УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2023 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429 – ИНТЕРФЕЙС СЕТЕВЫХ СОКЕТОВ»

Модулей
“LAN–ARINC429UDxx”

ОС WINDOWS, ОС Linux (Astra Linux)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.15106-01 33 01-ЛУ

От	Инженер-программист
_____	_____ М.С. Ступаков
«____»_____2023 г.	«____»_____2023 г.

2023

Из	*Под*	*Дат*

Литера

Подп. и	
Инв. № дубл	
Взам. инв. №	
Подп. и	
Инв. № подл	

Утвержден

RU.МСКЮ.15106-01 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429 – ИНТЕРФЕЙС СЕТЕВЫХ СОКЕТОВ»

Модулей
“LAN–ARINC429UDxx”

ОС WINDOWS, ОС Linux (Astra Linux)

Руководство программиста

RU.МСКЮ.15106-01 33 01

Листов - 43

2023

Инв. № подл	Подп. и	Взам. инв. №	Инв. № дубл	Подп. и

Из	Под	Дат	

Литера

АННОТАЦИЯ

В книге описываются технологические принципы, использованные в программном обеспечении «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429 – ИНТЕРФЕЙС СЕТЕВЫХ СОКЕТОВ». В частности, рассмотрены функциональное назначение и область применения, условия выполнения.

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ

ПО – программное обеспечение;

РК – разовая команда;

1 НАЗНАЧЕНИЕ ПРОГРАММЫ

1.1 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

Программное обеспечение «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429 – ИНТЕРФЕЙС СЕТЕВЫХ СОКЕТОВ» (далее – библиотека) обеспечивает вспомогательный сервисный функционал при взаимодействии с модулями: «LAN–429UDxx» (далее «xxx-429UDxx»).

Библиотека обеспечивает выполнение следующих основных задач:

– реализация сервисных функций поканально.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

Библиотека взаимодействия предназначена для функционирования в (ОС Linux (Astra Linux), а также Windows 7/10) и встраивания в прикладное ПО для инициализации и управления модулями «xxx-429UDxx».

Библиотека реализует свои функции посредством обращения к внешнему устройству через сетевые сокеты. Это, в свою очередь, требует наличия в ОС установленного и настроенного адаптера Ethernet, с доступным сетевым маршрутом до подключенного устройства.

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429

Библиотека взаимодействия разработана на языке С.

Для использования библиотеки в проекте необходимо в каталоге библиотеки выполнить команду «make», в результате которой появится файл exta429lib.so и подключить к целевому проекту полученный файл и заголовочные фалы библиотеки.

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429

Библиотека предназначена для функционирования в ОС Windows и ОС Linux (Astra Linux) и встраивания в прикладное ПО. В файле «a429ext_library.h» представлены сервисные функции библиотеки взаимодействия.

4.1.1 Открыть канал – exta429_open

Функция возвращает структуру сетевого дескриптора, используемую в дальнейшем для обращения к устройству. При наличии ошибки соединения, поле дескриптор сокета будет заполнено значением «-1». Описание функции приведено на рисунке 4.1.1.

```
#ifdef _WIN32

///
/// \brief Дескриптор канала внешнего устройства
///
typedef struct
{
    SOCKET socketFD; ///< дескриптор сокета
    unsigned int channel; ///< канал внешнего устройства
} A429ChannelFD;

#else

///
/// \brief Дескриптор канала внешнего устройства
///
typedef struct
{
    int socketFD; ///< дескриптор сокета
    unsigned int channel; ///< канал внешнего устройства
} A429ChannelFD;

#endif

/// \brief подключиться к каналу связи
/// \param ipv4_addr адрес устройства (строка, например "192.168.10.10")
/// \param port порт устройства
/// \param channel a429-канал устройства
/// \return дескриптор открытого канала (проверьте, что socketFd не -1)
A429EXT_API A429ChannelFD exta429_open(const char *address, uint16_t port,
uint16_t channel);
```

Рисунок 4.1.1– Листинг функции

4.1.2 Закрыть канал – exta429_close

Описание функции приведено на рисунке 4.1.2.

```
/// \brief закрыть канал связи
A429EXT_API int exta429_close(A429ChannelFD cfd);
```

Рисунок 4.1.2– Листинг функции

4.1.3 Получить инфо о плате – exta429_ getDeviceInfo

Описание функции приведено на рисунке 4.2.4.

```
/// \brief exta429_getDeviceInfo получить информацию о канале и плате
/// \param fd дескриптор канала
/// \param info инфо о канале и плате
/// \return результат операции
///
A429EXT_API unsigned int exta429_getDeviceInfo(A429ChannelFD cfd, VERSION* info);

/// \brief информация о драйвере и устройстве
typedef struct {
    unsigned int device_id;     ///< идентификатор устройства
    unsigned int vendor_id;     ///< вендор устройстваf
    unsigned int type;          ///< тип устройства (кол-во каналов)
    char revision;              ///< ревизия устройства
    char dev_name[30];          ///< имя символьного устройства
    int minor;                  ///< значение минора
    int irq;                    ///< номер прерывания
    long size_dma;              ///< размер ДМА буфера
    void* addr_dma_virt;        ///< виртуальный адрес ДМА буфера
    unsigned int pci_bars;      ///< адрес bar-пространства
    void* addr_bar_virt;        ///< виртуальный адрес bar-пространства
} VERSION;
```

Рисунок 4.2.4– Листинг функции

4.1.4 Получить версию драйвера – exta429_ getDriverVersion

Описание функции приведено на рисунке 4.2.5.

```
/// \brief exta429_getDriverVersion получить версию драйверу
/// \param fd дескриптор канала
/// \param version версия драйвера
/// \return результат операции
///
A429EXT_API unsigned int exta429_getDriverVersion(A429ChannelFD cfd, unsigned int*
version);
```

Рисунок 4.2.5– Листинг функции

4.1.5 Получить инфо о канале – exta429_ getChannelInfo

Описание функции приведено на рисунке 4.2.7.

```
/// \brief exta429_getChannelInfo получить информацию о канале
/// \param fd дескриптор канала
/// \param chInfo информация о канале
/// \return результат операции
///
A429EXT_API unsigned int exta429_getChannelInfo(A429ChannelFD cfd, T_INFO_CHANNEL*
chInfo);

#define T_INFO_CHANNEL_TYPE_RECIEVER        0
#define T_INFO_CHANNEL_TYPE_TRANSMITTER     1
#define T_INFO_CHANNEL_BAD_VALUE            -1

/// \brief The T_INFO_CHANNEL struct информация о канале
typedef struct {
    int typeChannel;        /// \brief тип канала (0 - приёмник, 1 - передатчик)
    int numberChannel;      /// \brief номер приёмника/передатчика (с 0)
} T_INFO_CHANNEL;
```

Рисунок 4.2.7– Листинг функции

4.1.6 Получить инфо о каналах платы – exta429_ getDevInfoChannels

Описание функции приведено на рисунке 4.2.8.

```
/// \brief exta429_getDevInfoChannels получить информацию о каналах девайса
/// \param fd дескриптор канала
/// \param devInfoCh информация о каналах девайса
/// \return результат операции
///
A429EXT_API unsigned int exta429_getDevInfoChannels(A429ChannelFD cfd,
T_INFO_DEVICE* devInfoCh);

/// \brief информация об плате
typedef struct {
    int countRecieverChannels;          ///< кол-во приёмников
    int countTransmitterChannels;       ///< кол-во передатчиков
} T_INFO_DEVICE;
```

Рисунок 4.2.8– Листинг функции

4.1.7 Записать регистр – exta429_ writeReg

Описание функции приведено на рисунке 4.2.9.

```
/// \brief exta429_writeReg записать значение регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
A429EXT_API unsigned int exta429_writeReg(A429ChannelFD cfd, SADDR_DATA* reg);

/// \brief запись/чтение регистров
typedef struct {
    unsigned long daddr;        ///< адрес регистра (смещение в соотв. со
спецификацией)
    unsigned int data;          ///< значение регистра
} SADDR_DATA;
```

Рисунок 4.2.9– Листинг функции

4.1.8 Прочитать регистр – exta429_ readReg

Описание функции приведено на рисунке 4.2.10.

```
/// \brief exta429_readReg прочитать значение регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
A429EXT_API unsigned int exta429_readReg(A429ChannelFD cfd, SADDR_DATA* reg);

/// \brief запись/чтение регистров
typedef struct {
    unsigned long daddr;        ///< адрес регистра (смещение в соотв. со
спецификацией)
    unsigned int data;          ///< значение регистра
} SADDR_DATA;
```

Рисунок 4.2.10– Листинг функции

4.1.9 Записать заданные биты регистра – exta429_ modifyReg

Описание функции приведено на рисунке 4.2.11.

```
/// \brief exta429_modifyReg изменить заданные биты регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
A429EXT_API unsigned int exta429_modifyReg(A429ChannelFD cfd, SADDR_DATA_BIT_MASK*
reg);

typedef struct {
    unsigned long daddr;        ///< адрес регистра (смещение в соотв. со
спецификацией)
    unsigned int data;          ///< значение регистр
    unsigned int mask;          ///< битовая маска (1 - бит записывается из data,
0 - бит остаётся неизменным)
} SADDR_DATA_BIT_MASK;
```

Рисунок 4.2.11– Листинг функции

4.1.10 Прочитать ДМА – exta429_ readDma

Описание функции приведено на рисунке 4.2.12.

```
/// \brief exta429_readDma чтение доступных данных из дма
/// (но не более 256 блоков за раз)
/// \param fd дескриптор канала
/// \param container контейнер данных дма
/// \return результат операции
///
A429EXT_API unsigned int exta429_readDma(A429ChannelFD cfd, T_CONTAINER_DMA*
container);

/// \brief максимальное кол-во блоков дма в контейнере
#define DMA_COUNT_BLOCKS                    256
/// \brief размер одного блока дма в байтах
#define DMA_RAW_BLOCK_SIZE                  16
/// \brief размер дма буфера
#define DMA_SIZE_BYTES                      1048576

#pragma pack(push, 1)
/// \brief блок данных DMA
typedef struct {
    unsigned int word1;         ///< слово 1
    unsigned int word2;         ///< слово 2
    unsigned int word3;         ///< слово 3
    unsigned int word4;         ///< слово 4
} T_BLOCK_DMA;

#define T_BLOCK_DMA_TYPE_CH(word1)  ((word1 >> 31) & 0x1)
#define T_BLOCK_DMA_NUM_CH(word1)   ((word1 >> 24) & 0x7)

/// \brief контейнер DMA
typedef struct {
    unsigned int count;                         ///< кол-во блоков
    T_BLOCK_DMA blocks[DMA_COUNT_BLOCKS];       ///< блоки дма
} T_CONTAINER_DMA;
#pragma pack(pop)
```

Рисунок 4.2.12– Листинг функции

4.1.11 Записать в RAM передатчика – exta429_ writeRam

Описание функции приведено на рисунке 4.2.13.

```
/// \brief exta429_writeRam запись в RAM передатчика
/// \param fd дескриптор канала
/// \param container контейнер данных ram
/// \return результат операции
///
A429EXT_API unsigned int exta429_writeRam(A429ChannelFD cfd, RAM_BLOCK*
container);

#define RAM_BLOCK_TYPE_DATA             1
#define RAM_BLOCK_TYPE_DESCRIPTOR       2

/// \brief блок данных для записи в RAM передатчика
typedef struct {
    unsigned int type;      ///< тип инструкций (DATA - 1, DESCRIPTOR - 2)
    unsigned int shift;     ///< смещение от начала буфера в 32-х разрядных словах
    unsigned int length;    ///< размер поля dwords в 32-х разрядных словах
    unsigned int* dwords;   ///< данные (массив 32-х разрядных слов)
} RAM_BLOCK;
```

Рисунок 4.2.13– Листинг функции

4.1.12 Прочитать из RAM передатчика – exta429_ readRam

Описание функции приведено на рисунке 4.2.14.

```
/// \brief exta429_readRam чтение из RAM передатчика
/// \param fd дескриптор канала
/// \param container контейнер данных ram
/// \return результат операции
///
A429EXT_API unsigned int exta429_readRam(A429ChannelFD cfd, RAM_BLOCK* container);

#define RAM_BLOCK_TYPE_DATA             1
#define RAM_BLOCK_TYPE_DESCRIPTOR       2

/// \brief блок данных для записи в RAM передатчика
typedef struct {
    unsigned int type;      ///< тип инструкций (DATA - 1, DESCRIPTOR - 2)
    unsigned int shift;     ///< смещение от начала буфера в 32-х разрядных словах
    unsigned int length;    ///< размер поля dwords в 32-х разрядных словах
    unsigned int* dwords;   ///< данные (массив 32-х разрядных слов)
} RAM_BLOCK;
```

Рисунок 4.2.14– Листинг функции

4.1.13 Управление слежением за входными РК – exta429_ manageTrackingInterrupt

Описание функции приведено на рисунке 4.2.15.

```
/// \brief exta429_manageTrackingInterrupt управление слежением за входными РК
/// \param fd дескриптор канала
/// \param info инфо об управлении входными РК
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageTrackingInterrupt(A429ChannelFD cfd,
T_TRACKING_INT* info);


#define T_TRACKING_INT_MAN_OFF      0
#define T_TRACKING_INT_MAN_ON       ~T_TRACKING_INT_MAN_OFF

/// \brief управление слежением за РК
typedef struct {
    unsigned int manage;                        ///< вкл/выкл слежение
    unsigned int startTrackingFreeTimer;        ///< начальное значение freetimer
} T_TRACKING_INT;
```

Рисунок 4.2.15– Листинг функции

4.1.14 Проверить инфо о возникших входных РК – exta429_ checkTrackingInterrupt

Описание функции приведено на рисунке 4.2.16.

```
/// \brief exta429_checkTrackingInterrupt проверить информацию о входных РК
/// \param fd дескриптор канала
/// \param info инфо о входных РК
/// \return результат операции
///
A429EXT_API unsigned int exta429_checkTrackingInterrupt(A429ChannelFD cfd,
T_CHECK_TRACKING_INT* info);

/// \brief кол-во прерываний РК
#define T_CHECK_TRACKING_INT_CNT    8

/// \brief номер входной РК 1
#define T_CHECK_TRACKING_INT_RKIN_1 0
/// \brief номер входной РК 2
#define T_CHECK_TRACKING_INT_RKIN_2 1
/// \brief номер входной РК 3
#define T_CHECK_TRACKING_INT_RKIN_3 2
/// \brief номер входной РК 4
#define T_CHECK_TRACKING_INT_RKIN_4 3
/// \brief номер входной РК 5
#define T_CHECK_TRACKING_INT_RKIN_5 4
/// \brief номер входной РК 6
#define T_CHECK_TRACKING_INT_RKIN_6 5
/// \brief номер входной РК 7
#define T_CHECK_TRACKING_INT_RKIN_7 6
/// \brief номер входной РК 8
#define T_CHECK_TRACKING_INT_RKIN_8 7
/// \brief состояние рк вход
```

```
typedef struct {
    unsigned int startTrackingFreeTimer;        ///< начальное значение freetimer
    unsigned int finishTrackingFreeTimer;       ///< начальное значение freetimer
    unsigned int rkInAppearCount[T_CHECK_TRACKING_INT_CNT];            ///< кол-во
возникновений РК Вход
} T_CHECK_TRACKING_INT;
```

Рисунок 4.2.16– Листинг функции

4.1.15 Сброс ДМА – exta429_ clearDma

Описание функции приведено на рисунке 4.2.17.

```
/// \brief exta429_clearDma сброс дма
/// \param fd дескриптор канала
/// \return результат операции
///
A429EXT_API unsigned int exta429_clearDma(A429ChannelFD cfd);
```

Рисунок 4.2.17– Листинг функции

4.1.16 Управление ДМА – exta429_ manageDma

Описание функции приведено на рисунке 4.2.18.

```
/// \brief exta429_manageDma управление дма платы
/// \param fd дескриптор канала
/// \param value выкл - 0, вкл - ~0
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageDma(A429ChannelFD cfd, unsigned int value);
```

Рисунок 4.2.18– Листинг функции

4.1.17 Управление ДМА канала – exta429_ manageChannelDma

Описание функции приведено на рисунке 4.2.19.

```
/// \brief exta429_manageChannelsDma управление разрешением работы дма канала
/// \param fd дескриптор каналak
/// \param value выкл - 0, вкл - ~0
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageChannelDma(A429ChannelFD cfd, unsigned int
value);
```

Рисунок 4.2.19– Листинг функции

4.1.18 Управление маской прерываний – exta429_ manageInterruptMask

Описание функции приведено на рисунке 4.2.20.

```
/// \brief exta429_manageInterruptMask управление маской прерываний
/// \param fd дескриптор канала
/// \param mask маска
/// \param value значение
```

```
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageInterruptMask(A429ChannelFD cfd, unsigned
int mask, unsigned int value);
```

Рисунок 4.2.20– Листинг функции

4.1.19 Старт/стоп передатчика – exta429_ txStartStop

Описание функции приведено на рисунке 4.2.21.

```
/// \brief exta429_txStartStop работа передатчика (старт/стоп)
/// \param fd дескриптор канала
/// \param action старт/стоп (см. A429_START в a429_common.h )
/// \return результат операции
///
A429EXT_API unsigned int exta429_txStartStop(A429ChannelFD cfd, unsigned char
action);

#define A429_STOP           0               ///< стоп
#define A429_START          ~A429_STOP      ///< старт
```

Рисунок 4.2.21– Листинг функции

4.1.20 Однократный запуск передатчика – exta429_ txStartOnce

Описание функции приведено на рисунке 4.2.22.

```
/// \brief exta429_txStartOnce однократная работа передатчика
/// \param fd дескриптор канала
/// \return результат операции
///
A429EXT_API unsigned int exta429_txStartOnce(A429ChannelFD cfd);
```

Рисунок 4.2.22– Листинг функции

4.1.21 Выбор режима передатчика – exta429_ txSetMode

Описание функции приведено на рисунке 4.2.23.

```
/// \brief exta429_txSetMode управление режима передатчика
/// \param fd дескриптор канала
/// \param mode режим (см. A429_MODE_ в a429_common.h )
/// \return результат операции
///
A429EXT_API unsigned int exta429_txSetMode(A429ChannelFD cfd, unsigned char mode);

#define A429_MODE_0         0               ///< режим передатчика 0
#define A429_MODE_1         1               ///< режим передатчика 1
#define A429_MODE_2         2               ///< режим передатчика 2
#define A429_MODE_3         3               ///< режим передатчика 3
```

Рисунок 4.2.23– Листинг функции

4.1.22 Запись в FIFO передатчика – exta429_ writeTxFifo

Описание функции приведено на рисунке 4.2.24.

```
/// \brief exta429_writeTxFifo запись в tx fifo
/// \param fd дескриптор канала
/// \param data слово данных
/// \return результат операции
///
A429EXT_API unsigned int exta429_writeTxFifo(A429ChannelFD cfd, unsigned int
data);
```

Рисунок 4.2.24– Листинг функции

4.1.23 Управление метками фильтрации – exta429_ setLabelConfReg

Описание функции приведено на рисунке 4.2.25.

```
/// \brief exta429_setLabelConfReg управление регистрами LBL_CONF_REG
/// \param fd дескриптор канала
/// \param numReg номер регистра [0..7]
/// \param mask битовая маска изменения
/// \param value битовая маска значений
/// \return результат операции
///
A429EXT_API unsigned int exta429_setLabelConfReg(A429ChannelFD cfd, int numReg,
unsigned mask, unsigned int value);
```

Рисунок 4.2.25– Листинг функции

4.1.24 Управление скоростью канала – exta429_ setLabelConfReg

Описание функции приведено на рисунке 4.2.26.

```
/// \brief exta429_setSpeed установить скорость
/// \param fd дескриптор канала
/// \param speed скорость (см. A429_SPEED_ в a429_common.h )
/// \return результат операции
///
A429EXT_API unsigned int exta429_setSpeed(A429ChannelFD cfd, int speed);

#define A429_SPEED_100          0       ///< 100 кбит/с
#define A429_SPEED_12_14        1       ///< 12.5 кбит/с
#define A429_SPEED_50           2       ///< 50 кбит/с
#define A429_SPEED_12           3       ///< 12 кбит/с
#define A429_SPEED_14_5         4       ///< 14.5 кбит/с
```

Рисунок 4.2.26– Листинг функции

4.1.25 Разрешение работы канала – exta429_ manageEnBit

Описание функции приведено на рисунке 4.2.27.

```
/// \brief exta429_manageEnBit включение/выключение приёмника/передатчика
/// \param fd дескриптор канала
/// \param action старт/стоп (см. A429_START в a429_common.h )
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageEnBit(A429ChannelFD cfd, int action);
```

Рисунок 4.2.27– Листинг функции

4.1.26 Проверка разрешения работы канала – exta429_ isEnBit

Описание функции приведено на рисунке 4.2.28.

```
/// \brief exta429_isEnBit проверить состояние бита 31
/// \param fd дескриптор канала
/// \param state состояние бита (см. A429_ON в a429_common.h )
/// \return результат операции
///
A429EXT_API unsigned int exta429_isEnBit(A429ChannelFD cfd, int* state);
```

Рисунок 4.2.28– Листинг функции

4.1.27 Управление битом паритета – exta429_ parityManage

Описание функции приведено на рисунке 4.2.29.

```
/// \brief exta429_parityManage управление настройками бита паритета
/// \param fd дескриптор канала
/// \param parcheck вкл/выкл проверку паритета
/// \param parity состояние/режим бита паритета
/// \return результат операции
///
A429EXT_API unsigned int exta429_parityManage(A429ChannelFD cfd, int parcheck, int
parity);
```

Рисунок 4.2.29– Листинг функции

4.1.28 Управление работой фильтрации приёмника – exta429_ rxFiltration

Описание функции приведено на рисунке 4.2.30.

```
/// \brief exta429_rxFiltration включение/выключение фильтрации приёмника
/// \param fd дескриптор канала
/// \param action вкл/выкл
/// \return результат операции
///
A429EXT_API unsigned int exta429_rxFiltration(A429ChannelFD cfd, int action);
```

Рисунок 4.2.30– Листинг функции

4.1.29 Управление метками фильтрации приёмника – exta429_ rxManageFiltrationLabel

Описание функции приведено на рисунке 4.2.31.

```
/// \brief exta429_rxManageFiltrationLabel управление фильтрацией
LBL_CONF_REG_PCI_x
/// \param fd дескриптор канала
/// \param label метка (адрес)
/// \param action вкл/выкл
/// \return результат операции
A429EXT_API unsigned int exta429_rxManageFiltrationLabel(A429ChannelFD cfd, int
label, int action);
```

Рисунок 4.2.31– Листинг функции

4.1.30 Сброс меток фильтрации приёмника – exta429_ rxClearFiltrationLabel

Описание функции приведено на рисунке 4.2.32.

```
/// \brief exta429_rxClearFiltrationLabel сброс LBL_CONF_REG_PCI_x
/// \param fd дескриптор канала
/// \return результат операции
A429EXT_API unsigned int exta429_rxClearFiltrationLabel(A429ChannelFD cfd);
```

Рисунок 4.2.32– Листинг функции

4.1.31 Управление битом 9 и 10 приёмника – exta429_ rxManageRcvSdi

Описание функции приведено на рисунке 4.2.33.

```
/// \brief exta429_rxManageRcvSdi управление битом 9 и 10 по ГОСТ 18977-79
/// \param fd дескриптор канала
/// \param action вкл/выкл декодирования
/// \param bit9 состояние бита 0 - 1
/// \param bit10 состояние бита 0 - 1
/// \return результат операции
A429EXT_API unsigned int exta429_rxManageRcvSdi(A429ChannelFD cfd, int action, int
bit9, int bit10);
```

Рисунок 4.2.33– Листинг функции

4.1.32 Деинициализация канала – exta429_ deinit

Описание функции приведено на рисунке 4.2.34.

```
/// \brief exta429_deinit деинициализация
/// \param fd дескриптор канала
/// \return результат операции
///
A429EXT_API unsigned int exta429_deinit(A429ChannelFD cfd);
```

Рисунок 4.2.34– Листинг функции

4.1.33 Управление порядком хода бит метки – exta429_ manageReverce

Описание функции приведено на рисунке 4.2.35.

```
/// \brief exta429_manageReverce управление порядком бита метки
/// \param fd дескриптор канала
/// \param action 1/0
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageReverce(A429ChannelFD cfd, int action);
```

Рисунок 4.2.35– Листинг функции

4.1.34 Становить время паузы – exta429_ txGapBits

Описание функции приведено на рисунке 4.2.36.

```
/// \brief exta429_txGapBits установить время паузы
/// \param fd дескриптор канала
/// \param gapBit пауза
/// \return результат операции
///
A429EXT_API unsigned int exta429_txGapBits(A429ChannelFD cfd, unsigned int
gapBit);
```

Рисунок 4.2.36– Листинг функции

4.1.35 Управление дискретностью таймера RRT – exta429_ txRrD

Описание функции приведено на рисунке 4.2.37.

```
/// \brief exta429_txRrD установить делитель частоты таймера RRT
/// \param fd дескриптор канала
/// \param txrr делитель частоты таймера RRT (1 или 10 мс)  (см. A429_TXRR_ в
a429_common.h )
/// \return результат операции
///
A429EXT_API unsigned int exta429_txRrD(A429ChannelFD cfd, unsigned int txrr);

#define A429_TXRR_10MS      0       ///< txrr 10 ms
#define A429_TXRR_1MS       1       ///< txrr 1 ms
```

Рисунок 4.2.37– Листинг функции

4.1.36 Управление таймером RRT – exta429_ txSkipRrt

Описание функции приведено на рисунке 4.2.38.

```
/// \brief exta429_txSkipRrt управление битом skip rrt
/// \param fd дескриптор канала
/// \param action 1/0
/// \return результат операции
///
A429EXT_API unsigned int exta429_txSkipRrt(A429ChannelFD cfd, int action);
```

Рисунок 4.2.38– Листинг функции

4.1.37 Управление выходными РК – exta429_ manageScOut

Описание функции приведено на рисунке 4.2.39.

```
/// \brief exta429_manageScOut управление выходными РК
///
/// в качестве параметра action см. A429_SC_OUT_ в a429_common.h
///
/// \param fd дескриптор канала
/// \param action1 упр ВЫХ РК1
/// \param action2 упр ВЫХ РК2
/// \param action3 упр ВЫХ РК3
/// \param action4 упр ВЫХ РК4
/// \param action5 упр ВЫХ РК5
/// \param action6 упр ВЫХ РК6
/// \param action7 упр ВЫХ РК7
/// \param action8 упр ВЫХ РК8
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageScOut(A429ChannelFD cfd, int action1, int
action2, int action3, int action4, int action5, int action6, int action7, int
action8);

#define A429_SC_OUT_1           (1)         ///< ВЫХ РК в 0
#define A429_SC_OUT_0           (1<<1)      ///< ВЫХ РК в 1
#define A429_SC_OUT_NO_ACT      0           ///< ВЫХ РК не трогать
```

Рисунок 4.2.39– Листинг функции

4.1.38 Сброс разрешений входных РК – exta429_ clearScIntMask

Описание функции приведено на рисунке 4.2.40.

```
/// \brief exta429_clearScIntMask сброс разрешений
/// \param fd дескриптор канала
/// \return результат операции
///
A429EXT_API unsigned int exta429_clearScIntMask(A429ChannelFD cfd);
```

Рисунок 4.2.40– Листинг функции

4.1.39 Управление фронтом входных РК – exta429_ manageScIntMask

Описание функции приведено на рисунке 4.2.41.

```
/// \brief exta429_manageScIntMask управление фронтом входной РК
/// \param fd дескриптор канала
/// \param numRk номер ВХ РК (см. A429_IN_RK_ в a429_common.h)
/// \param stateRk фронт (см. A429_SC_RISE в a429_common.h)
/// \param actionRk вкл/выкл
/// \return результат операции
///
A429EXT_API unsigned int exta429_manageScIntMask(A429ChannelFD cfd, int numRk, int
stateRk, int actionRk);

#define A429_IN_RK_1    0       ///< вх рк 1
#define A429_IN_RK_2    1       ///< вх рк 2
#define A429_IN_RK_3    2       ///< вх рк 3
#define A429_IN_RK_4    3       ///< вх рк 4
#define A429_IN_RK_5    4       ///< вх рк 5
#define A429_IN_RK_6    5       ///< вх рк 6
#define A429_IN_RK_7    6       ///< вх рк 7
#define A429_IN_RK_8    7       ///< вх рк 8

#define A429_SC_RISE    0                   ///< передний фронт
#define A429_SC_FALL    ~A429_SC_RISE       ///< задний фронт
```

Рисунок 4.2.41– Листинг функции

4.1.40 Запуск передатчика с RRT – exta429_ txStartWithRrt

Описание функции приведено на рисунке 4.2.42.

```
/// \brief exta429_txStartWithRrt запуск с rrt в бесконечный цикл
/// \param fd дескриптор канала
/// \param rrtValue rrt значение
/// \return результат операции
///
A429EXT_API unsigned int exta429_txStartWithRrt(A429ChannelFD cfd, unsigned int
rrtValue);
```

Рисунок 4.2.42– Листинг функции

4.1.41 Управление передачей по запросу для передатчика – exta429_ txSetRequestTransferByRkIn

Описание функции приведено на рисунке 4.2.43.

```
/// \brief exta429_txSetRequestTransferByRkIn установить передачу по запросу для
передатчика по вх рк
/// \param fd дескриптор канала
/// \param rknum номер РК или отсутствие
/// \return результат операции
///
A429EXT_API unsigned int exta429_txSetRequestTransferByRkIn(A429ChannelFD cfd,
unsigned int rknum);
```

Рисунок 4.2.43– Листинг функции

4.1.42 Управление приёмом по готовности приёмника – exta429_ rxSetReceiveReadyByRkIn

Описание функции приведено на рисунке 4.2.44.

```
/// \brief exta429_rxSetReceiveReadyByRkIn установить готовность к приёму
приёмника по вх рк
/// \param fd дескриптор канала
/// \param rknum номер РК или отсутствие
/// \return результат операции
///
A429EXT_API unsigned int exta429_rxSetReceiveReadyByRkIn(A429ChannelFD cfd,
unsigned int rknum);
```

Рисунок 4.2.44– Листинг функции

4.1.43 Сброс RAM – exta429_ ramCls

Описание функции приведено на рисунке 4.2.45.

```
/// \brief exta429_ramCls RAM_CLS
/// \param fd дескриптор канала
/// \return результат операции
///
A429EXT_API unsigned int exta429_ramCls(A429ChannelFD cfd);
```

Рисунок 4.2.45– Листинг функции

5 СООБЩЕНИЯ

5.1 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429

Описание кодов ошибок приведено на рисунке 5.1.2.

```
/// \brief невалидное значение, свидетельствующее об ошибке
#define LIB_BAD_VALUE       0xFFFFFFFF
/// \brief значение отсутсвия ошибки
#define LIB_GOOD            0
```

Рисунок 5.1.2- Листинг

ПРИЛОЖЕНИЕ Б Пример программы.

Пример программы с использованием библиотеки взаимодействия.

```
#ifndef CTEST1_H
#define CTEST1_H

#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>
#include <unistd.h>
#include <strings.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>

#include "global.h"

///
/// \brief The TParamCTest1 struct параметры теста 1
///
struct TParamCTest1 {
    int mSpeed;             ///< скорость
    int mParityBit;         ///< бит чётности
    int mReverce;           ///< реверс
    int mMode;              ///< режим
    int mPrintTimeout;      ///< период выдачи информации в сек
    int mCountMessages;     ///< кол-во сообщений, отправленных за один такт
};

///< скорость
#define T_CTEST1_SPEED          ((char)'s')
///< чётность
#define T_CTEST1_PARITY         ((char)'p')
///< реверс
#define T_CTEST1_REVERCE        ((char)'r')
///< режим
#define T_CTEST1_MODE           ((char)'m')
///< таймаут
#define T_CTEST1_TIMEOUT        ((char)'t')
///< кол-во сообщений
#define T_CTEST1_CNT_MSG        ((char)'c')

extern char configFileName[];           ///< имя файла конфигурации, считанного из
аргументов
extern char fullConfigFileName[];       ///< полное имя файла конфигурации в папке
config
extern struct TParamCTest1 params;      ///< параметры теста 1
extern struct TChannel* channels;       ///< каналы
extern struct TDevice* devices;         ///< девайсы

///
```

```
/// \brief main точка входа в программу
/// \param argc кол-во аргументов
/// \param argv аргументы
/// \return код возврата
///
int main(int argc, char* argv[]);
///
/// \brief constructFullConfigFileName построить полное имя файла конфигурации
/// \param fullname полное имя
/// \param size размер
/// \param name имя
///
void constructFullConfigFileName(char* fullname, int size, char* name);
///
/// \brief readConfigFile чтение конфигурации из файла
/// \param fullname полное имя файла конфигурации
/// \param params параметры теста
/// \return код результата
///
int readConfigFile(char* fullname, struct TParamCTest1* params);
///
/// \brief procCTest1 процедура консольного теста 1
/// \param params параметры теста
///
void procCTest1(struct TParamCTest1 params);
///
/// \brief initLogCTest1 настройка логов теста 1
///
void initLogCTest1(void);
///
/// \brief initDefaultParamsCTest1 инициализация параметров ао-умолчанию теста 1
/// \param params параметры теста
///
void initDefaultParamsCTest1(struct TParamCTest1* params);

#endif // CTEST1_H
```

Листинг А.1 – ctest1.h

```
#include "ctest1.h"
#include <math.h>

# define timersub(a, b, result)                                                 \
  do {                                                                          \
    (result)->tv_sec = (a)->tv_sec - (b)->tv_sec;                               \
    (result)->tv_usec = (a)->tv_usec - (b)->tv_usec;                            \
    if ((result)->tv_usec < 0) {                                                \
      --(result)->tv_sec;                                                       \
      (result)->tv_usec += 1000000;                                             \
    }                                                                           \
  } while (0)

char configFileName[MAX_LEN_CONFIG_FILE];
char fullConfigFileName[MAX_LEN_FULL_CONFIG_FILE];
struct TParamCTest1 params;
struct TChannel* channels;
struct TDevice* devices;

int main(int argc, char* argv[])
{
    initLogCTest1();
    LOGGER_PRINT("%s %s\n","ctest1 - version", VERSION_APP);
    int ret = GOOD_CODE;
    if (!(argc == 2 || argc == 3)) { // проверка кол-ва аргументов
        usage:
#ifdef HAS_EXTA429_LIBRARY
        LOGGER_PRINT("%s\n","Usage: ctest1 <configfilename> <possible external
device IP address:port>\n" \
                            "Example: ctest1 ctest1.txt 192.168.12.34:19191\n");
#endif // HAS_EXTA429_LIBRARY

        LOGGER_PRINT("%s\n","Usage: ctest1 <configfilename>\n" \
                            "Example: ctest1 ctest1.txt\n");

        ret = BAD_CODE;
    } else { // кол-во аргументов - ок
        if (sscanf(argv[1], "%s", configFileName) != 1) { // чтение имени
конфигурационного файла
          LOGGER_PRINT("%s\n", "invalid configfilename\n");
          goto usage;
        } else { //имя конфигурационного файла - ок
          if(argc == 3)
          {
#ifdef HAS_EXTA429_LIBRARY
              if(!setNetworkDevice(argv[2])) // при наличии, парсим ip адрес
#endif
                  goto usage;
          }

          constructFullConfigFileName(fullConfigFileName,
MAX_LEN_FULL_CONFIG_FILE, configFileName);
          initDefaultParamsCTest1(&params);
          ret = readConfigFile(fullConfigFileName, &params); // чтение параметров
из конфигурационного файла
          if (ret == BAD_CODE) { //параметры из конфигурационного файла - error
              LOGGER_PRINT("Can not read test params from '%s' \n",
fullConfigFileName);
```

```
            } else { //параметры из конфигурационного файла - ок
                procCTest1(params);
            }
        }
    }
    LOGGER_FLUSH();
    LOGGER_CLOSE();

    close_all_abonents();
    return ret;
}

void constructFullConfigFileName(char* fullname, int size, char* name) {
    snprintf(fullname, size, "%s%s", STR_CONFIG, name);
}

int readConfigFile(char* fullname, struct TParamCTest1* params) {
    FILE* file = fopen(fullname, "r");
    char* str;
    if (file == NULL) {
        LOGGER_PRINT("Can not open config file '%s'\n", fullname);
        return BAD_CODE;
    }
    LOGGER_PRINT("parsing '%s':\n", fullname);
    while (!feof(file)) {
        readline(file, &str);
        LOGGER_PRINT("%s\n", str); // for debug
        char typeArgStr[1];
        int valArg = BAD_CODE;
        sscanf(str, "%s %d", typeArgStr, &valArg);
        if (valArg == BAD_CODE)
            continue;
        switch (typeArgStr[0]) {
            case T_CTEST1_SPEED:    params->mSpeed = valArg;        break;
            case T_CTEST1_PARITY:   params->mParityBit = valArg;    break;
            case T_CTEST1_REVERCE:  params->mReverce = valArg;      break;
            case T_CTEST1_MODE:     params->mMode = valArg;         break;
            case T_CTEST1_TIMEOUT:  params->mPrintTimeout = valArg; break;
            case T_CTEST1_CNT_MSG:  params->mCountMessages = valArg; break;
        }
    }
    fclose(file);
    LOGGER_PRINT("PARAMS: speed= %d, parity= %d, reverce= %d, mode= %d,
timeout= %d, count messages= %d\n\n", \
           params->mSpeed, params->mParityBit, params->mReverce, params->mMode,
params->mPrintTimeout, params->mCountMessages);
    return GOOD_CODE;
}

void procCTest1(struct TParamCTest1 params) {
    struct timeval tvStart, tvCur, tvDiff;
    int countDevices = 0;
    gettimeofday(&tvStart, NULL);
    getArrayDevices(&devices, &countDevices);
    LOGGER_PRINT("Has founded %d devices \n\n", countDevices);
    int countChannels = 0;
    getArrayChannels(&channels, &countChannels);
    LOGGER_PRINT("Has founded %d channels \n\n", countChannels);
    int counters[countChannels];    // счётчики данных блоков ДМА
    // инициализация девайсов
```

```
    for (int index = 0; index < countDevices; index++) {
        Abonent abonent = open_abonent_from_string(devices[index].mNameChannel);
        IS_EXT_WRAPPER(abonent, a429_clearDma);
        IS_EXT_WRAPPER(abonent, a429_manageDma, A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageInterruptMask,
                                1<<8 | 1<<9 | 1<<10 | 1<<11 | 1<<12 | 1<<13 |
1<<14 | 1<<15,
                                1<<8 | 1<<9 | 1<<10 | 1<<11 | 1<<12 | 1<<13 |
1<<14 | 1<<15);
        IS_EXT_WRAPPER(abonent, a429_clearScIntMask);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_1, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_2, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_3, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_4, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_5, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_6, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_7, A429_SC_RISE,
A429_ON);
        IS_EXT_WRAPPER(abonent, a429_manageScIntMask, A429_IN_RK_8, A429_SC_RISE,
A429_ON);
        T_TRACKING_INT trackInt;
        // включаем слежение за РК ВХ
        trackInt.manage = T_TRACKING_INT_MAN_ON;
        IS_EXT_WRAPPER(abonent, a429_manageTrackingInterrupt, &trackInt);
    }
    for (int index = 0; index < countChannels; index++) {
        Abonent abonent = open_abonent_from_string(channels[index].mNameChannel);
        IS_EXT_WRAPPER(abonent, a429_deinit);
    }
    // инициализация каналов
    for (int index = 0; index < countChannels; index++) {
        counters[index] = 0;
        Abonent abonent = open_abonent_from_string(channels[index].mNameChannel);
        IS_EXT_WRAPPER(abonent, a429_manageEnBit, A429_OFF);
        IS_EXT_WRAPPER(abonent, a429_manageChannelDma, A429_ON);
        IS_EXT_WRAPPER(abonent, a429_setSpeed, params.mSpeed);
        IS_EXT_WRAPPER(abonent, a429_parityManage, A429_ON, params.mParityBit);
        IS_EXT_WRAPPER(abonent, a429_manageReverce, params.mReverce);
        if (channels[index].mInfo.typeChannel == T_INFO_CHANNEL_TYPE_TRANSMITTER)
{
            IS_EXT_WRAPPER(abonent, a429_txSetMode, params.mMode);
            IS_EXT_WRAPPER(abonent, a429_txGapBits, 4);
        }

        if (channels[index].mInfo.typeChannel == T_INFO_CHANNEL_TYPE_TRANSMITTER)
{
            IS_EXT_WRAPPER(abonent, a429_manageEnBit, A429_ON);
            IS_EXT_WRAPPER(abonent, a429_txStartStop, A429_START);
        } else {
            IS_EXT_WRAPPER(abonent, a429_rxSetReceiveReadyByRkIn, RKIN_NUM_1);
        }
    }
    LOGGER_PRINT("%s\n", "test is processing...");
    // выполнение алгоритма тестирования
    for(;;) {
```

```
        for (int index = 0; index < countDevices; index++) {
            Abonent abonent =
open_abonent_from_string(devices[index].mNameChannel);
            IS_EXT_WRAPPER(abonent, a429_manageScOut, A429_SC_OUT_1,
A429_SC_OUT_1, A429_SC_OUT_1, A429_SC_OUT_1, A429_SC_OUT_1, A429_SC_OUT_1,
A429_SC_OUT_1, A429_SC_OUT_1);
        }
        for (int index = 0; index < countChannels; index++) {
            T_CONTAINER_DMA dma;
            dma.count = 0;
            // чтение данных
            Abonent abonent =
open_abonent_from_string(channels[index].mNameChannel);
            IS_EXT_WRAPPER(abonent, a429_readDma, &dma);

            counters[index] += dma.count;
            printErrA429Rcv(&channels[index], &dma);
        }
        gettimeofday(&tvCur, NULL);
        timersub(&tvCur, &tvStart, &tvDiff);    // вычисляем время прошедшее с
предыдущего вывода результатов
        if (tvDiff.tv_sec * 1000000 + tvDiff.tv_usec >= (params.mPrintTimeout
/*seconds*/ * 1000000)) { //проверка таймаута
            for (int index = 0; index < countDevices; index++) {
                Abonent abonent =
open_abonent_from_string(devices[index].mNameChannel);
                IS_EXT_WRAPPER(abonent, a429_manageScOut, A429_SC_OUT_0,
A429_SC_OUT_0, A429_SC_OUT_0, A429_SC_OUT_0, A429_SC_OUT_0, A429_SC_OUT_0,
A429_SC_OUT_0, A429_SC_OUT_0);
            }
            // выдача информации на экран и в лог и сброс накопленных счётчиков
принятых/переданных данных
            for (int index = 0; index < countChannels; index++) {
                LOGGER_PRINT("Channel '%s' = %d blocks \n",
channels[index].mNameChannel, counters[index]);
                counters[index] = 0;
            }
            for (int index = 0; index < countChannels; index++) {
                if (channels[index].mInfo.typeChannel ==
T_INFO_CHANNEL_TYPE_TRANSMITTER) {
                    Abonent abonent =
open_abonent_from_string(channels[index].mNameChannel);
                    // отправка данных
                    for (int i=0; i<params.mCountMessages; i++) {
                        IS_EXT_WRAPPER(abonent, a429_writeTxFifo, 0xAAAAAAAA);
                    }
                }
            }
            for (int index = 0; index < countDevices; index++) {
                Abonent abonent =
open_abonent_from_string(devices[index].mNameChannel);
                T_INFO_DEVICE info;
                IS_EXT_WRAPPER(abonent, a429_getDevInfoChannels, &info);
                int cntRk = max(2, min(info.countRecieverChannels,
info.countTransmitterChannels) / 2);

                T_CHECK_TRACKING_INT trackCheckInt;
                // проверяем результат ВХ РК
                IS_EXT_WRAPPER(abonent, a429_checkTrackingInterrupt,
&trackCheckInt);
```

```
                LOGGER_PRINT("Device '%s' devId= %d - checking",
devices[index].mNameChannel, devices[index].mDevId);

                for (int i=0; i < cntRk; i++) {
                    LOGGER_PRINT(" IN_RK %d= %d;", i+1,
trackCheckInt.rkInAppearCount[i]);
                }

                LOGGER_PRINT("%s", "\n");
            }
            for (int index = 0; index < countDevices; index++) {
                Abonent abonent =
open_abonent_from_string(devices[index].mNameChannel);
                T_TRACKING_INT trackInt;
                trackInt.manage = T_TRACKING_INT_MAN_OFF;
                IS_EXT_WRAPPER(abonent, a429_manageTrackingInterrupt, &trackInt);
                trackInt.manage = T_TRACKING_INT_MAN_ON;
                IS_EXT_WRAPPER(abonent, a429_manageTrackingInterrupt, &trackInt);
            }
            tvStart = tvCur;
            LOGGER_PRINT("-has passed %d seconds-\n", params.mPrintTimeout);
        }
        LOGGER_FLUSH();
    }
}

void initLogCTest1(void) {
    char filename[LOGGER_FILENAME_LENGTH];
    constructLogFilename(filename, LOGGER_FILENAME_LENGTH, "ctest1");
    LOGGER_OPEN(filename);
}

void initDefaultParamsCTest1(struct TParamCTest1* params) {
    params->mSpeed = 2;
    params->mParityBit = 1;
    params->mReverce = 0;
    params->mMode = 0;
    params->mPrintTimeout = 3;
    params->mCountMessages = 50;;
}
```

Листинг А.2 – ctest1.c

```
#ifndef GLOBAL_H
#define GLOBAL_H

#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>
#include <unistd.h>
#include <strings.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <dirent.h>

#ifdef HAS_A429_LIBRARY
#include "a429_library.h"
#endif

#ifdef HAS_EXTA429_LIBRARY
#include "a429ext_library.h"
#endif

#define VERSION_APP                     "15.02.2023"

/// \brief паттерн имени девайса
#define STR_CH0_PATTERN                 "-ch-0"
/// \brief паттерн имени канала
#define STR_DEV_PATTERN                 "a429dev"
/// \brief каталог с каналами
#define STR_DEV                         "/dev"
/// \brief каталог с конфигами тестов
#define STR_CONFIG                      "./config/"
/// \brief каталог с логами
#define STR_LOGS                        "./logs/"
/// \brief каталог с mem файлами
#define STR_MEM                         "./mem/"
/// \brief длина имени каталога config
#define LEN_STR_CONFIG                  9
/// \brief длина имени config файла
#define MAX_LEN_CONFIG_FILE             90
/// \brief длина полного имени config файла
#define MAX_LEN_FULL_CONFIG_FILE        MAX_LEN_CONFIG_FILE + LEN_STR_CONFIG
/// \brief длина имени канала
#define CHANNEL_NAME_LENGTH             30
/// \brief длина имени log файла
#define LOGGER_FILENAME_LENGTH          160
/// \brief длина имени mem файла
#define MEM_FILENAME_LENGTH             160
/// \brief код значения - хорошо
#define GOOD_CODE                       0
/// \brief код значения - плохо
#define BAD_CODE                        -1
```

```
///
/// \brief The TChannel struct информация о канале
///
struct TChannel {
    char mNameChannel[CHANNEL_NAME_LENGTH];             ///< полное имя канала
    T_INFO_CHANNEL mInfo;                               ///< инфо о канале
};

///
/// \brief The TDevice struct информация о девайсе
///
struct TDevice {
    char mNameChannel[CHANNEL_NAME_LENGTH];             ///< полное имя канала
    unsigned int mDevId;                                ///< device id
};

#ifdef HAS_EXTA429_LIBRARY

/// Задать сетевое устройство
bool setNetworkDevice(const char *device);

#endif // HAS_EXTA429_LIBRARY

///
/// \brief getCountDevices получить количество девайсов
/// \return кол-во девайсов
///
int getCountDevices(void);
///
/// \brief getCountChannels получить кол-во каналов ДПК
/// \return кол-во каналов
///
int getCountChannels(void);
///
/// \brief getArrayDevices получить массив описателей девайсов
/// \param devices массив описателей девайсов
/// \param length длина массива
///
void getArrayDevices(struct TDevice** devices, int* length);
///
/// \brief getArrayChannels получить массив описателей каналов ДПК
/// \param channels массив описателей каналов
/// \param length длина массива
///
void getArrayChannels(struct TChannel** channels, int* length);
///
/// \brief readline чтение строки
/// \param stream указатель текущего положения в файле
/// \param line указатель на строку
///
void readline(FILE* stream, char** line);
///
/// \brief constructLogFilename сгенерить имя лог файла
/// \param filename полное имя лог файла
/// \param length длина
/// \param testname имя теста
///
void constructLogFilename(char* filename, int length, char* testname);
///
/// \brief hasDmaBlockA429RcvErr проверка наличия ошибок в блоке дма примника a429
```

```
/// \param block блок дма
/// \return 1 - есть, 0 - не
///
int hasDmaBlockA429RcvErr(T_BLOCK_DMA* block);
///
/// \brief printErrA429Rcv печать ошибок из блоков дма приёмника a429
/// \param ch канал
/// \param dma контейнер дма
///
void printErrA429Rcv(struct TChannel* ch, T_CONTAINER_DMA* dma);

/// \remark LOGGER

///
/// \brief LOGGER_FILE дескриптор файла логов
///
extern FILE* LOGGER_FILE;
/// \brief открыть лог файл
#define LOGGER_OPEN(filename) \
    LOGGER_FILE = fopen(filename, "w");
/// \brief закрыть лог файл
#define LOGGER_CLOSE() \
    fclose(LOGGER_FILE)
/// \brief сбросить лог файл буфер на диск
#define LOGGER_FLUSH() \
    fflush(LOGGER_FILE)
/// \brief принт в лог и в терминал
#define LOGGER_PRINT(fmt, args...) \
    printf(fmt, args); \
    fprintf(LOGGER_FILE, fmt, args)

#ifndef max
#define max(x, y) (((x) > (y)) ? (x) : (y))
#endif

#ifndef min
#define min(x, y) (((x) < (y)) ? (x) : (y))
#endif

typedef struct
{
    char name[ CHANNEL_NAME_LENGTH ];
    bool isExt; // режим взаимодействия (true для внешнего устройства, с
библиотекой сокетов)

#ifdef HAS_A429_LIBRARY
    int fd; // дескриптор канала (при взаимодействии с драйвером)
#endif

#ifdef HAS_EXTA429_LIBRARY
    A429ChannelFD extFd; // дескриптор канала (при взаимодействии с сокетом)
#endif

} Abonent;

#if defined HAS_A429_LIBRARY && defined HAS_EXTA429_LIBRARY

/**
```

```
 \brief обёртка для точно такого же вызова, но с приставкой ext, для внешних
устройств
 \param abonent структура "абонент"
 \param FUNC название команды для выполнения, после которого следуют аргументы,
передаваемые команде
*/
#define IS_EXT_WRAPPER(abonent, FUNC, ...) \
    ((abonent.isExt) ? ext##FUNC( abonent.extFd, ##__VA_ARGS__ ) :
FUNC( abonent.fd, ##__VA_ARGS__ ))

#else

#ifdef HAS_A429_LIBRARY

#define IS_EXT_WRAPPER(abonent, FUNC, ...) \
    FUNC( abonent.fd, ##__VA_ARGS__ )

#else
#ifdef HAS_EXTA429_LIBRARY
#define IS_EXT_WRAPPER(abonent, FUNC, ...) \
    ext##FUNC( abonent.extFd, ##__VA_ARGS__ )

#endif
#endif
#endif

/// Элемент кэша абонентов
struct AbonentItem
{
    Abonent abonent; ///< абонент
    struct AbonentItem *next; ///< следующий элемент
};

/// Получить открытый дескриптор по имени устройства (результат кэшируется)
Abonent open_abonent_from_string(char *path);

/// Закрыть все дескрипторы из кэша
void close_all_abonents();

#endif // GLOBAL_H
```

Листинг А.3 – global.h

```
#include "global.h"
FILE* LOGGER_FILE;

#ifdef HAS_EXTA429_LIBRARY
const char *g_networkDevice = NULL;
char g_networkIP[30];
int g_networkPort = 0;

bool setNetworkDevice(const char *device)
{
    int ip[4];

    if(sscanf(device, "%d.%d.%d.%d:%d", &ip[0], &ip[1], &ip[2], &ip[3],
&g_networkPort) != 5)
        return false;

    for(int i = 0; i < 4; ++i)
        if(ip[i] < 0 || ip[i] > 255)
            return false;

    if(g_networkPort < 0 || g_networkPort > 0xFFFF)
        return false;

    sprintf(g_networkIP, "%d.%d.%d.%d", ip[0], ip[1], ip[2], ip[3]);
    g_networkDevice = device;
    return true;
}
#endif

int getCountDevices(void)
{
#ifdef HAS_EXTA429_LIBRARY
    if(g_networkDevice != NULL)
        return 1;
#endif

    int count = 0;
    DIR* dir = opendir(STR_DEV);
    if (dir) {
        struct dirent *direntries;
        while (((direntries = readdir(dir)) != NULL)) {
            char* isFindedChannel = strstr(direntries->d_name, STR_DEV_PATTERN);
            char* isFindedChannel0 = strstr(direntries->d_name, STR_CH0_PATTERN);
            if ((isFindedChannel != NULL) && (isFindedChannel0 != NULL)) {
//                printf("device: %s\n", direntries->d_name);
                count++;
            }
        }
        closedir(dir);
    }
    return count;
}

int getCountChannels(void)
{
#ifdef HAS_EXTA429_LIBRARY
    if(g_networkDevice != NULL)
    {
        char strFirstChannel[50];
```

```
        sprintf(strFirstChannel, "%s:%d:0", g_networkIP, g_networkPort);
        Abonent abonent = open_abonent_from_string(strFirstChannel);
        VERSION ver;
        T_INFO_DEVICE devInfo;
        IS_EXT_WRAPPER(abonent, a429_getDeviceInfo, &ver);
        if(GOOD == IS_EXT_WRAPPER(abonent, a429_getDevInfoChannels, &devInfo))
        {
            // debug
            LOGGER_PRINT("%s - devId %d - revision %d - приёмники %d (шт),
передатчики %d (шт)\n", strFirstChannel, ver.device_id, ver.revision,
devInfo.countRecieverChannels, devInfo.countTransmitterChannels);
            return devInfo.countRecieverChannels +
devInfo.countTransmitterChannels;
        }

        return 0;
    }
#endif

    int count = 0;
    DIR* dir = opendir(STR_DEV);
    if (dir) {
        struct dirent *direntries;
        while (((direntries = readdir(dir)) != NULL)) {
            char* isFinded = strstr(direntries->d_name, STR_DEV_PATTERN);
            if (isFinded != NULL) {
//                printf("%s\n", direntries->d_name);
                count++;
            }
        }
        closedir(dir);
    }
    return count;
}

void requestDeviceInfo(struct TDevice** devices, int index)
{
    Abonent abonent = open_abonent_from_string((*devices)[index].mNameChannel);
    VERSION ver;
    T_INFO_DEVICE devInfo;
    IS_EXT_WRAPPER(abonent, a429_getDeviceInfo, &ver);
    (*devices)[index].mDevId = ver.device_id;
    IS_EXT_WRAPPER(abonent, a429_getDevInfoChannels, &devInfo);
    // debug
    LOGGER_PRINT("%s - devId %d - приёмники %d (шт), передатчики %d (шт)\n",
(*devices)[index].mNameChannel, (*devices)[index].mDevId,
devInfo.countRecieverChannels, devInfo.countTransmitterChannels);
}

void getArrayDevices(struct TDevice** devices, int* length)
{
    *length = getCountDevices();
    *devices = malloc(sizeof(struct TDevice) * *length);
    int index = 0;

#ifdef HAS_EXTA429_LIBRARY
    if(g_networkDevice != NULL)
    {
        snprintf((*devices)[index].mNameChannel, CHANNEL_NAME_LENGTH, "%s:%d:0",
g_networkIP, g_networkPort);
```

```
        requestDeviceInfo(devices, index);
        return;
    }
#endif

    DIR* dir = opendir(STR_DEV);
    if (dir) {
        struct dirent *direntries;
        while (((direntries = readdir(dir)) != NULL)) {
            char* isFindedChannel = strstr(direntries->d_name, STR_DEV_PATTERN);
            char* isFindedChannel0 = strstr(direntries->d_name, STR_CH0_PATTERN);
            if ((isFindedChannel != NULL) && (isFindedChannel0 != NULL)) {
                snprintf((*devices)[index].mNameChannel, CHANNEL_NAME_LENGTH,
"%s/%s", STR_DEV, direntries->d_name);
                requestDeviceInfo(devices, index);
                index++;
            }
        }
        closedir(dir);
    }
}

void requestChannelInfo(struct TChannel **channels, int index)
{
    Abonent abonent = open_abonent_from_string((*channels)[index].mNameChannel);
    IS_EXT_WRAPPER(abonent, a429_getChannelInfo, &((*channels)[index].mInfo));
    // debug
    if ((*channels)[index].mInfo.typeChannel == T_INFO_CHANNEL_TYPE_RECIEVER) {
        LOGGER_PRINT("%s - num %d - T_INFO_CHANNEL_TYPE_RECIEVER\n",
(*channels)[index].mNameChannel, (*channels)[index].mInfo.numberChannel);
    } else {
        LOGGER_PRINT("%s - num %d - T_INFO_CHANNEL_TYPE_TRANSMITTER\n",
(*channels)[index].mNameChannel, (*channels)[index].mInfo.numberChannel);
    }
}

void getArrayChannels(struct TChannel **channels, int *length)
{
    *length = getCountChannels();
    *channels = malloc(sizeof(struct TChannel) * *length);
    int index = 0;

#ifdef HAS_EXTA429_LIBRARY
    if(g_networkDevice != NULL)
    {
        while(index < *length)
        {
            snprintf((*channels)[index].mNameChannel, CHANNEL_NAME_LENGTH,
"%s:%d:%d", g_networkIP, g_networkPort, index);
            requestChannelInfo(channels, index);
            index++;
        }
        return;
    }
#endif

    DIR* dir = opendir(STR_DEV);
    if (dir) {
        struct dirent *direntries;
        while (((direntries = readdir(dir)) != NULL)) {
```

```
            char* isFinded = strstr(direntries->d_name, STR_DEV_PATTERN);
            if (isFinded != NULL) {
                snprintf((*channels)[index].mNameChannel, CHANNEL_NAME_LENGTH,
"%s/%s", STR_DEV, direntries->d_name);
                requestChannelInfo(channels, index);
                index++;
            }
        }
        closedir(dir);
    }
}

void _readline(FILE* stream, char** line, int* count) {
    int ch = fgetc(stream);
    if ((ch == '\n') || (ch == EOF)) {
//        printf("end line - %d\n", *count);
        *line = malloc(*count + 1);
        (*line)[*count] = 0;
//        printf("down to up\n");
        return;
    } else {
        (*count)++;
//        printf("down - %d\n", *count);
        _readline(stream, line, count);
//        printf("up - %d\n", *count);
        (*line)[--(*count)] = ch;
    }
}

void readline(FILE* stream, char** line) {
    int count = 0;
    _readline(stream, line, &count);
}

void constructLogFilename(char* filename, int length, char* testname) {
    time_t rawtime;
    struct tm * timeinfo;
    time(&rawtime);
    timeinfo = localtime(&rawtime);
    snprintf(filename, length, "%s%s-%dy-%dm-%dd-%dh-%dm-%ds.txt", STR_LOGS,
testname, \
             timeinfo->tm_year + 1900, timeinfo->tm_mon + 1, timeinfo->tm_mday,
timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
}

int hasDmaBlockA429RcvErr(T_BLOCK_DMA* block) {
    if (block->word2 & (1<<21)) {
        return 1;
    }
    if (block->word2 & (1<<22)) {
        return 1;
    }
    if (block->word2 & (1<<23)) {
        return 1;
    }
    return 0;
}

void printErrA429Rcv(struct TChannel* ch, T_CONTAINER_DMA* dma) {
    if (ch->mInfo.typeChannel == T_INFO_CHANNEL_TYPE_RECIEVER) {
```

```
        for (int i=0; i < dma->count; i++) {
            if (hasDmaBlockA429RcvErr( &dma->blocks[i] )) {
                LOGGER_PRINT("Error in A429 %s dma word2= 0x%x:\n", ch-
>mNameChannel, dma->blocks[i].word2);
                if (dma->blocks[i].word2 & (1<<21)) {
                    LOGGER_PRINT("%s;\n", "Ошибка кодировки бита");
                }
                if (dma->blocks[i].word2 & (1<<22)) {
                    LOGGER_PRINT("%s;\n", "Ошибка паузы");
                }
                if (dma->blocks[i].word2 & (1<<23)) {
                    LOGGER_PRINT("%s;\n", "Ошибка чётности");
                }
            }
        }
    }
}

struct AbonentItem *g_abonentCache = NULL;

void close_all_abonents()
{
    struct AbonentItem *tmp;

    while(g_abonentCache != NULL)
    {
        if(g_abonentCache->abonent.isExt)
        {
#ifdef HAS_EXTA429_LIBRARY
            exta429_close(g_abonentCache->abonent.extFd);
#endif
        }
        else
        {
#ifdef HAS_A429_LIBRARY
            a429_close(g_abonentCache->abonent.fd);
#endif
        }

        tmp = g_abonentCache->next;
        free(g_abonentCache);
        g_abonentCache = tmp;
    }
}

Abonent open_abonent_from_string(char *path)
{
    int ipAddr[4], port, channel;
    Abonent result;

    for(struct AbonentItem *abonentCacheTmp = g_abonentCache; abonentCacheTmp !=
NULL; abonentCacheTmp = abonentCacheTmp->next)
        if(0 == strncmp(path, abonentCacheTmp->abonent.name, CHANNEL_NAME_LENGTH))
            return abonentCacheTmp->abonent;

    memset(&result, 0, sizeof(Abonent));
    result.isExt = false;

#ifdef HAS_A429_LIBRARY
```

```
    result.fd = -1;
#endif

#ifdef HAS_EXTA429_LIBRARY
    result.extFd.socketFD = -1;
#endif

    strncpy(result.name, path, CHANNEL_NAME_LENGTH);

    if(6 == sscanf(path, "%d.%d.%d.%d:%d:%d", &ipAddr[0], &ipAddr[1], &ipAddr[2],
&ipAddr[3], &port, &channel))
    {
#ifdef HAS_EXTA429_LIBRARY
        char str_ipAddr[100];
        sprintf(str_ipAddr, "%d.%d.%d.%d", ipAddr[0], ipAddr[1], ipAddr[2],
ipAddr[3]);

        result.isExt = true;
        result.extFd = exta429_open(str_ipAddr, port, channel);
#endif
    }
    else
    {
#ifdef HAS_A429_LIBRARY
        // локальное устройство
        result.fd = a429_open(path);
#endif
    }

    {
        struct AbonentItem *nextAbonentItem;

        if(g_abonentCache == NULL)
        {
            g_abonentCache = (struct AbonentItem *) malloc(sizeof(struct
AbonentItem));
            nextAbonentItem = g_abonentCache;
        }
        else
        {
            nextAbonentItem = g_abonentCache;

            while(nextAbonentItem->next != NULL)
                nextAbonentItem = nextAbonentItem->next;

            nextAbonentItem->next = (struct AbonentItem *) malloc(sizeof(struct
AbonentItem));
            nextAbonentItem = nextAbonentItem->next;
        }

        memset(nextAbonentItem, 0, sizeof(struct AbonentItem));
        nextAbonentItem->abonent = result;
    }

    return result;
}
```

Листинг А.4 – global.c

Лист регистрации

Номера листов					*Всего листов (страниц) в докум.*	*№ докумен -та*	*Входящий № сопроводи-тельного докум. и дата*	*Подп.*	*Да-та*
Изм.	*изменен-ных*	*заменен-ных*	*новых*	*аннули-рован-ных*					