) и через конфигурационный блок (служебные фреймы: отчеты об ошибках работы, состояние моста).

Транзитный трафик проходит из одного порта SpW в другой порт SpW через коммуникационный блок. Предусмотрены режимы для анализа транзитного трафика, при котором весь трафик (или его часть) будет дублироваться в порт Ethernet.

2.3 Функции моста

Мост Ethernet-SpaceWire предусматривает (несколько) режимов работы с данными и (несколько) режимов работы с управляющими кодами.

Три режима передачи SpW-пакетов данных:

1. Режим прямой передачи
2. Коммутационный режим
3. Транзитный режим

В **режиме прямой передачи** пакет SpW передается из Ethernet-порта в один из портов SpW согласно заданной настройке Моста. Используется следующий формат Ethernet-фрейма, Рисунок 5:

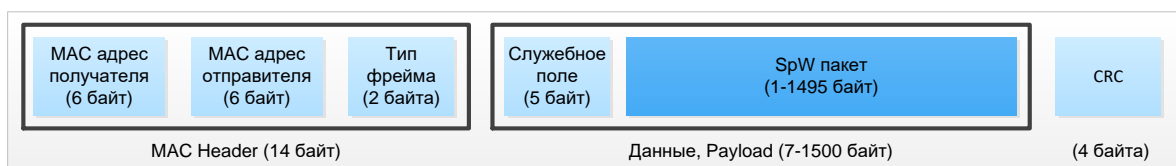


Рисунок 5 – Формат Ethernet-фрейма SpW-пакета при работе с данными

В **коммутационном режиме** в служебном поле перед SpW-пакетом включается дополнительный *коммутационный байт* (Рисунок 6), который при передаче данных из порта Ethernet в порты SpW определяет, в какой из двух портов SpW будет отправлен пакет SpW; при приеме пакетов данных из портов SpW показывает, с какого SpW-порта пришел пакет.



Рисунок 6 – Формат начального Ethernet фрейма SpW пакета при работе с данными в режиме 1

В **коммутационном режиме** пакеты передаются в порт SpW1 или порт SpW2, либо в оба порта одновременно согласно коммутационному байту: при значении 0x0 – в SpW1, при значении 0x1 – в SpW2, при значениях от 0x2 до 0xFF отправка в оба порта SpaceWire одновременно.

При работе со SpW-пакетами длиной больше Ethernet-фрейма (поле пакета SpW имеет длину 1494 или 1495 байт в зависимости от режима передачи данных SpW-пакетов) SpW-пакет автоматически разделяется Мостом и его программным обеспечением (ПО) на необходимое количество Ethernet-фреймов. В коммутационном режиме начальный Ethernet-фрейм SpW пакета имеет формат согласно Рисунок 6, все последующие фреймы этого SpW-пакета – формат согласно Рисунок 5. В режиме прямой передачи данных все Ethernet фреймы SpW-пакетов имеют формат согласно Рисунок 5.

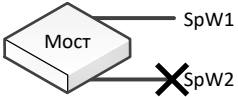
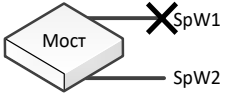
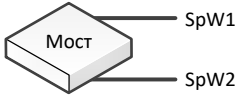






































Каждый из режимов передачи данных способен принимать конфигурационные фреймы по каналу Ethernet с ПК.



У моста предусмотрено **четыре режима передачи управляющих кодов SpW**:

1. Передача управляющих кодов запрещена
2. Режим прямой передачи только на заданный **SpW** порт
3. Широковещательная рассылка: рассылка управляющих кодов осуществляется по двум портам **SpW**
4. Транзитная передача управляющих кодов

Режимы управления каналами для приема/передачи данных через порты SpW

0. Управление каналами отключено. Передача по подключенному порту (SpW1 или SpW2). Если подключено два порта, то используется SpW1.
Коммутационный байт SpW пакета не влияет на работу.
Режим по умолчанию.
1. Передача и прием только через первый порт (SpW1)
Коммутационный байт SpW пакета не влияет на работу.
2. Передача и прием только через второй порт (SpW2)
Коммутационный байт SpW пакета не влияет на работу.
3. Управление каналами включено. Работает одновременно два порта (SpW1 и SpW2). При приеме данных из сети SpW к началу пакета добавляется байт с номером порта, на который пришел пакет. При передаче данных с ПК на мост коммутационный байт указывает, с какого порта SpW будут передаваться данные. Коммутационный байт должен задаваться пользователем при формировании SpW пакета, либо должна быть использована соответствующая [функция API передачи SpW пакетов из заданного порта](#).
4. Автоматический режим. В случае подключения двух портов (SpW1 и SpW2) работает управление каналами (режим 3). В случае подключения одного SpW-устройства мост начинает работать в режиме 1 или 2 соответственно.
5. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire.
Транзитный трафик не дублируется в Ethernet
6. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire.
Прослушивается порт SpW1, его входящий трафик дублируется в Ethernet
7. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire.
Прослушивается порт SpW2, его входящий трафик дублируется в Ethernet
8. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire.
Прослушиваются порты SpW1 и SpW2, их входящий трафик дублируется в Ethernet

Работа режимов управления каналами для приема/передачи данных из Ethernet в SpW при различных типах подключения SpW-устройств				
				
Режим	Ethernet-фрейм	Подключение типа 1	Подключение типа 2	Подключение типа 3
	 коммутационный байт,  байты SpW пакета			
0	0x0  	Передача в SpW1	Передача в SpW2	Передача в SpW1
	0x1  	Передача в SpW1	Передача в SpW2	Передача в SpW1
	0x2  	Передача в SpW1	Передача в SpW2	Передача в SpW1
	и все другие			
1	0x0  	Передача в SpW1	Отбрасывается	Передача в SpW1
	0x1  	Передача в SpW1	Отбрасывается	Передача в SpW1
	0x2  	Передача в SpW1	Отбрасывается	Передача в SpW1
	и все другие			
2	0x0  	Отбрасывается	Передача в SpW2	Передача в SpW2
	0x1  	Отбрасывается	Передача в SpW2	Передача в SpW2
	0x2  	Отбрасывается	Передача в SpW2	Передача в SpW2
	и все другие			
3	0x0  	Передача в SpW1	Отбрасывается	Передача в SpW1  
	0x1  	Отбрасывается	Передача в SpW2	Передача в SpW2  
	0x2  	Отбрасывается	Отбрасывается	Передается в оба порта SpW
	и все другие			
4	0x0  	Передача в SpW1	Передача в SpW2	Передача в SpW1  
	0x1  	Передача в SpW1	Передача в SpW2	Передача в SpW2  

	0x2   и все другие	Передача в SpW1	Передача в SpW2	Передается в оба порта SpW
--	---	-----------------	-----------------	-------------------------------

При приеме/передачи данных из SpW в Ethernet в режиме 3 и режиме 4 (в случае подключения обоих каналов SpW) к пакетам SpW добавляется коммуникационный байт, определяющий порт SpW, из которого пришел пакет.

Для пакета, принятого из порта SpW1, добавляется коммуникационный байт со значением 0x0.

Для пакета, принятого из порта SpW2, добавляется коммуникационный байт со значением 0x1.

При передаче данных из Ethernet в SpW в режиме 3 и режиме 4 (в случае подключения обоих каналов SpW) в случае, если значение коммутационного байта 2 и более, то отправка пакетов будет осуществляться в оба порта SpW.

При передаче данных из порта Ethernet в SpW во всех режимах в случае отсутствия возможности передачи данных в порт SpW часть пакетов может быть записана в буферную память моста (FIFO). После заполнения буфера, последующие поступающие пакеты отбрасываются.

В случае восстановления соединения все пакеты из буфера отправляются в соответствующий SpW-порт согласно заданному режиму работы.

Буфер сбрасывается только после отключения питания моста. Программный сброс буфера отсутствует.

Режимы управления каналами для приема/передачи управляющих кодов

0. Работа с подключенным портом (SpW1 или SpW2). Если подключены оба порта, то используется первый (SpW1).
1. Использование только порта SpW1.
2. Использование только порта SpW2.
3. Передача и прием с двух портов одновременно
Режим по умолчанию.
5. Транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Транзитный трафик не дублируется в Ethernet
6. Транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW1, его входящий трафик дублируется в Ethernet
7. Транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW2, его входящий трафик дублируется в Ethernet
8. Транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Прослушиваются порты SpW1 и SpW2, их входящий трафик дублируется в Ethernet

Состояние моста

Позволяет отправить Ethernet фрейм с текущим состоянием моста по сети Ethernet. Фрейм содержит детальную информацию о трафике. Состояние моста доступно в виде структуры.

По умолчанию отправка фрейма состояния моста включена с частотой отправки раз в секунду.

Структура состояния моста для каждого SpW-порта

```
struct spw_eth_port_state {
    unsigned long long tx_byte_count;
    unsigned long long rx_byte_count;
    unsigned tx_packet_count;
    unsigned rx_packet_count;
    unsigned rx_packet_count_EEP;
    unsigned tx_byte_count_sec;
    unsigned rx_byte_count_sec;
    u8 state;
} __attribute__((packed));
```

Общая структура состояния моста

Приходит всегда

```
struct spw_eth_state_new {
    unsigned time;
    struct spw_eth_port_state spw1;
    struct spw_eth_port_state spw2;
    u8 speed;
    unsigned short loopback_err_count1;
    unsigned short delay1;
    unsigned short loopback_err_count2;
    unsigned short delay2;
    __u8 switch_mode;
} __attribute__((packed));
```

Описание полей структуры состояния моста

Поле структуры	Описание
time	Время работы моста в секундах.
tx_byte_count	Количество переданных байт
rx_byte_count	Количество принятых байт
tx_packet_count	Количество переданных пакетов
rx_packet_count	Количество принятых пакетов (EOP и EEP)
rx_packet_count_EEP	Количество принятых EEP
tx_byte_count_sec	Количество отправленных байт за последнюю секунду
rx_byte_count_sec	Количество принятых байт за последнюю секунду
state	Состояние SpW 5 (101) – соединение по SpW установлено
speed	Установленная скорость соединения
loopback_err_count1	Служебное поле
delay1	Служебное поле
loopback_err_count2	Служебное поле

delay2	Служебное поле
switch_mode	Режим коммутатора (см. switch_mode в таблице «Описание полей структуры данных spw_eth_conf_header_2»)

Фильтрация по MAC-адресу

Режим фильтрации по MAC-адресу используется в случае подключения моста к ПК через сетевой коммутатор (см. [раздел 6.4](#)). В этом случае режим фильтрации гарантирует передачу трафика с моста только на требуемый ПК. Другие устройства, подключенные к сетевому коммутатору, не будут получать трафик от моста. Трафик сети не будет загружен информацией с моста.

По умолчанию фильтрация по MAC адресу включена.

Обнаружение ошибок последовательности фреймов Ethernet

Режим обеспечивает обнаружение ошибок последовательности фреймов Ethernet и информирование пользователя об этом.

Режим включен всегда.

Кредитование Ethernet

Режим обеспечивает гарантированную передачу SpW- пакетов при большой загрузке сети SpW.

Режим включен всегда.

Автоматическая настройка адреса моста

Режим автоматической настройки MAC-адреса моста. Два байта адреса выбираются случайно.

Автоматическая настройка адреса устройства для приема пакетов SpW с моста

Режим автоматической настройки MAC-адреса устройства для приема пакетов с моста.

В качестве MAC-адреса назначения пакетов, отправляемых с моста, будет указан MAC-адрес устройства, отправившего конфигурационный Ethernet-фрейм.

Конфигурационный фрейм используется для настройки моста (см. [раздел 6.6](#)).

Перезагрузка моста

Режим сброса следующих параметров моста в состояние по умолчанию:

- MAC-адреса удаленного узла
- MAC-адреса моста
- Режим фильтрации
- Скорость портов SpW

3 Требования к компьютеру

Компьютер, к которому подключается мост, должен быть:

- Оснащен ОС:
 - семейства **Windows**. Работа моста проверялась на Windows 7, Windows 10 или
 - семейства **Linux**. Работа моста проверялась на Ubuntu 12.04 LTS, Ubuntu 14.04 LTS, Ubuntu 16.04 LTS, CentOS Linux release 7.5.1804 (Core), Linux Debian 4.9.0;
- Оснащен интерфейсом Ethernet;
- Оснащен средой разработки
 - ОС семейства **Linux**: компилятор **gcc 4.3.5** и выше (для работы из консоли) или среда разработки, позволяющая компоновать пользовательские библиотеки (например, **IDE QtCreator**, которая используется в документации);
 - ОС семейства **Windows**: среда разработки, позволяющая компоновать пользовательские библиотеки (например, **IDE QtCreator**, которая используется в документации).

4 Требования к устройствам SpaceWire

Подключаемые SpaceWire устройства должны соответствовать стандарту ECSS-E-50-12C.

5 Требования к питанию

Мост SpaceWire потребляет от 5 до 12 В.

Подача питания осуществляется от сети через внешний адаптер питания (поставляется в комплекте с мостом).

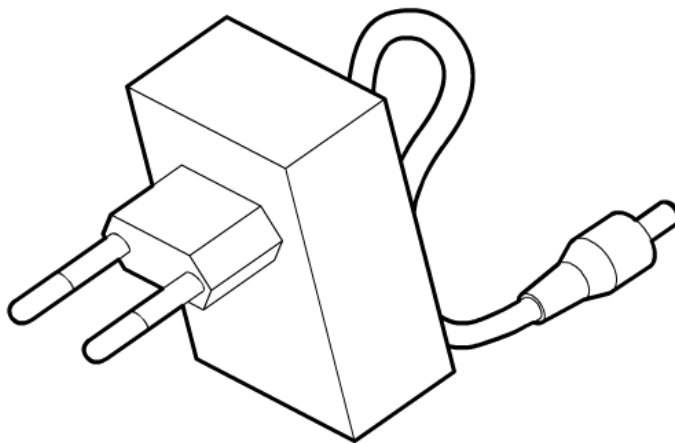


Рисунок 7 – Адаптер питания

6 Подключение и работа с мостом

6.1 Общие правила подключения моста к ПК и SpaceWire-устройствам

1. Вначале осуществляется подключение к мосту кабелей SpaceWire и Ethernet, затем подключается кабель питания.

- Мост подключается к ПК через Ethernet напрямую или не более чем через один сетевой коммутатор.

6.2 Установка SDK моста Ethernet-SpaceWire

Предоставляемый программный интерфейс приложений (API) обеспечивает поддержку работы моста Ethernet-SpaceWire для приложений пользователя:

- Прием и передача пакетов SpaceWire;
- Прием и передача управляющих кодов (time-коды, interrupt-коды, acknowledge-коды, коды неопределенные стандартом CC01 и CC11);
- Настройка скорости портов SpaceWire.

Для поддержки API предоставляется SDK в виде библиотеки на языке Си.

Список файлов SDK-библиотеки

WpdPack_4_1_2.zip
WinPcap_4_1_3.exe
SpWEth_bridge
 src
 headers
 spw_eth.h
 spw_eth_print.h
 spw_eth_structure.h
 libs
 winpcap (список файлов аналогичен содержанию WpdPack_4_1_2.zip)
 Include
 Lib
 libspw_eth_unix.a
 libspw_eth_win.a
 demo
 conf_packet.c (исходный код приложения *configure_bridge*)
 Makefile
 spw_eth_test.c (исходный код приложения *bridge_app*)
 receive_packets.c (исходный код приложения *example_receive_packets*)
 qt_pro_files
 configure_bridge.pro
 bridge_app.pro
 example_receive_packets.pro

Примечание. Файлы *.h и *.c используют кодировку UTF 8

Библиотека работает в пространстве пользователя путем компоновки к пользовательским приложениям.

6.2.1 Установка SDK на компьютер с Linux

Для работы с предоставляемой SDK на ОС семейства **Linux** необходимо подключить заголовочные файлы библиотеки в коде пользовательского приложения и произвести компоновку с предоставленным файлом статической библиотеки.

1. Скопировать все файлы с поставляемого диска в рабочий каталог компьютера.
2. Компиляцию приложений (см. [раздел 8](#)) можно осуществить двумя способами:
 1. Перейти в каталог `demo` и выполнить команду компиляции `make`.
 2. Открыть необходимый проект из каталога `qt_pro_files` в IDE **QtCreator** и осуществить его компиляцию.

ВАЖНО! В случае использования других сред разработки необходимо:

1. Создать проект консольного приложения на языке Си
2. Осуществить компоновку библиотеки SDK моста
3. Подключить заголовочные файлы SDK в коде написанных приложений
4. Использовать один из [C-файлов приложения для работы с мостом](#)

3. Запустить с правами суперпользователя одно из приложений.

Примечание. Рекомендуется использовать **Qt Creator** с компилятором **MinGW**. Это гарантирует корректное распознавание библиотекой текущей ОС, на которой идет работа. В случае возникновения проблем (ни один из макросов `WIN`, `LINUX` в файле `spw_eth_structure.h` не проинициализирован, строки 7-13), необходимо самостоятельно проинициализировать макросы характеризующие ОС, в заголовочном файле `spw_eth_structure.h` (строки 15, 16). Для работы в ОС **Windows** служит макрос `WIN`, для ОС **Linux** — макрос `LINUX`.

```
kiwi@kiwi-System-Product-Name:~/work/SpWEth_bridge_work/work_copy/reliz$ cd demo/
kiwi@kiwi-System-Product-Name:~/work/SpWEth_bridge_work/work_copy/reliz/demo$ ls
conf_packet.c  Makefile  receive_packets.c  spw_eth_test.c
kiwi@kiwi-System-Product-Name:~/work/SpWEth_bridge_work/work_copy/reliz/demo$ make
gcc conf_packet.c -L ../src/libs/ -lspw_eth_unix -o configure_bridge
gcc spw_eth_test.c -L ../src/libs/ -lspw_eth_unix -o bridge_app
gcc receive_packets.c -L ../src/libs/ -lspw_eth_unix -o example_receive_packets
kiwi@kiwi-System-Product-Name:~/work/SpWEth_bridge_work/work_copy/reliz/demo$ sudo ./configure_bridge
[sudo] password for kiwi:
Client: bind with device --> eth0
=====
Current source      address: 0:24:8c:3c:cf:4e
Current destination address: 0:1:2:3:4:5
=====
Menu
=====
Conf packet:: 24 bytes
[+0000]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
[+0010]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
  1 Create new packet:
  2 Load packet:
  3 Save current packet:
  4 Send current packet:
  5 Set destination address:
  0 Exit:
Enter value:
```

Рисунок 8 – Компиляция и запуск приложений на компьютере с Linux

Примечание. [Исходные коды приложений](#) содержат примеры работы с основным функционалом API. Их можно использовать для создания пользовательских программ.

Пример Makefile, который компилирует приложения под ОС семейства Linux

```
C=gcc
CFLAGS=-c -Wformat=0 -fno-stack-protector

all: configure_bridge bridge_app example_receive_packets

configure_bridge:
$(CC) conf_packet.c -L ../src/libs/ -lspw_eth_unix -o
configure_bridge

example_receive_packets:
$(CC) receive_packets.c -L ../src/libs/ -lspw_eth_unix -o
example_receive_packets

bridge_app:
$(CC) spw_eth_test.c -L ../src/libs/ -lspw_eth_unix -o bridge_app

cleanall:
rm -rf *.o configure_bridge bridge_app example_receive_packets

clean:
rm -rf *.o
```

6.2.2 Установка SDK на компьютер с Windows

Для работы с мостом Ethernet-SpaceWire в ОС **Windows** дополнительно используется библиотека **WinPcap (Packet Capture)**. Для установки SDK необходимо:

- 1.1. Скопировать все файлы с поставляемого диска в рабочий каталог компьютера.
- 1.2. Установить **WinPcap версии 4.1.3** (допускается использование **Win10Pcap** и **Npcap in WinPcap API - compatible mode** (Npcap в режиме совместимости с WinPcap API).
- 1.3. Установить **WinPcap Developer's Pack версия 4.1.2**.
- 1.4. Открыть среду разработки.

Примечание. Можно использовать любую среду разработки, позволяющую компановать пользовательские библиотеки (в документации и примере используется **IDE QtCreator**).

- 1.5. При использовании **QtCreator** открыть любой проект из каталога **qt_pro_files**.

ВАЖНО! В случае использования других сред разработки необходимо:

1. Создать проект консольного приложения на языке Си
2. Осуществить компановку библиотек: SDK моста и библиотек WinPcap
3. Подключить заголовочные файлы SDK в коде написанных приложений
4. Использовать один из [C-файлов приложения для работы с мостом](#)

- 1.6. Провести компиляцию проекта

Примечание. Рекомендуется использовать **Qt Creator** с компилятором **MinGW**. Это гарантирует корректное распознавание библиотекой текущей ОС, на которой идет работа. В случае возникновения проблем (ни один из макросов WIN, LINUX в файле **spw_eth_structure.h** не проинициализирован, строки 7-13), необходимо самостоятельно проинициализировать макросы, характеризующие ОС, в заголовочном файле **spw_eth_structure.h** (строки

15,16). Для работы в ОС **Windows** служит макрос **WIN** , для ОС **Linux** – макрос **LINUX**.

Пример компоновки SDK моста и библиотек WinPcap к пользовательской программе в среде IDE QtCreator (*.pro файл), компилирующей приложение как для ОС семейства Windows, так и для ОС семейства Linux

```
#configure_bridge.pro
TEMPLATE = app
CONFIG += console
CONFIG -= qt

win32 {
    INCLUDEPATH += "../src/libs"
    LIBS += ../src/libs/libspw_eth_win.a
    INCLUDEPATH += "../src/libs/winpcap/Include"
    INCLUDEPATH += "../src/libs/winpcap/Lib"
    LIBS += -L ../src/libs/winpcap/Lib -lwinpcap
}
unix {
    INCLUDEPATH += "../src/libs"
    LIBS += ../src/libs/libspw_eth_unix.a
}

HEADERS += \
    ../src/headers/spw_eth_structure.h \
    ../src/headers/spw_eth_print.h \
    ../src/headers/spw_eth.h
SOURCES += ../demo/conf_packet.c
```

Примечание. Приложения для работы с мостом содержат примеры работы со основным функционалом API. Их можно использовать для создания пользовательских программ.

6.3 Архитектура SDK в ОС

6.3.1 Архитектура библиотеки SDK в Linux

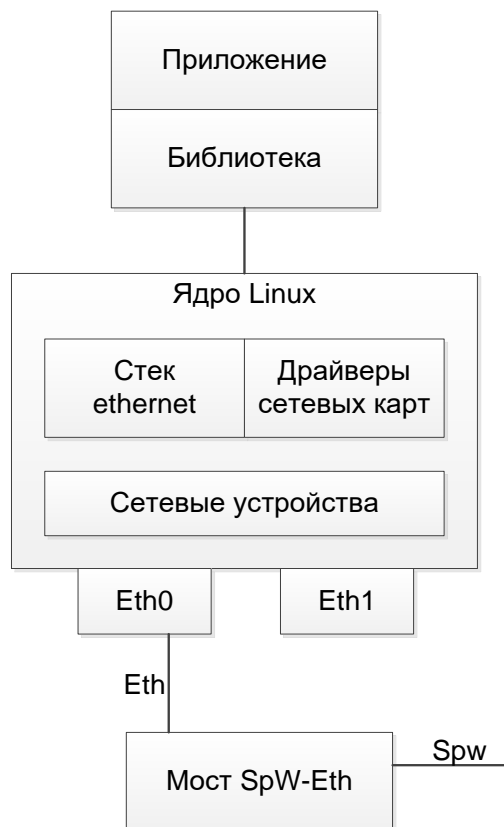


Рисунок 9 – Архитектура библиотеки в общей архитектуре ПО ОС Linux

6.3.2 Архитектура библиотеки SDK в Windows

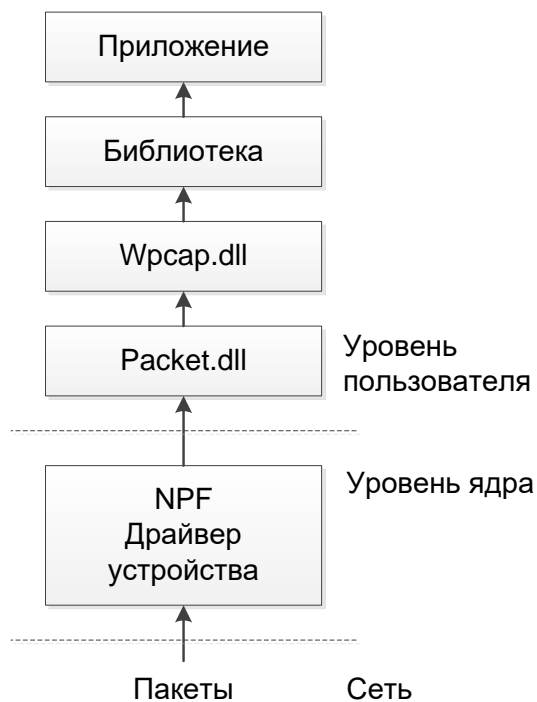


Рисунок 10 – Архитектура библиотеки SDK в общей архитектуре ПО ОС Windows

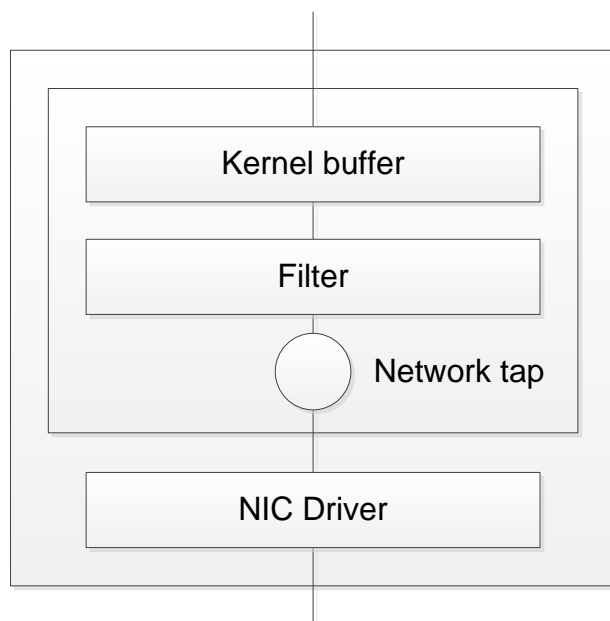


Рисунок 11 – Структура Netgroup Packet Filter (NPF)

- **NIC Driver** позволяет управлять сетевой картой (выявлять прерывания, отправлять пакеты в сеть, выключать и перезагружать сетевую карту и др.);
- **Network tap** перехватывает трафик;
- **Filter** настроен на отбор пакетов протокола SpW-Ethernet (type 0x06AB) ;
- **Kernel buffer** хранит пакеты до момента их обработки;
- **Wpcap.dll** необходима для работы с SDK. Связывают уровень драйвера с библиотекой SDK;
- В состав **WinPcap** входят драйверы для ОС Windows, которые используют **NDIS** (Network Driver Interface Specification) для чтения пакетов, которые получает сетевая карта, и низкоуровневые библиотеки для взаимодействия с драйверами сетевых интерфейсов.

6.4 Инициализация моста

Для инициализации моста необходимо:

1. Установить SDK моста Ethernet-SpaceWire (см. [раздел 6.2](#))
2. Подключить мост к ПК через Ethernet-кабель напрямую:

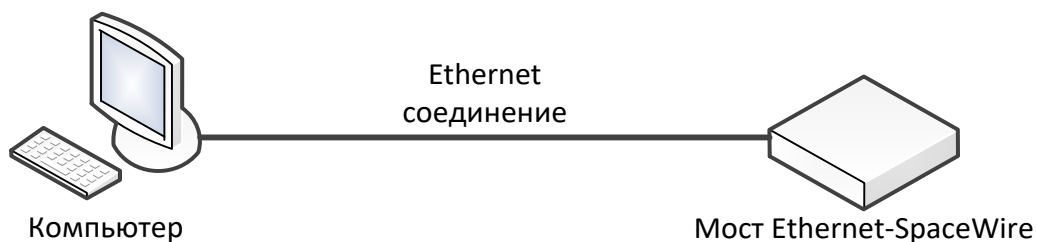


Рисунок 12 – Подключение моста напрямую к компьютеру

или не более чем через один сетевой коммутатор

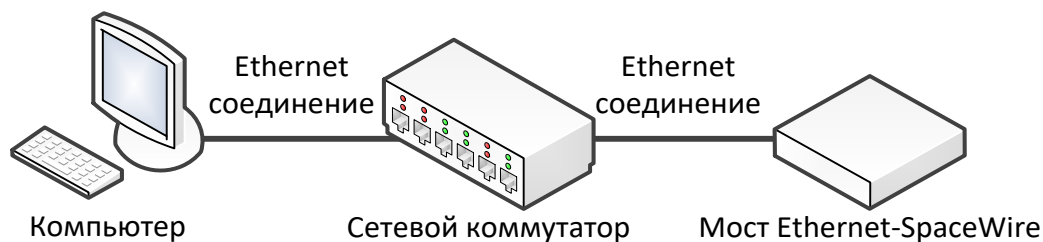


Рисунок 13 – Подключение моста к компьютеру через сетевой коммутатор

3. Подать питание на мост.
4. Доступно два способа инициализации моста:
 1. Использование функции **SpW_Socket_Init**. Для ОС Windows есть 2 варианта:
 - 1.1. Инициализации моста в диалоговом режиме. Используется функция **SpW_Socket_Init_Dialog** библиотеки моста.
 - 1.2. Инициализации моста по номеру сетевого устройства. Для получения списка доступных сетевых устройств используются функция **Create_List_Devices** библиотеки моста. Для инициализации по номеру сетевого устройства используются функция **SpW_Socket_Init** библиотеки моста.
 2. Через приложения (см. [раздел 8](#)). В них функция инициализации запускается автоматически.

Примечание. При работе с SDK-библиотекой необходимо выбирать имя сетевого интерфейса, к которому подключен мост посредством кабеля Ethernet. Для ОС Windows этот выбор необходимо выполнять при запуске приложений, для ОС семейства Linux этот выбор необходимо выполнять в случае если подключение произведено не по сетевому интерфейсу с именем “eth0”.

Функция инициализации моста в диалоговом режиме **SpW_Socket_Init_Dialog** (только для ОС Windows)

Windows

```
pcap_t* SpW_Socket_Init_Dialog();
```

Выбор устройства (моста)

Выбор устройства происходит в диалоговом режиме.

Возвращаемое значение: сокет для приема/передачи

В случае успешной инициализации, функция вернет сокет, через который в дальнейшем будет производиться работа с мостом.

В случае возникновения ошибки инициализации моста функция возвращает отрицательное значение (код ошибки).

Функция получения списка сетевых устройств Create_List_Devices (только для ОС Windows)	
Windows	
<code>pcap_if_t * Create_List_Devices();</code>	
Возвращаемое значение: структура содержащая список сетевых устройств	
<p>Возвращается структура содержащая список сетевых устройств, нумерация устройств начинается с 0. Для отображения описания устройств необходимо использовать поля name и description полученного списка типа pcap_if_t*.</p> <p>Также при выполнении функции заполняются служебные поля библиотеки.</p> <p>В случае возникновения ошибки инициализации моста функция возвращает отрицательное значение (код ошибки).</p>	

Функция инициализации моста SpW_Socket_Init	
Linux	Windows
<code>int SpW_Socket_Init(char *device);</code>	<code>pcap_t* SpW_Socket_Init(unsigned device);</code>
Выбор устройства (моста)	
<p>Входной параметр:</p> <p>device – имя сетевого устройства Ethernet (например, eth0).</p>	<p>Входной параметр:</p> <p>device – номер сетевого устройства Ethernet, через который подключен мост. Нумерация начинается с 0. Для получения списка сетевых устройств необходимо использовать функцию Create_List_Devices библиотеки моста.</p>
Возвращаемое значение: сокет для приема/передачи	
<p>В случае успешной инициализации, функция вернет сокет, через который в дальнейшем будет производиться работа с мостом.</p> <p>В случае возникновения ошибки инициализации моста функция возвращает отрицательное значение (код ошибки).</p>	

Пример использования функции инициализации моста	
Linux	Windows
<pre>int rawsock; rawsock = SpW_Socket_Init("eth0");</pre>	<pre>pcap_t *rawsock; rawsock = SpW_Socket_Init_Dialog();</pre>
Пример использования см. исходные коды приложений .	

**Коды ошибок при неудачной попытке инициализации моста
(для ОС Windows коды ошибок справедливы для функций Create_List_Devices,
SpW_Socket_Init, SpW_Socket_Init_Dialog)**

Linux		Windows	
Код	Значение	Код	Значение
-1	Ошибка при создании raw-сокета	-1	Ошибка при поиске сетевых устройств
-2	Ошибка получения индекса сетевого устройства	-2	Сетевое устройство не найдено, необходимо убедиться в том, что библиотека WinPcap установлена
-3	Ошибка связывания raw-сокета с сетевым устройством	-3	Ошибка при вводе номера сетевого устройства (выход за границы диапазона)
-4	Не удалось получить адрес сетевого устройства	-4	Невозможно открыть сетевое устройство (не поддерживается библиотекой WinPcap)
		-5	Ошибка компиляции фильтра (неправильный синтаксис)
		-6	Ошибка при установке фильтра

6.5 Подключение к мосту устройств с интерфейсом SpaceWire

Перед процедурой настройки моста необходимо завершить процедуру инициализации моста (см. [раздел 6.4](#)).

Устройство(а) с интерфейсом SpaceWire подключаются к мосту одним из допустимых способов:

2.1.1. Подключение только канала SpaceWire 1:

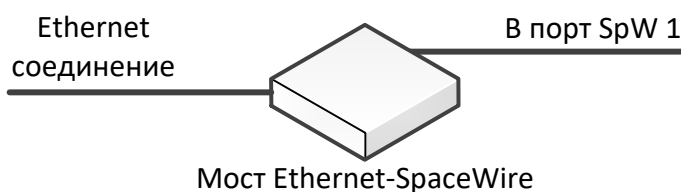


Рисунок 14 – Подключение к мосту SpW-канала в порт 1

2.1.2. Подключение только канала SpaceWire 2:

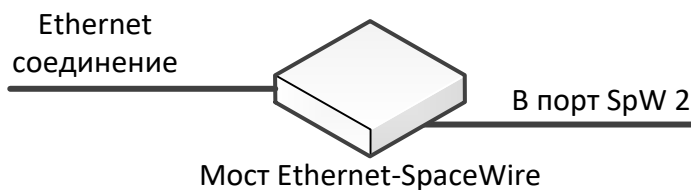


Рисунок 15 – Подключение к мосту SpW-канала в порт 2

3. Подключение двух каналов SpaceWire (1 и 2):

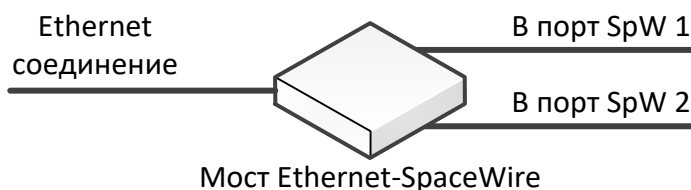


Рисунок 16 – Подключение к мосту SpW-каналов в порты 1 и 2

Для использования двух портов SpW используется специальный режим управления каналами (см. [функцию установки режима управления каналами](#)).

6.6 Настройка моста

Для настройки моста необходимо отправить на мост через сеть Ethernet конфигурационный фрейм. Доступно три способа отправки конфигурационного фрейма:

1. Через **приложение `configure_bridge`** (см. [раздел 6.6.2](#)).
2. Через функцию **API `SpW_Send_Conf_Packet`**, позволяющую настроить все параметры моста (см. [раздел 6.6.3.1](#)).
3. Через функции API, позволяющую настроить каждый параметр моста отдельно (см. [раздел 6.6.3.2](#)).

Параметры настройки моста:

1. Установка скорости портов SpW
2. Установка MAC-адреса моста
3. Установка MAC-адреса удаленного узла
4. [Автоматическая настройка адреса моста](#)
5. [Автоматическая настройка удаленного узла](#)
6. Установка [режима фильтрации по MAC-адресу](#)
7. [Перезагрузка моста](#)
8. Установка режима управления каналами
 1. Для [приема/передачи данных](#)
 2. Для [приема/передачи управляющих кодов](#)

ВАЖНО! Прием/передача данных, прием/передача управляющих кодов должны настраиваться одновременно.

9. Установка частоты посылки фреймов [состояния моста](#).

ВАЖНО! В случае, если необходимо задать MAC-адрес моста или удаленного узла **11 : 22 : 33 : 44 : 55 : 66**, то в конфигурационном фрейме (и приложении) необходимо задавать значение MAC-адреса зеркально:
66 : 55 : 44 : 33 : 22 : 11.

- Примечания.**
1. Параметры моста не сохраняются при отключении питания.
 2. В случае использования в сети Ethernet-SpaceWire более одного моста Ethernet-SpaceWire необходимо предварительно настроить каждый мост по отдельности (см [раздел 6.8](#)).
 3. В текущей версии SDK-библиотеки моста Ethernet SpaceWire при работе под ОС **Windows** MAC-адрес сетевого оборудования ПК автоматически не распознается и не передается в приложения. Это не влияет на работу с мостом. В этой ситуации MAC-адрес источника в

отправляемых фреймах будет иметь нулевое значение (00:00:00:00:00:00). При работе в операционных системах семейства Linux приложения будут распознавать реальный MAC-адрес сетевого оборудования ПК.

Требуемый MAC-адрес можно задать с помощью функции `int set_source_MAC_address (unsigned char* address, int len)` библиотеки API.

6.6.1 Настройки моста по умолчанию

1. Скорость портов **SpW**: 400 Мбит/с
2. **MAC-адрес моста**: 00:01:02:03:04:05
3. **MAC-адрес удаленного узла**: FF:FF:FF:FF:FF:FF
По умолчанию через порт Ethernet мост работает в режиме широковещания (broadcast)
4. Режим фильтрации включен
5. Установка режима управления каналами
 1. Для приема/передачи данных: [режим 0](#)
 2. Для приема/передачи управляющих кодов: [режим 3](#)
6. Режим отправки состояния моста включен с частотой отправки раз в секунду

6.6.2 Приложение `configure_bridge`

Приложение `configure_bridge` позволяет настраивать работу моста посредством создания и отправки конфигурационных фреймов.

Для работы с приложением необходимо предварительно осуществить его сборку (см. [раздел 8.1](#)).

Функции приложения:

1. Создание конфигурационного фрейма в диалоговом режиме.
2. Сохранение заданных настроек моста в файл (**config.dat**).
3. Загрузка настроек моста из файла (**config.dat**).
4. Отправка заданных настроек моста в конфигурационном фрейме на мост.
5. Задание MAC-адреса моста, на который будет отправлен конфигурационный фрейм.

Примечание. При сохранении заданных настроек моста в файл указанный MAC-адрес моста не сохраняется.

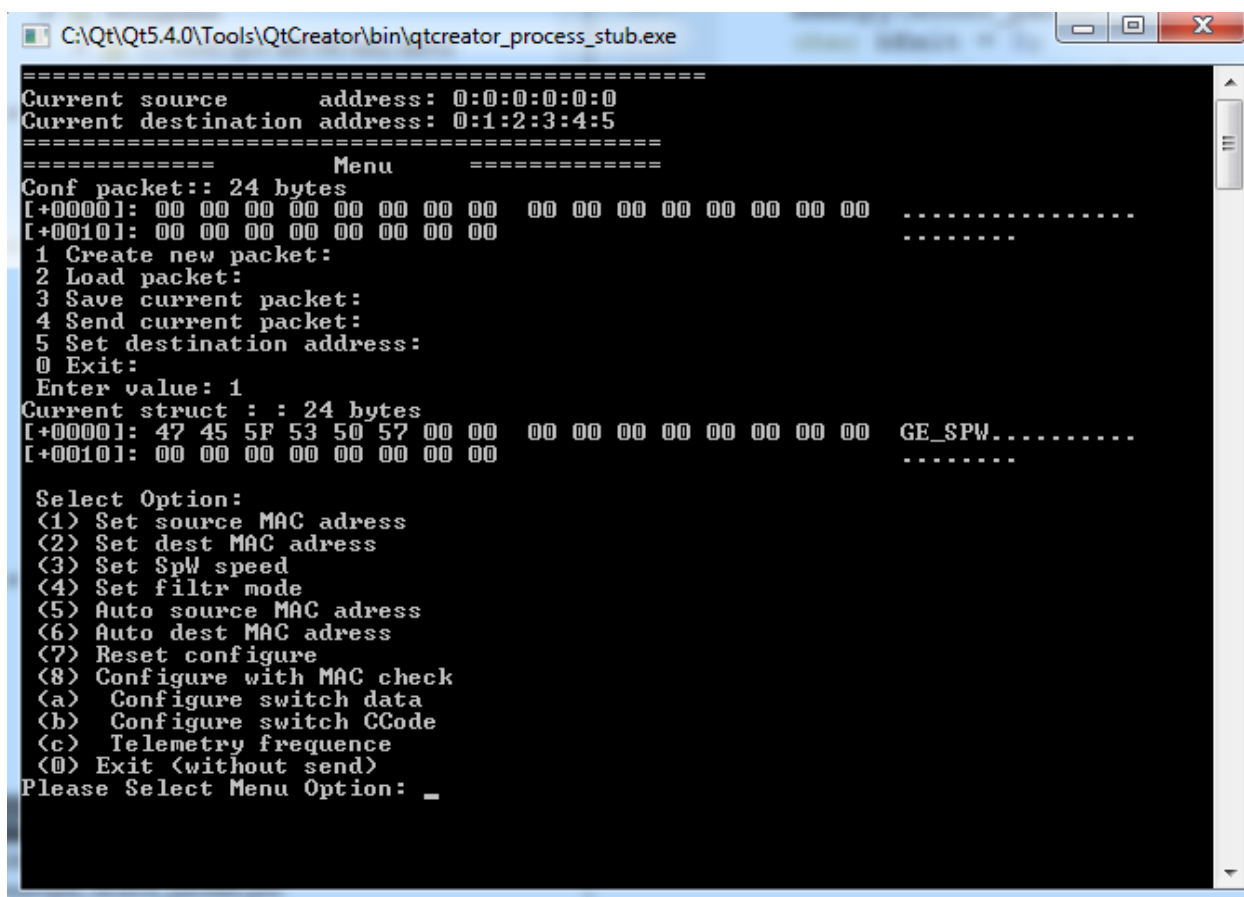


Рисунок 17 – Меню configure_bridge и меню настройки конфигурации моста

6.6.3 Настройка моста через функции API

Для первоначальной настройки моста необходимо настроить следующие параметры:

1. Скорость портов SpW.
2. Период отправки пакетов состояний моста.
3. Режим управления каналами для передачи данных.
4. Режим управления каналами для передачи управляющих кодов.

6.6.3.1 Настройка всех параметров моста функцией SpW_Send_Conf_Packet

Для работы функции необходимо заполнить структуру данных `spw_eth_conf_header_2` (располагается в файле `spw_eth_structure.h`).

Примечание. Для заполнения структуры рекомендуются использовать макросы, представленные в заголовочном файле `spw_eth_structure.h`.

Описание полей структуры данных <code>spw_eth_conf_header_2</code>	
Название полей	Описание
<code>GE_SPW[ETH_ALEN]</code>	Строка "GE_SPW"
<code>edit_0</code>	Маска устанавливаемых параметров

edit_1	Бит 0 — установка MAC-адреса моста Бит 1 — установка MAC-адреса удаленного узла Бит 2 — установка скорости SpW Бит 3 — установка режима фильтрации Бит 4 — автоматическая настройка MAC-адреса моста Бит 5 — автоматическая настройка MAC-адреса удаленного узла Бит 6 — сброс в конфигурацию по умолчанию Бит 7 — проверка MAC-адреса при настройке моста. Должен быть всегда равен 1.
	Бит 0 — установка режима управления каналами (прием/передача данных и управляющих кодов) Бит 1 — установка частоты отправки фреймов состояния моста Биты 2-7 — резерв
MAC-src	Адрес моста
MAC-dst	Адрес удаленного узла, на который мост будет посылать данные
ВАЖНО! В случае, если необходимо задать MAC-адрес моста или удаленного узла 11 : 22 : 33 : 44 : 55 : 66 , то в конфигурационном фрейме (и приложении) необходимо задавать значение MAC-адреса зеркально: 66 : 55 : 44 : 33 : 22 : 11.	
SpW-speed	Скорость соединения SpW 0 — 400 Мбит/с 1 — 300 Мбит/с 2 — 200 Мбит/с 4 — 100 Мбит/с 8 — 50 Мбит/с 16 — 10 Мбит/с 32 — 500 Мбит/с Неверное значение параметра установит скорость по умолчанию (400 Мбит/с). Смена скорости возможна только при правильном значении MAC-адреса
filtr	Фильтрация пакетов
switch mode	Биты 0-3. Режимы передачи данных 0 — управление каналами отключено. Передача по подключенному порту. Если подключено по двум портам, то используется порт SpW1 1 — передача и прием только через порт SpW1 2 — передача и прием только через порт SpW2 3 — режим управления каналами. Работают оба порта. При приеме данных из сети SpW к началу пакета добавляется байт с номером порта, на который пришел пакет. При передаче в обратном

	<p>направлении первый байт указывает с какого порта передавать данные</p> <p>4 – автоматический режим. Если подключено 2 SpW-порта, то работает управление каналами (режим 3). В случае подключения одного SpW устройства мост начинает работать в режиме 1 или 2 соответственно</p> <p>5 – транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Транзитный трафик не дублируется в Ethernet</p> <p>6 – транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW1, его входящий трафик дублируется в Ethernet</p> <p>7 – транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW2, его входящий трафик дублируется в Ethernet</p> <p>8 – транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушиваются порты SpW1 и SpW2, их входящий трафик дублируется в Ethernet</p> <p>Биты 4-7. Режимы передачи управляющих кодов (Ccode)</p> <p>0 – управление каналами отключено, передача по подключенному порту. Если подключено два порта — передача только по первому</p> <p>1 – передача и прием только в первый порт</p> <p>2 – передача и прием только во второй порт</p> <p>3 – передача и прием по двум портам</p> <p>5 – транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Транзитный трафик не дублируется в Ethernet</p> <p>6 – транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW1, его входящий трафик дублируется в Ethernet</p> <p>7 – транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW2, его входящий трафик дублируется в Ethernet</p> <p>8 – транзитный режим передачи управляющих кодов с порта SpaceWire на порт SpaceWire. Прослушиваются порты SpW1 и SpW2, их входящий трафик дублируется в Ethernet</p>
stat_freq	<p>Время между отправками фреймов состояния моста в секундах.</p> <p>0 – отправка фреймов состояния моста отключена.</p>

Пример использования функции **SpW_Send_Conf_Packet** для настройки моста (Linux и Windows)

```
struct spw_eth_conf_header_2 config_data;
config_data.edit0 = CONF_SET_SPEED;
config_data.Spw_Speed = 2; /* установка скорости 200 Мбит/с */
config_data.edit1 = CONF_SET_SWITCH_MODE | CONF_SET_STATUS_FREQ;
config_data.stat_freq = 0; /* отключение отправки состояния моста */
config_data.switch_mode =
SWITCH_MODE_DATA_SET(SWITCH_MODE_DATA_ENABLE) |
SWITCH_MODE_CCODE_SET(SWITCH_MODE_CCODE_DOUBLE); /* управление
каналами включено в режиме 3 */
SpW_Eth_Send_Conf_Packet(rawsock, config_data); /* отправка
конфигурации на мост */
```

Пример использования функции в [коде приложения configure_bridge](#).

6.6.3.2 Настройка каждого параметра моста функциями API

Параметры моста возможно изменять, отправляя новые конфигурационные фреймы.

Функция установки скорости порта SpW

Linux		Windows	
int SpW_Eth_set_SpW_Speed (int s, int speed);		int SpW_Eth_set_SpW_Speed (pcap_t *fp, int speed);	
Входные параметры			
s	сокет приема	fp	дескриптор для приема
speed	значение скорости: 10, 50, 100, 200, 300, 400	speed	значение скорости: 10, 50, 100, 200, 300, 400

Функция установки MAC-адреса удаленного узла, на который мост будет высылать информацию

Linux		Windows	
int SpW_Eth_Set_MAC_Dest (int s, char MAC[]);		int SpW_Eth_Set_MAC_Dest (pcap_t *fp, char MAC[]);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи
MAC	значение MAC адреса (6 байт)	MAC	значение MAC адреса (6 байт)

Функция установки MAC-адреса моста

Linux		Windows	
int SpW_Eth_Set_MAC_Source (int s, char MAC[]);		int SpW_Eth_Set_MAC_Source (pcap_t *fp, char MAC[]);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи
MAC	значение MAC адреса (6 байт)	MAC	значение MAC адреса (6 байт)

Функция автоматической настройки MAC-адреса назначения			
Linux		Windows	
int SpW_Eth_Set_Auto_MAC_Dest (int s);		int SpW_Eth_Set_Auto_MAC_Dest (pcap_t *fp);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи

Функция автоматической настройка MAC-адреса устройства			
Linux		Windows	
int SpW_Eth_Set_Auto_MAC_Source (int s);		int SpW_Eth_Set_Auto_MAC_Source (pcap_t *fp);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи

Функция установки режима фильтрации			
Linux		Windows	
int SpW_Eth_Set_Filtr (int s, unsigned filtr);		int SpW_Eth_Set_Filtr (pcap_t *fp, unsigned filtr);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи
filtr	фильтрация 1 – включена, 0 – выключена	filtr	фильтрация 1 – включена, 0 – выключена

Функция перезагрузки моста			
Linux		Windows	
int SpW_Eth_Reset(int s);		int SpW_Eth_Reset(pcap_t *fp);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи

Функция установки режима управления каналами			
Linux		Windows	
int SpW_Eth_Set_Switch_Mode (int s, unsigned mode_data, unsigned mode_ccode);		int SpW_Eth_Set_Switch_Mode (pcap_t *fp, unsigned mode_data, unsigned mode_ccode);	
Входные параметры			
s	сокет для передачи	fp	сокет для передачи
mode_data	режим управления каналами для приема/передачи данных	mode_data	режим управления каналами для приема/передачи данных
mode_ccode	режим управления каналами для приема/передачи управляющих кодов	mode_ccode	режим управления каналами для приема/передачи управляющих кодов
Режимы управления каналами для приема/передачи данных (mode_data)			
<div>0. Управление каналами отключено. Передача по подключенному порту. Если подключено два порта, то используется порт SpW1 (настройка по умолчанию)</div> <div>1. Использование только порта SpW 1</div> <div>2. Использование только порта SpW 2</div> <div>3. Режим управления каналами. Работают оба порта. При приеме данных из сети SpW к началу пакета добавляется байт с номером порта, на который пришел пакет. При передаче в обратном направлении первый байт указывает на то, с какого порта SpW передавать данные (0x0 – передача в SpW 1, 0x1 – передача в SpW 2, 0x2-0xFF – передача в оба порта SpW)</div> <div>4. Автоматический режим. Если подключено 2 SpW-порта, то работает управление каналами (режим 3). В случае подключения одного SpW-устройства мост начинает работать в режиме 1 или 2 соответственно</div> <div>5. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Транзитный трафик не отправляется в Ethernet</div> <div>6. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW1, его входящий трафик транслируется в Ethernet</div> <div>7. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW2, его входящий трафик транслируется в Ethernet</div> <div>8. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушиваются порты SpW1 и SpW2, их входящий трафик транслируется в Ethernet</div>			
Режимы управления каналами для приема/передачи управляющих кодов (mode_ccode)			
<div>0. Работа с подключенным портом. Если подключены оба порта, то используется первый</div> <div>1. Использование только порта SpW 1</div> <div>2. Использование только порта SpW 2</div> <div>3. Передача и прием с двух портов одновременно (настройка по умолчанию)</div> <div>5. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Транзитный трафик не дублируется в Ethernet</div> <div>6. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW1, его входящий трафик дублируется в Ethernet</div> <div>7. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушивается порт SpW2, его входящий трафик дублируется в Ethernet</div> <div>8. Транзитный режим передачи данных с порта SpaceWire на порт SpaceWire. Прослушиваются порты SpW1 и SpW2, их входящий трафик дублируется в Ethernet</div>			

Функция установки частоты отправки фреймов состояния моста			
Linux		Windows	
int SpW_Eth_Set_Status_freq (int s, unsigned period);		int SpW_Eth_Set_Status_freq (pcap_t *fp, unsigned period);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи
period	частота отправки в секундах. При нулевом значении, фреймы состояния моста не отправляются	period	частота отправки в секундах. При нулевом значении, фреймы состояния моста не отправляются

Пример настройки моста с использованием отдельных функций библиотеки (Linux и Windows)
<pre>SpW_Eth_set_SpW_Speed(rawsock, 200); //установка скорости (по умолчанию 400 Мбит/с) SpW_Eth_Set_Status_freq(rawsock, 0); //отправка состояния моста отключена SpW_Eth_Set_Switch_Mode(rawsock, 0x3, 0x3); // управление каналами включено</pre>

6.7 Основная работа с мостом

Работа с мостом осуществляется тремя способами:

1. Через приложение **bridge_app** (см. [раздел 6.7.1](#))
2. Через приложение **example_receive_packets** (см. [раздел 6.7.2](#))
3. Через соответствующие функции API (см. [раздел 6.7.3](#))

6.7.1 Приложение bridge_app

Приложение **bridge_app** предназначено для приема и передачи пакетов данных, управляющих кодов и других операций с мостом.

Для работы с приложением необходимо предварительно осуществить его сборку (см. [раздел 8.1](#)).

Функции приложения:

1. Отправка SpW-пакетов заданной длины:
 - а. Отправка одиночного SpW-пакета (дополнительно указывается длина и значения путевого адреса)
 - б. Отправка заданного количества SpW-пакетов (дополнительно указывается длина и значение путевого адреса пакетов)
 - в. Отправка одиночного SpW-пакета с заданного порта (дополнительно указывается длина и значение путевого адреса)
2. Прием пакетов:
 - а. Прием одиночного SpW-пакета
 - б. Прием заданного количества SpW-пакетов

- с. Прием одиночного Ehternet-фрейма заданного типа
- 3. Отображение состояния моста
- 4. Установка скорости в портах SpaceWire
- 5. Сброс устройства на состояние по умолчанию
- 6. Отправка одного или группы управляющих кодов следующих типов:
 - а. Отправка маркера времени (**Time-code**)
 - б. Отправка кода прерывания (**Int-code**)
 - с. Отправка кода подтверждения (**Ack-code**)
 - д. Отправка кода неопределенного стандартом (**CC11-code**)
 - е. Отправка кода неопределенного стандартом (**CC01-code**)

Примечание. Группа может содержать только один тип кодов.

- 7. Установка MAC-адреса моста, на который осуществляется отправка.
- 8. Изменение типа управляющих кодов: 5-битные / 6-битные.
По умолчанию отправляются 6-битные управляющие коды.
- 9. Включение/отключение следующих функций обработчиков:
 - а. Прием управляющего кода
 - б. Прием фрейма состояния моста
 - с. Прием фрейма отчета об ошибке

Примечание. 1. В приложении размер SpW-пакета максимально возможный для приема установлен в 7500 байт (в [коде файла spw_eth_test.c](#) макрос **BUF_SIZE**).
2. При использовании функций приема SpW-пакетов из сети Ethernet необходимо учитывать, что если используются функции, не обеспечивающие фильтрацию принимаемых SpW-пакетов по MAC-адресу, то функции будут также принимать отправляемые SpW-пакеты (см. [раздел 8.2](#)).

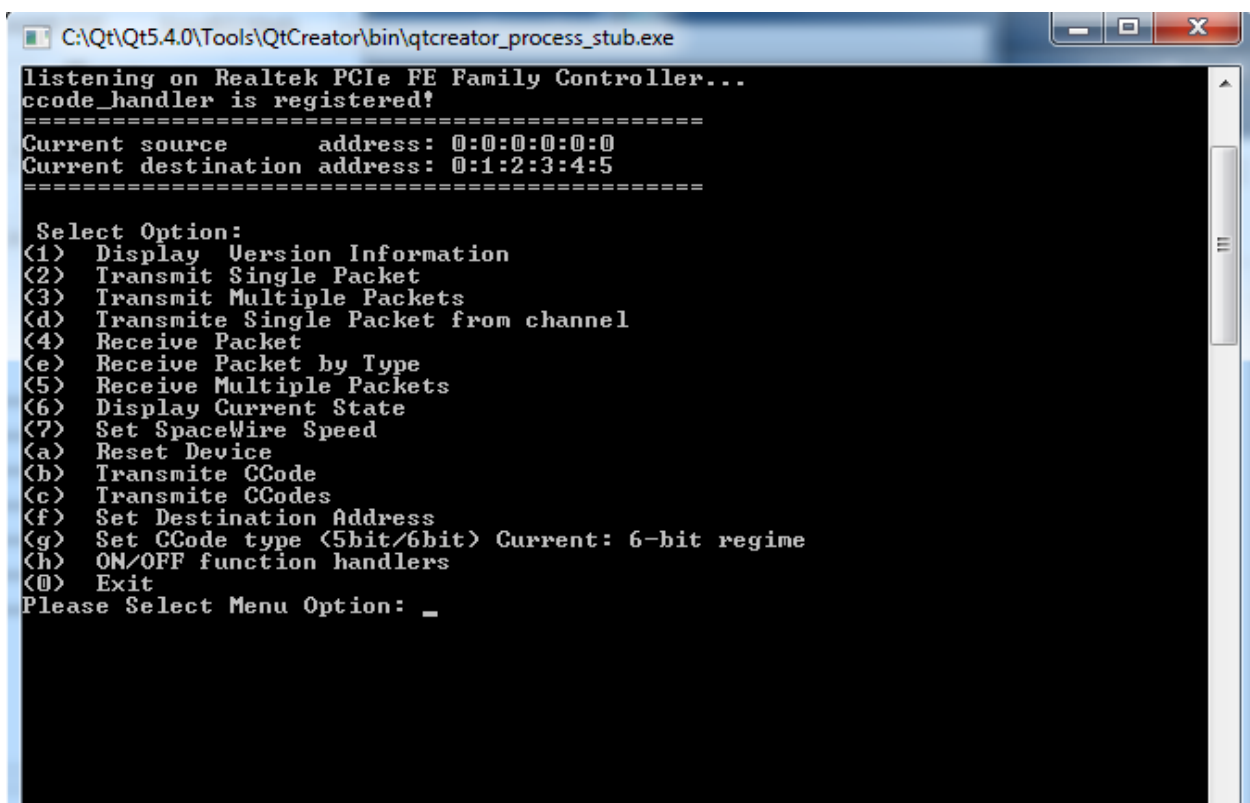


Рисунок 18 – Меню приложения 2, bridge_app

6.7.2 Приложение example_receive_packets

Приложение `example_receive_packets` служит для приема SpW-пакетов, проходящих через сетевой интерфейс Ethernet.

Для работы с приложением необходимо предварительно осуществить его сборку (см. [раздел 8.1](#)).

Настройки по умолчанию:

1. В приложении счетчик принимаемых SpW-пакетов установлен на значении 100000 (в [коде файла receive_packets.c](#) макрос `NUM_PACKETS`)
2. В приложении максимально возможный размер SpW-пакета установлен в 7500 байт (в [коде файла receive_packets.c](#) макрос `BUF_SIZE`)
3. В приложении включена обработка приходящих управляющих кодов. Обработка фреймов состояний моста и фреймов с отчетами об ошибках отключены. Прием этих типов фреймов не увеличивает счетчик принятых SpW-пакетов.

Для включения и отключения функций обработчиков событий (приход управляющего кода, приход отчета об ошибке, приход фрейма состояния моста) необходимо удалить комментарий / закомментировать в коде файла `receive_packets.c` строки регистрации функций обработчиков:

- `register_err_frame_event_handler(err_frame_recv)`
- `register_status_event_handler(status_recv)`
- `register_ccode_event_handler(ccode_recv)`

Примечание. Приложение `example_receive_packets` будет одновременно принимать фреймы с нескольких мостов в сети, если:

1. В приложении не задан MAC-адрес конкретного моста.
 2. В мостах не указано кому передаются данные.
- Подробнее см. [раздел 8.2](#).

```

C:\Qt\Qt5.4.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
1. \Device\NPF_{9C0D8B02-2575-435B-A8DD-E1EBFC8442C9} <Microsoft>
2. \Device\NPF_{E77A5932-3540-448E-9576-947BDBA3C5D1} <UMware Virtual Ethernet A
dapter>
3. \Device\NPF_{96FC1E42-9708-4F19-8365-02783AA0B4A9} <Realtek PCIe FE Family Co
ntroller>
4. \Device\NPF_{209A04B5-644B-493E-AEB5-5C1EECFB990} <UMware Virtual Ethernet A
dapter>
5. \Device\NPF_{421482B5-4313-426A-A9B2-0E511CE18CEB} <Oracle>
Enter the interface number <1-5>:3

listening on Realtek PCIe FE Family Controller...
ccode_handler is registered?
Do You want to get single packet from certain device?
Press [1] - Yes, [Any key] - No
Enter value: 1
Enter MAC of device in format:
Example: ff ff ff ff ff ff
0 1 2 3 4 5
Receiving single packet
Wait packet #0:

Recieved packet from MAC: 0:1:2:3:4:5
packet: : 101 bytes
[+0000]: 01 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB .#...l..R.I....
[+0010]: B3 A6 DB 3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE ...<...>.$^....G.
[+0020]: B3 12 4D C8 43 BB 8B A6 1F 03 5A 7D 09 38 25 1F ..M.C.....Z}.8%
[+0030]: 5D D4 CB FC 96 F5 45 3B 13 0D 89 0A 1C DB AE 32 l.....E;.....2
[+0040]: 2D 9A 50 EE 40 78 36 FD 12 49 32 F6 9E 7D 49 DC .P.6x6..I2..}I.
[+0050]: AD 4F 14 F2 44 40 66 D0 6B C4 30 B7 32 3B A1 22 .0..Def.k.0.2;."
[+0060]: F6 22 91 9D E1 .....
Wait packet #1:
>> MAC of CCode sender: 0:1:2:3:4:5
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7
>> MAC of CCode sender: 0:1:2:3:4:5
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7
  
```

Рисунок 19 –Пример работы приложения приема данных `example_receive_packets` для моста с заданным MAC-адресом

6.7.3 Отправка данных через функции API

Для отправки данных с моста API предоставляет две функции:

- **SpW_Send_Packet**
передача пакетов SpW и фреймов Ethernet заданного типа. Чтобы ознакомиться с доступными пользователю для передачи типами см. [раздел 6.7.5](#).
- **SpW_Send_Packet_From_Channel**
передача пакетов SpW и фреймов Ethernet заданного типа из заданного порта SpaceWire. Функция для работы с мостом в режиме управления каналами.

Функция отправки данных заданного типа			
Linux		Windows	
int SpW_Send_Packet(int s, char* buf, int buf_size, int type);		int SpW_Send_Packet(pcap_t *fp, unsigned char* buf, int buf_size, int type);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи
buf	указатель на адрес начала данных	buf	указатель на адрес начала данных
buf_size	размер данных	buf_size	размер данных
type	тип Ethernet фреймов	type	тип Ethernet фреймов
Возвращаемое значение: количество отправленных байт			
Пример использования функции в коде приложения bridge_app .			

Функция отправки данных из заданного SpW-порта (при двух включенных каналах в режиме управления каналами)			
Linux		Windows	
Int SpW_Send_Packet_From_Channel(int s, unsigned char* buf, int buf_size, int type, int channel);		int SpW_Send_Packet_From_Channel(pcap_t *fp, unsigned char* buf, int buf_size, int type, int chn);	
Входные параметры			
s	сокет для передачи	fp	дескриптор для передачи
buf	указатель на адрес начала данных	buf	указатель на адрес начала данных
buf_size	размер данных	buf_size	размер данных
type	тип Ethernet фреймов	type	тип Ethernet фреймов
channel	номер порта для отправки	chn	номер порта для отправки
Возвращаемое значение: количество отправленных байт			
Пример использования функции в коде приложения bridge_app.			

6.7.4 Прием данных через функции API

Для приема данных мостом API предоставляет следующие функции:

- **SpW_Recv_Packet**
прием SpW-пакетов с данными
- **SpW_Recv_Packet_From_MAC**
прием SpW-пакетов с данными от заданного MAC-адреса
- **SpW_Recv_Frame_Type**
прием Ethernet-фреймов заданного типа
- **SpW_Recv_Frame_Type_From_MAC**
прием Ethernet-фреймов заданного типа от заданного MAC-адреса
- **SpW_Recv_Packet_extended**
прием SpW-пакетов с данными, также позволяет принимать фреймы с управляющими кодами, статусом моста и отчетами об ошибках без использования функций обработчиков.
- **SpW_Recv_Packet_From_MAC_extended**
прием SpW-пакетов с данными заданного MAC-адреса, также позволяет принимать фреймы с управляющими кодами, статусом моста и отчетами об ошибках без использования функций обработчиков

Блокирующая функция приема			
Linux		Windows	
<pre>int SpW_Recv_Packet(int s, char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet);</pre>		<pre>int SpW_Recv_Packet(pcap_t *fp, unsigned char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet);</pre>	
Входные параметры			
s	сокет для приема	fp	дескриптор для приема
buf	буфер приема	buf	приемный буфер
buf_size	размер буфера	buf_size	размер буфера
mac	MAC-адрес источника (6 байт)	mac	MAC-адрес источника (6 байт)
end_packet	тип принятого пакета (0 – EOP, 1 – EEP)	end_packet	тип принятого пакета (0 – EOP, 1 – EEP)
Возвращаемое значение: размер принятого пакета.			
Параметр end_packet заполняется в процессе выполнения функции приема пакета SpaceWire.			

Блокирующая функция приема данных с заданного MAC-адреса			
Linux		Windows	
<pre>int SpW_Recv_Packet_From_MAC (int s, char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet);</pre>		<pre>int SpW_Recv_Packet_From_MAC (pcap_t *fp, unsigned char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet);</pre>	
Входные параметры			
s	сокет для приема	fp	дескриптор для приема
buf	буфер приема	buf	буфер приема
buf_size	размер буфера	buf_size	размер буфера
mac	MAC-адрес с которого ожидается прием (должен быть задан)	mac	MAC-адрес с которого ожидается прием (должен быть задан)
end_packet	тип принятого пакета (0 – EOP, 1 – EEP)	end_packet	тип принятого пакета (0 – EOP, 1 – EEP)
Возвращаемое значение: размер принятых данных			
Параметр end_packet заполняется в процессе выполнения функции приема пакета SpaceWire.			

Блокирующая функция приема Ethernet-фреймов заданного типа			
Linux		Windows	
<pre>int SpW_Recv_Frame_Type(int s, unsigned char* buf, int buf_size, unsigned mask_types, unsigned *type, unsigned char *mac);</pre>		<pre>int SpW_Recv_Frame_Type(pcap_t *fp, unsigned char* buf, int buf_size, unsigned mask_types, unsigned *type, unsigned char *mac);</pre>	
Входные параметры			
s	сокет для приема	fp	дескриптор для приема
buf	буфер приема	buf	приемный буфер
buf_size	размер буфера	buf_size	размер буфера
mask_types	маска типов фреймов	mask_types	маска типов фреймов
type	тип принятого Ethernet фрейма	type	тип принятого Ethernet фрейма

mac	MAC-адрес источника	mac	MAC-адрес источника
<p>Возвращаемое значение: размер принятых данных</p> <p>Параметр type заполняется в процессе выполнения функции приема пакета SpaceWire.</p>			
<p>Примечание. Параметр mask_types — маска типов пакетов, которые необходимо принять, макросы для заполнения маски объявлены в файле spw_eth_structure.h, макросы имеют формат Type_Mask_SpW_Eth_*. В маске может быть задано несколько типов фрейма для приема.</p>			

Блокирующая функция приема Ethernet-фреймов заданного типа с заданного MAC-адреса			
Linux		Windows	
<pre>int SpW_Recv_Frame_Type_From_MAC (int s unsigned char* buf, int buf_size, unsigned mask_types, unsigned *type, unsigned char *mac);</pre>		<pre>int SpW_Recv_Frame_Type_From_MAC (pcap_t *fp, unsigned char* buf, int buf_size, unsigned mask_types, unsigned *type, unsigned char *mac);</pre>	
Входные параметры			
s	сокет для приема	fp	дескриптор для приема
buf	буфер приема	buf	буфер приема
buf_size	размер буфера	buf_size	размер буфера
mask_types	маска типов фреймов	mask_types	маска типов фреймов
type	тип принятого Ethernet фрейма	type	тип принятого Ethernet фрейма
mac	MAC-адрес с которого ожидается прием (должен быть задан)	mac	MAC-адрес с которого ожидается прием (должен быть задан)
Возвращаемое значение: размер принятого пакета			
Параметр type заполняется в процессе выполнения функции приема пакета SpaceWire.			
Примечание. Параметр mask_types — маска типов пакетов, которые необходимо принять, макросы для заполнения маски объявлены в файле spw_eth_structure.h, макросы имеют формат Type_Mask_SpW_Eth_*. В маске может быть задано несколько типов фрейма для приема.			

Блокирующая функция приема с расширенным функционалом			
Linux		Windows	
<pre>int SpW_Recv_Packet_extended (int s, char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet, unsigned mask);</pre>		<pre>int SpW_Recv_Packet_extended (pcap_t *fp, unsigned char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet, unsigned mask);</pre>	
Входные параметры			
s	сокет для приема	fp	дескриптор для приема
buf	буфер приема	buf	приемный буфер
buf_size	размер буфера	buf_size	размер буфера
mac	MAC-адрес источника (6 байт)	mac	MAC-адрес источника (6 байт)
end_packet	тип принятого пакета (0 – EOP, 1 – EEP, SpW_Eth_CCode = 0x6, SpW_Eth_EFrame = 0x7,	end_packet	тип принятого пакета (0 – EOP, 1 – EEP, SpW_Eth_CCode = 0x6, SpW_Eth_EFrame = 0x7,

mask	SpW_Eth_Status 0x8) маска типов фреймов для приема	mask	SpW_Eth_Status 0x8) маска типов фреймов для приема
------	---	------	---

Возвращаемое значение: размер принятого пакета.

Параметр end_packet заполняется в процессе выполнения функции приема пакета SpaceWire.

Примечание. Параметр mask — маска типов фреймов, которые необходимо принять. Допустимо использование типов SpW_Eth_CCode (фреймы с управляющими кодами), SpW_Eth_EFrame (фреймы с кодами ошибок), SpW_Eth_Status (фреймы состояния моста). Данные макросы объявлены в файле spw_eth_structure.h

Блокирующая функция приема данных с заданного MAC-адреса с расширенным функционалом

Linux	Windows
<pre>int SpW_Recv_Packet_From_MAC_extended (int s, char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet, unsigned mask);</pre>	<pre>int SpW_Recv_Packet_From_MAC_extended (pcap_t *fp, unsigned char* buf, int buf_size , unsigned char* mac, unsigned char *end_packet, unsigned mask);</pre>

Входные параметры

s	сокет для приема	fp	дескриптор для приема
buf	буфер приема	buf	буфер приема
buf_size	размер буфера	buf_size	размер буфера
mac	MAC-адрес с которого ожидается прием (должен быть задан)	mac	MAC-адрес с которого ожидается прием (должен быть задан)
end_packet	тип принятого пакета (0 – EOP, 1 – EEP, SpW_Eth_CCode = 0x6, SpW_Eth_EFrame = 0x7, SpW_Eth_Status 0x8)	end_packet	тип принятого пакета (0 – EOP, 1 – EEP, SpW_Eth_CCode = 0x6, SpW_Eth_EFrame = 0x7, SpW_Eth_Status 0x8)
mask	маска типов фреймов для приема	mask	маска типов фреймов для приема

Возвращаемое значение: размер принятых данных

Параметр end_packet заполняется в процессе выполнения функции приема пакета SpaceWire.

Примечание. Параметр mask — маска типов фреймов, которые необходимо принять. Допустимо использование типов SpW_Eth_CCode (фреймы с управляющими кодами), SpW_Eth_EFrame (фреймы с кодами ошибок), SpW_Eth_Status (фреймы состояния моста). Данные макросы объявлены в файле spw_eth_structure.h.

6.7.5 Типы Ethernet-фреймов и функции-обработчики

Все типы фреймов			
Название	Описание	Значение	Доступно пользователю при передаче
SpW_Eth_MOF (MOF)	Передается промежуточный Ethernet-фрейм пакета SpW	0x00	Нет

SpW_Eth_Full_EOF (FEOF)	Передается пакет SpW целиком, который заканчивается символом конца пакета (EOP)	0x01	Да
SpW_Eth_Full_EEF (FEEF)	Передается пакет SpW целиком, который заканчивается символом конца пакета (EEF)	0x02	Да
SpW_Eth_SOF (SOF)	Передается начальный Ethernet-фрейм пакета SpW	0x03	Нет
SpW_Eth_EOF (EOF)	Передается заключительный Ethernet-фрейма пакета SpW, который заканчивается символом конца пакета EOP	0x04	Нет
SpW_Eth_EEF (EEF)	Передается заключительный Ethernet фрейм пакета SpW, который заканчивается символом конца пакета EEP	0x05	Нет
SpW_Eth_CCode	Передается Ethernet-фрейм, содержащий 1 или несколько управляющих кодов	0x06	Да
SpW_Eth_EFrame	Ethernet-фрейм с кодами ошибок (см. раздел 6.7.7), содержит информацию об ошибках	0x07	Нет
SpW_Eth_Status	Ethernet-фрейм состояния моста	0x08	Нет
SpW_Eth_Configure	Конфигурационный Ethernet-фрейм настройки моста	0xFF	Да

Для нижеперечисленных типов фреймов предусмотрен механизм функций-обработчиков:

- фреймы, содержащие коды ошибок;
- фреймы состояния моста;
- фреймы, содержащие управляющие коды.

Пример настройки функции обработки управляющих кодов

1. Инициализация сокета для работы с мостом

```
rawsock = SpW_Socket_Init("eth0"); // для Linux
rawsock = SpW_Socket_Init(); // для Windows
```

2. Регистрация функции-обработчика

```
register_ccode_event_handler(ccode_rcv);
```

3. Прототип функции

```
typedef void (*SpW_Eth_CCode_Function)(const void *buf, unsigned len,
unsigned char *MAC_rcv);
```

4. Пример функции обработки управляющих кодов

```
void ccode_rcv(const void *buf, unsigned len, unsigned char *MAC_rcv)
{
    int i;
    unsigned char val;
    for(i = 0; i < len; ++i) {
        val = ((const unsigned char *)buf)[i];
        printf(">> MAC of CCode sender: %X:%X:%X:%X:%X:%X \n",
MAC_rcv[0], MAC_rcv[1], MAC_rcv[2], MAC_rcv[3], MAC_rcv[4],
MAC_rcv[5]);
    }
}
```



```

        printf(">> CCode recieved: value %X (FULL), value (5bit) hex = %X
dec = %d\n", val, (val & 0x1f), (val & 0x1f));
    }
}

```

5. Выгрузка функции-обработчика

```
void disable_handler_ccode();
```

Пример настройки функции обработки фрейма с кодами ошибок

1. Инициализация сокета для работы с мостом

```
rawsock = SpW_Socket_Init("eth0"); // для Linux
rawsock = SpW_Socket_Init(); // для Windows

```

2. Регистрация функции

```
register_err_frame_event_handler(err_frame_recv);
```

3. Прототип функции

```
typedef void (*SpW_Eth_Err_Frame_Function)(const void *buf, unsigned len,
unsigned char *MAC_recv);
```

4. Пример функции обработки управляющих кодов

```

void err_frame_recv(const void *buf, unsigned len, unsigned char
*MAC_recv)
{
    int i;
    unsigned char frame_num, err_code;
    for(i = 0; i < len; i += 2) {
        err_code = ((const unsigned char *)buf)[i];
        frame_num = ((const unsigned char *)buf)[i + 1];
        printf(">> MAC of EFrame sender: %X:%X:%X:%X:%X:%X\n",
MAC_recv[0], MAC_recv[1], MAC_recv[2], MAC_recv[3], MAC_recv[4],
MAC_recv[5]);
        printf(">> Error frame recieved: frame_num = %d , err_code %d\n",
frame_num ,err_code);
        print_error_type(err_code);
    }
}

```

5. Выгрузка функции-обработчика

```
void disable_handler_err_frame();
```

Пример настройки функции обработки фрейма состояния моста

1. Инициализация сокета для работы с мостом

```
rawsock = SpW_Socket_Init("eth0"); // для Linux
rawsock = SpW_Socket_Init(); // для Windows

```

2. Регистрация функции

```
register_status_event_handler(status_recv);
```

3. Прототип функции

```
typedef void (*SpW_Eth_Status_Function)(const void *buf, unsigned len,
unsigned char *MAC_recv);
```

4. Пример функции обработки фрейма состояния моста

```
void status_recv(const void *buf, unsigned len, unsigned char *MAC_recv)
{
    printf(">> MAC of Status sender: %X:%X:%X:%X:%X:%X", MAC_recv[0],
MAC_recv[1], MAC_recv[2], MAC_recv[3], MAC_recv[4], MAC_recv[5]);
    print_spw_eth_state((struct spw_eth_state_new *)buf);
}
```

5. Выгрузка функции-обработчика

```
void disable_handler_status();
```

6.7.6 Взаимодействие функций-обработчиков и функций приема данных

В случаях, когда функции-обработчики событий инициализированы, и запущена функция приема, то до момента приема SpW-пакета данных все происходящие события (прием фрейма состояния моста, управляющих кодов и отчета об ошибках) будут обрабатываться, и результаты обработки будут отображаться в консоли. В приложениях предоставлены примерные варианты обработки событий, их можно изменять и подстраивать их работу под свои нужды. Также в SDK-библиотеке созданы функции для просмотра статуса инициализации функции-обработчика и удаления инициализированных функций-обработчиков.

Для проверки, инициализирована ли функция-обработчик, используются следующие функции:

```
SpW_Eth_Status_Function get_handler_status_state();
SpW_Eth_Err_Frame_Function get_handler_err_frame_state();
SpW_Eth_CCode_Function get_handler_ccode_state();
```

Если функции не инициализированы, они возвращают NULL.

Пример использования см. в [коде spw_eth_test.c](#).

Для отключения работы функций-обработчиков необходимо вызвать следующие функции:

```
void disable_handler_status();
void disable_handler_err_frame();
void disable_handler_ccode();
```

Эти функции не имеют возвращаемого значения. Пример использования см. в [коде spw_eth_test.c](#).

6.7.7 Фрейм с кодами ошибок

Структура фрейма отчета об ошибках

Отчет об ошибке, произошедшей при передаче данных, состоит из двух однобайтовых полей. Первый байт означает номер ошибки, второй байт обозначает номер фрейма, при передаче которого произошла ошибка.

Типы ошибок и предупреждений

Номер ошибки	Описание ошибки и предупреждение	Обработка ошибки
0x0	Информация о кредитовании.	В случае отсутствия свободных буферов кредитования возможно отбрасывание Ethernet-фреймов.
0x1	Пришедший SOF, FEEF, FEOF фрейм данных имеет номер на 2 и более больше, чем предыдущий. Предыдущий фрейм данных был SOF или MOF.	Генерируется событие – «потерян фрейм данных», генерируется код EEP, который отправляется в сеть SpW, после этого передаются принятые данные из фрейма.
0x2	Пришедший SOF, FEEF, FEOF фрейм данных имеет номер на 2 и более больше, чем предыдущий. Предыдущий фрейм данных был EOF, EEF, FEOF или FEEF.	Генерируется событие – «потерян фрейм данных», передаются данные из фрейма.
0x3	Пришедший MOF, EOF или EEF фрейм данных имеет номер на 2 и более больше, чем предыдущий. Предыдущий фрейм данных был MOF или SOF.	Генерируется событие – «потерян фрейм данных», генерируется код EEP, который отправляется в сеть SpW, принятые данные из фрейма передаются, после отправки последнего байта конечного фрейма (EOF или EEF) отправляется EEP или текущий фрейм, и последующие фреймы данных до конечного фрейма (включительно) отбрасываются.
0x4	Пришедший MOF, EOF или EEF фрейм данных имеет номер на 2 и более больше, чем предыдущий. Предыдущий фрейм данных был EOF, EEF, FEOF или FEEF.	Генерируется событие – «потерян фрейм данных», принятые данные из фрейма передаются, после отправки последнего байта конечного фрейма (EOF или EEF) отправляется EEP или текущий фрейм, и последующие фреймы данных до конечного фрейма (включительно) отбрасываются.
0x5	Пришедший фрейм управляющего кода на 2 и более больше, чем предыдущий фрейм управляющих кодов.	Генерируется событие «потерян фрейм управляющих кодов», пришедший фрейм отправляется в сеть.

Ситуации появления ошибок при передаче фреймов

Приход отчета об ошибках возникает в следующих ситуациях:

1. **Ситуация 1.** Отчет об ошибке придет из-за разных номеров последовательности фреймов в приложении и на самом мосту. Возникновение ошибки и приход отчета об ошибке не влияет на факт передачи данных.
 1. Запуск моста
 2. Запуск приложения
 3. Передача данных (управляющих кодов)
 4. Завершение работы приложения
 5. Запуск приложения
 6. Передача данных (управляющих кодов)
 7. Приход фрейма с отчетом об ошибке. При передаче данных код ошибки будет равен 2. При отправке управляющих кодов код ошибки будет равен 5.
2. **Ситуация 2.** Отчет об ошибке придет из-за разных номеров последовательности фреймов в приложении и на самом мосту. Возникновение ошибки и приход отчета об ошибке не влияет на факт передачи данных.
 1. Запуск моста
 2. Запуск приложения 1
 3. Запуск приложения 2
 4. Передача данных (управляющих кодов) из приложения 1
 5. Передача данных (управляющих кодов) из приложения 2
 6. Приход фрейма с отчетом об ошибке. При передаче данных код ошибки будет равен 2. При отправке управляющих кодов код ошибки будет равен 5.
3. **Ситуация 3.** При передаче пакета SpaceWire, превышающего по размеру максимальный размер фрейма Ethernet, пакет SpaceWire будет разбиваться на несколько фреймов Ethernet. В случае потери промежуточного фрейма Ethernet при передаче будет сгенерирован отчет об ошибке. Эта ситуация будет обработана согласно данным в таблице (коды ошибок и их обработка, см. [раздел 6.7.7](#)).

6.7.8 Прием и передача управляющих кодов

Прием управляющих кодов осуществляется в функциях приема данных. Значение управляющего кода содержит в себе поле типа управляющего кода и значение управляющего кода.

Структура управляющих кодов:

- Для 6-битных управляющих кодов

Тип		Значение						
0	0	V5	V4	V3	V2	V1	V0	Маркер времени
0	1	V5	V4	V3	V2	V1	V0	Код прерывания
1	0	V5	V4	V3	V2	V1	V0	Код подтверждения
1	1	V5	V4	V3	V2	V1	V0	Код CC11

- Для 5-битных управляющих кодов

Тип			Значение					
0	0	0	V4	V3	V2	V1	V0	Маркер времени
1	0	0	V4	V3	V2	V1	V0	Код прерывания
1	0	1	V4	V3	V2	V1	V0	Код подтверждения
1	1	0	V4	V3	V2	V1	V0	Код CC11
0	1	0	V4	V3	V2	V1	V0	Код CC01

Для обработки принятых управляющих кодов необходимо создать и зарегистрировать **функцию-обработчики управляющих кодов** ([пример](#)).

Передача управляющих кодов производится с помощью специальных функций библиотеки моста (**SpW_Send_Time_Code**, **SpW_Send_ACK_Code**, **SpW_Send_INT_Code**, **SpW_Send_CC01_Code**, **SpW_Send_CC11_Code** а также с помощью аналогичных функций для передачи нескольких управляющих кодов в одном фрейме). Пример использования данных функций можно найти в исходных кодах программы `bridge_app.c`.

6.8 Работа с несколькими мостами Ethernet-SpaceWire через коммутатор Ethernet

Инициализация и настройка

Каждый мост инициализируется и настраивается по отдельности.

1. Если имеются подключенные мосты, то следует отключить все мосты от сети Ethernet.
Важно! Не отключайте мосты от питания!
В противном случае настройки сбросятся, и мосты придется настраивать заново.
2. Подключить новый мост к питанию и сети Ethernet.
3. Провести инициализацию (см. [раздел 6.4](#)) и настройку моста (см. [раздел 6.6](#)).

4. Подключить мосты, отключенные на п.1, к сети Ethernet.

Основная работа

В случае работы через приложения (см. [раздел 8](#)) необходимо запустить для каждого моста свой экземпляр приложения. Например, один экземпляр приложения будет взаимодействовать с мостом 1, другой – с мостом 2.

7 Завершение и перезагрузка работы с мостом

Функция завершения работы с библиотекой моста			
Linux		Windows	
void SpW_Socket_Close(int s);		int SpW_Socket_Close(pcap_t *fp);	
Входные параметры			
s	сокет для приема/передачи	fp	дескриптор для приема/передачи данных

Функция перезагрузки работы с библиотекой моста			
Linux		Windows	
int SpW_Socket_Reset (char *device, int s);		pcap_t* SpW_Socket_Reset (pcap_t *fp);	
Входные параметры			
device	имя сетевого устройства Ethernet в Linux (к примеру, eth0)	fp	текущий дескриптор для приема/передачи данных
s	текущий сокет для приема/передачи		
Возвращаемое значение			
новый сокет для приема/передачи		новый дескриптор для приема/передачи данных	

ВАЖНО! 1. Функции сбрасывают все зарегистрированные ранее функции обработки.
 2. Функции сбрасывают значения счетчиков последовательности пакетов (они становятся равными 0)

8 Приложения для работы с мостом

Для удобства работы с мостом предоставляются три консольных приложения:

1. Приложение 1 для настройки моста: `configure_bridge` (см. [раздел 6.6.2](#)).
2. Приложение 2 для работы с мостом: `bridge_app` (см. [раздел 6.7.1](#)).
3. Приложение 3 для приема пакетов: `example_receive_packets` (см. [раздел 6.7.2](#)).

Приложения предоставляются в виде исходных файлов. Для работы с приложениями необходимо осуществить их сборку (см. [раздел 8.1](#)).

ВАЖНО! 1. В операционных системах семейства **Linux** приложения необходимо запускать с правами суперпользователя.

2. В операционных системах семейства **Linux** при запуске приложений в качестве аргумента можно задать параметр, характеризующий имя сетевого устройства, к которому подключен мост. В случае если аргумент при запуске приложения не был задан, в консоли будут выведены имена всех доступных сетевых устройств ПК, из которых нужно будет выбрать правильное имя сетевого устройства.

Пример запуска приложений с параметром:

```
sudo ./bridge_app eth0
```

```
sudo ./configure_bridge enp7s0
```

3. В используемой ОС должна быть поддержка утилиты **ip**. Данная утилита имеется в составе пакета **iproute2** – набора утилит для управления параметрами сетевых устройств в ядре Linux: <https://git.kernel.org/pub/scm/network/iproute2/iproute2.git/tag/?h=v4.18.0>. Обычно пакет утилит устанавливается командой установки, например: **sudo apt-get install iproute2** или **sudo yum install iproute2**, в зависимости от ОС.

Исходники приложений также служат примером использования предоставляемого API.

8.1 Сборка приложений для работы с мостом

Сборку приложений необходимо осуществить на компьютере один раз. Далее собранные приложения запускаются по мере необходимости.

- Существует два варианта сборки приложений на ОС семейства **Linux**
 1. в каталоге `demo` выполнить команду `make`. В результате одновременно осуществляется сборка всех трех приложений.
 2. Использовать QtCreator.
 1. Открыть файл проекта необходимого приложения (*.pro):
 - Для приложения `configure_bridge` – **configure_bridge.pro**
 - Для приложения `bridge_app` – **bridge_app.pro**
 - Для приложения `example_receive_packets` – **example_receive_packets.pro**
 2. Произвести компиляцию проекта.

- Сборка приложений на ОС **Windows**:
 1. Использовать QtCreator. Открыть файл проекта необходимого приложения (*.pro):

- Для приложения `configure_bridge` – **`configure_bridge.pro`**
- Для приложения `bridge_app` – **`bridge_app.pro`**
- Для приложения `example_receive_packets` – **`example_receive_packets.pro`**

ВАЖНО! В случае использования других сред разработки необходимо:

1. Создать проект консольного приложения на языке Си.
 2. Осуществить компоновку библиотек: SDK моста и WinPcap.
 3. Подключить заголовочные файлы SDK в коде написанных приложений.
 4. Использовать файлы кода (*.c) приложения.
2. Произвести компиляцию проекта.

8.2 Отображение принимаемых пакетов в приложениях

При использовании функций приема из сети Ethernet необходимо учитывать, что если используются функции, не обеспечивающие фильтрацию принимаемых данных по MAC-адресу, то функции будут также принимать пакеты данных SpW, отправляемые с ПК на мост.

Для наглядности рассмотрим пример работы моста.

Схема подключения и настройка моста.

1. Мост SpaceWire-Ethernet подключен к ПК.
2. Порты SpaceWire моста соединены между собой кабелем.
3. Режим передачи данных: 4.
4. Режим передачи управляющих кодов: 3.
5. На ПК запускаются: `bridge_app` для отправки данных, и дважды `example_receive_packets` для приема данных.
6. В первом экземпляре приложения `example_receive_packets` фильтрация принимаемых данных по MAC-адресу отключена, во втором экземпляре фильтрация по MAC-адресу включена. По умолчанию MAC-адрес моста 00:01:02:03:04:05.

Отправляемые данные

В `bridge_app` следует выбирать отправку одиночного SpW-пакета (пункт меню №2) длиной в 100 байт.

Для корректной отправки, учитывая, что на мосту режим отправки данных равен 4, и соединение есть на двух портах SpaceWire, следует указать размер путевого адреса равный 1, а его значение – равное 0. Благодаря этому пакет SpaceWire будет отправлен из

порта SpW1 в порт SpW2. Этот путь адрес будет отброшен при отправке пакета из порта SpW1 моста, а при получении пакета из SpW2 к пакету будет добавлен 1 байт заголовка со значением 1, как признак приема пакета из порта SpW2.

Далее производится отправка одного управляющего кода (в нашем случае это код прерывания) со значением 7. В текущем примере режим отправки управляющих кодов установлен в значение 3, т.е. прием и отправка управляющих кодов производится по двум портам SpaceWire.

```

Please Select Menu Option: 2
Transmit_SinglePacket
Enter packet size: 100
Enter path len: 1
path[0]= 0
Transmit packet:
: 101 bytes
[+0000]: 00 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB .#...l..R.I.....
[+0010]: B3 A6 DB 3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE ...<...>.$^.....G.
[+0020]: B3 12 4D C8 43 BB 8B A6 1F 03 5A 7D 09 38 25 1F ...M.C.....Z>.8%
[+0030]: 5D D4 CB FC 96 F5 45 3B 13 0D 89 0A 1C DB AE 32 l.....E;.....2
[+0040]: 20 9A 50 EE 40 78 36 FD 12 49 32 F6 9E 7D 49 DC .P.Qx6..I2..>I.
[+0050]: AD 4F 14 F2 44 40 66 D0 6B C4 30 B7 32 3B A1 22 .O..Dcf.k.Q.2;."
[+0060]: F6 22 91 9D E1 .....
=====
Current source address: 0:0:0:0:0:0
Current destination address: 0:1:2:3:4:5
=====

Select Option:
<1> Display Version Information
<2> Transmit Single Packet
<3> Transmit Multiple Packets
<d> Transmite Single Packet from channel
<4> Receive Packet
<e> Receive Packet by Type
<5> Receive Multiple Packets
<6> Display Current State
<7> Set SpaceWire Speed
<a> Reset Device
<b> Transmite CCode
<c> Transmite CCodes
<f> Set Destination Address
<g> Set CCode type <5bit/6bit> Current: 6-bit regime
<h> ON/OFF function handlers
<0> Exit
Please Select Menu Option: b
Transmite_CCode
Enter type of CCode <0-time , 1-int, 2-ack> or Exit - 5: 1
Enter INT code value: ?
=====
Current source address: 0:0:0:0:0:0
Current destination address: 0:1:2:3:4:5
=====

```

Рисунок 20 – Задание отправляемых данных и управляющих кодов в приложении bridge_app

Прием данных без фильтрации по MAC-адресу

При работе example_receive_packets без настроенной фильтрации по MAC-адресу будут приняты следующие данные:

1. Пакет данных длиной в 101 байт, который отправлялся с ПК на мост с путь адресом длиной в 1 байт и значением 0 (признак того, что пакет данных должен отправиться через порт SpW1). MAC-адрес источника пакета: 00:00:00:00:00:00 ([MAC-адрес в ОС Windows по умолчанию](#)), что является MAC-адресом ПК в нашем случае.
2. Пакет данных длиной в 101 байт, который был принят на ПК, с путь адресом длиной в 1 байт и значением 1 (признак того, что пакет данных пришел на мост через порт SpW2). MAC-адрес источника пакета: 00:01:02:03:04:05, что является MAC-адресом моста по умолчанию.

3. Управляющий код (код прерывания), который отправлялся с ПК. MAC-адрес источника пакета с управляющим кодом: 00:00:00:00:00:00, что является MAC-адресом ПК в нашем случае.
4. Управляющий код (код прерывания), который был принят через порт SpW1. MAC-адрес источника пакета с управляющим кодом: 00:01:02:03:04:05, что является MAC-адресом моста по умолчанию.
5. Управляющий код (код прерывания), который был принят через порт SpW2. MAC-адрес источника пакета с управляющим кодом: 00:01:02:03:04:05, что является MAC-адресом моста по умолчанию.

```

C:\Qt\Qt5.4.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
1. \Device\NPF_{9C0D8B02-2575-435B-A8DD-E1EBFC8442C9} <Microsoft>
2. \Device\NPF_{E77A5932-3540-448E-9576-947BDBA3C5D1} <UMware Virtual Ethernet A
dapter>
3. \Device\NPF_{96FC1E42-9708-4F19-8365-02783AA0B4A9} <Realtek PCIe FE Family Co
ntroller>
4. \Device\NPF_{209A04B5-644B-493E-AEB5-5C1EECFB990} <UMware Virtual Ethernet A
dapter>
5. \Device\NPF_{421482B5-4313-426A-A9B2-0E511CE18CEB} <Oracle>
Enter the interface number <1-5>:3

listening on Realtek PCIe FE Family Controller...
ccode_handler is registered!
Do You want to get single packet from certain device?
Press [1] - Yes, [Any key] - No
Enter value: 3
Receiving single packet from any device
Wait packet #0:

Recieved packet from MAC: 0:0:0:0:0:0
packet: : 101 bytes
[+0000]: 00 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB .#...l..R.I.....
[+0010]: B3 A6 DB 3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE ...<...>.$^....G.
[+0020]: B3 12 4D C8 43 BB 8B A6 1F 03 5A 7D 09 38 25 1F ..M.C.....Z}.8%.
[+0030]: 5D D4 CB FC 96 F5 45 3B 13 0D 89 0A 1C DB AE 32 1.....E;.....2
[+0040]: 20 9A 50 EE 40 78 36 FD 12 49 32 F6 9E 7D 49 DC .P.ex6..I2...I.
[+0050]: AD 4F 14 F2 44 40 66 D0 6B C4 30 B7 32 3B A1 22 .0..Def.k.0.2;."
[+0060]: F6 22 91 9D E1 ."...
Wait packet #1:

Recieved packet from MAC: 0:1:2:3:4:5
packet: : 101 bytes
[+0000]: 01 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB .#...l..R.I.....
[+0010]: B3 A6 DB 3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE ...<...>.$^....G.
[+0020]: B3 12 4D C8 43 BB 8B A6 1F 03 5A 7D 09 38 25 1F ..M.C.....Z}.8%.
[+0030]: 5D D4 CB FC 96 F5 45 3B 13 0D 89 0A 1C DB AE 32 1.....E;.....2
[+0040]: 20 9A 50 EE 40 78 36 FD 12 49 32 F6 9E 7D 49 DC .P.ex6..I2...I.
[+0050]: AD 4F 14 F2 44 40 66 D0 6B C4 30 B7 32 3B A1 22 .0..Def.k.0.2;."
[+0060]: F6 22 91 9D E1 ."...
Wait packet #2:
>> MAC of CCode sender: 0:0:0:0:0:0
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7
>> MAC of CCode sender: 0:1:2:3:4:5
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7
>> MAC of CCode sender: 0:1:2:3:4:5
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7

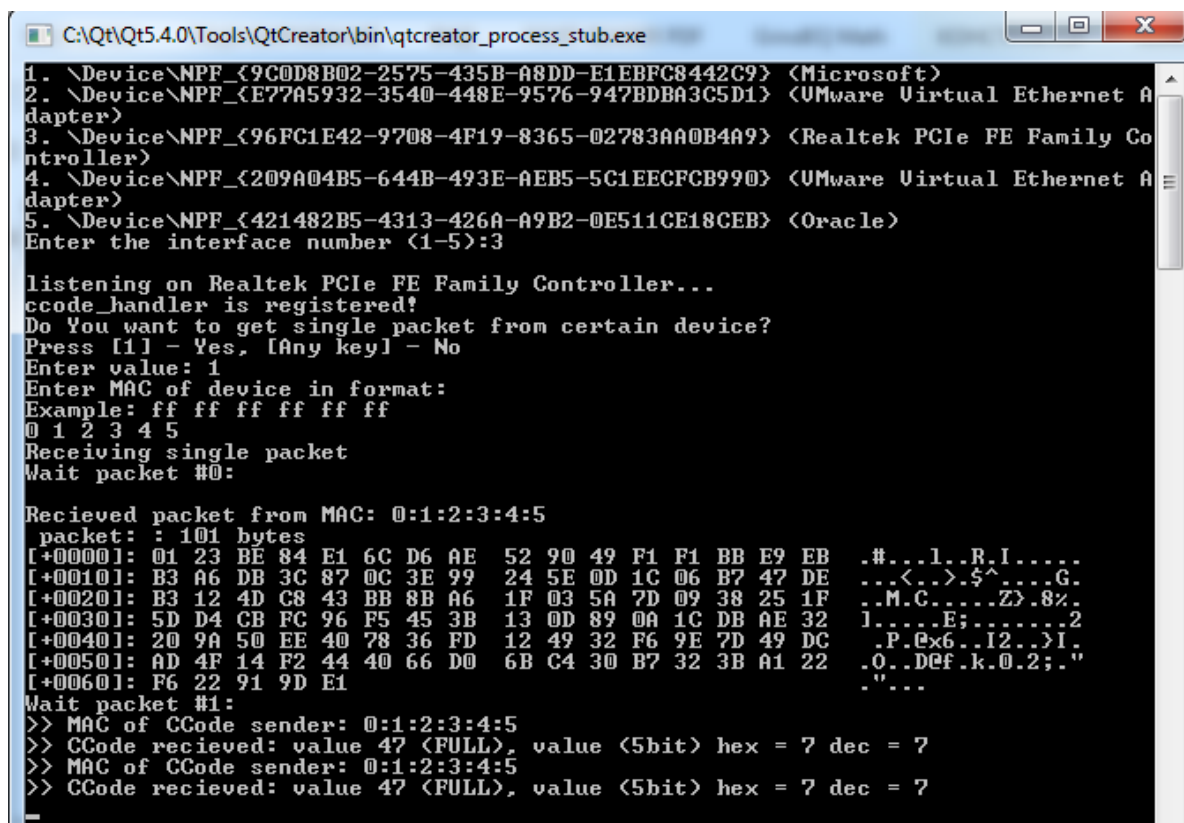
```

Рисунок 21 – Принимаемые данные и управляющие коды без фильтрации по MAC-адресу

Принимаемые данные с фильтрацией по MAC адресу

При работе приложения 3 с настроенной фильтрацией по MAC-адресу будут приняты только пакеты, пришедшие с портов SpaceWire на мост. Будут приняты следующие данные:

1. Пакет данных длиной в 101 байт, который был принят на ПК с путевым адресом длиной в 1 байт и значением 1. (признак того, что пакет данных пришел на мост через порт SpW2). MAC-адрес источника пакета: 00:01:02:03:04:05, что является MAC-адресом моста по умолчанию.
2. Управляющий код (код прерывания), который был принят через порт SpW1. MAC-адрес источника пакета с управляющим кодом: 00:01:02:03:04:05, что является MAC-адресом моста по умолчанию.
3. Управляющий код (код прерывания), который был принят через порт SpW2. MAC-адрес источника пакета с управляющим кодом: 00:01:02:03:04:05, что является MAC-адресом моста по умолчанию.



```
C:\Qt\Qt5.4.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
1. \Device\NPF_{9C0D8B02-2575-435B-A8DD-E1EBFC8442C9} <Microsoft>
2. \Device\NPF_{E77A5932-3540-448E-9576-947BDBA3C5D1} <UMware Virtual Ethernet A
dapter>
3. \Device\NPF_{96FC1E42-9708-4F19-8365-02783AA0B4A9} <Realtek PCIe FE Family Co
ntroller>
4. \Device\NPF_{209A04B5-644B-493E-AEB5-5C1EECFB990} <UMware Virtual Ethernet A
dapter>
5. \Device\NPF_{421482B5-4313-426A-A9B2-0E511CE18CEB} <Oracle>
Enter the interface number <1-5>:3

listening on Realtek PCIe FE Family Controller...
ccode_handler is registered?
Do You want to get single packet from certain device?
Press [1] - Yes, [Any key] - No
Enter value: 1
Enter MAC of device in format:
Example: ff ff ff ff ff ff
0 1 2 3 4 5
Receiving single packet
Wait packet #0:

Recieved packet from MAC: 0:1:2:3:4:5
packet: : 101 bytes
[+0000]: 01 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB .#...l..R.I.....
[+0010]: B3 A6 DB 3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE ..<..>.$^....G.
[+0020]: B3 12 4D C8 43 BB 8B A6 1F 03 5A 7D 09 38 25 1F .M.C.....Z>.8%.
[+0030]: 5D D4 CB FC 96 F5 45 3B 13 0D 89 0A 1C DB AE 32 l.....E;.....2
[+0040]: 20 9A 50 EE 40 78 36 FD 12 49 32 F6 9E 7D 49 DC .P.0x6..I2..>I.
[+0050]: AD 4F 14 F2 44 40 66 D0 6B C4 30 B7 32 3B A1 22 .O..Def.k.0.2;."
[+0060]: F6 22 91 9D E1 ....."
Wait packet #1:
>> MAC of CCode sender: 0:1:2:3:4:5
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7
>> MAC of CCode sender: 0:1:2:3:4:5
>> CCode recieved: value 47 <FULL>, value <5bit> hex = 7 dec = 7
```

Рисунок 22 – Принимаемые данные и управляющие коды с фильтрацией по MAC адресу

9 Создание пользовательских приложений для работы с мостом

Любое пользовательское приложение, где применяются функции API, должно использовать:

1. Заголовочные файлы
 - [spw_eth.h](#) – прототипы функций API
 - [spw_eth_print.h](#) – прототипы вспомогательных функций вывода
 - [spw_eth_structure.h](#) – макросы и структуры данных API
2. Библиотека SDK (ее необходимо подключить к пользовательской среде разработки)
 - 2.1 Linux
[libspw_eth_unix.a](#)
 - 2.2 Windows
[libspw_eth_win.a](#)
3. Для Windows подключить файлы библиотеки **SDK WinPcap**
4. Файл(ы) исходных кодов пользовательской программы

ВАЖНО! В операционных системах типа Linux приложения необходимо запускать с правами суперпользователя.

Примечание. Библиотека моста содержит дополнительный функционал, который можно использовать при создании пользовательских приложений. Описание вспомогательного функционала библиотеки находится в файле [spw_eth.h](#).