УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2021 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «ДРАЙВЕР А429» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429»

Модулей
“PCIe–429UD88”
“mPCIe–429UD84”
“XMC–429UDxx”

(ОС LINUX)
(Astra Linux)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.15101-04 33 01-ЛУ

От	Инженер-программист
_____	_____ В.В. Колосов
«____»_____2021 г.	«____»_____2021 г.

2021

Из	*Под*	*Дат*

Литера

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

Утвержден

RU.МСКЮ.15101-04 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР А429» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429»

Модулей
“PCIe–429UD88”
“mPCIe–429UD84”
“XMC–429UDxx”

(ОС LINUX)
(Astra Linux)

Руководство программиста

RU.МСКЮ.15101-04 33 01

Листов - 52

2021

Из	*Под*	*Дат*

Литера

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

АННОТАЦИЯ

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

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ

ПО – программное обеспечение;

РК – разовая команда;

1 НАЗНАЧЕНИЕ ПРОГРАММЫ

1.1 ДРАЙВЕР А429

Программное обеспечение «ДРАЙВЕР А429» (далее – драйвер) обеспечивает возможность управления PCI-устройством.

Драйвер обеспечивает выполнение следующих основных задач:

– определение и инициализация устройства на шине PCI;

– инициализация символьных устройств каналов для обеспечения взаимодействия из юзерспейс пространства;

– реализация команд управления каналами.

1.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429

Программное обеспечение «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429» (далее – библиотека) обеспечивает вспомогательный сервисный функционал при взаимодействии с модулями:
«mPCIe–429UD84», «PCIe–429UD88» и «XMC–429UDxx”»(далее «xxx-429UDxx»).

Библиотека обеспечивает выполнение следующих основных задач:

– реализация сервисных функций поканально.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 ДРАЙВЕР A429

Драйвер является модулем ядра и предназначен для функционирования в ОС Linux (Astra Linux). Перед использованием драйвера необходимо его собрать и установить, инструкция по сборке и настройке приведена в Приложении А.

2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

Библиотека взаимодействия предназначена для функционирования в ОС Linux (Astra Linux) и встраивания в прикладное ПО для инициализации и управления модулями «xxx-429UDxx».

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 ДРАЙВЕР A429

Драйвер является модулем ядра ОС Linux (Astra Linux), разработан на языке С, после сборки представляет собой исполняемый объектный модуль с именем «a429_driver.ko».

Взаимодействие с каналами ARINC-429 осуществляется через символьные устройства поканально, которые расположение в каталоге «/dev».

Шаблон имени символьного устройства (канала): «a429dev-N-ch-M», где N-порядковый номер pci-платы, M — порядковый номер канала на pci-плате.

Нумерация плат (параметр N) с 0, нумерация каналов (параметр M) с 0.

Взаимодействие с каналами происходит посредством ioctl-команд.

3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

Библиотека взаимодействия разработана на языке С.

Для использования библиотеки в проекте необходимо в каталоге библиотеки выполнить команду «make», в результате которой появится файл a429lib.so и подключить к целевому проекту полученный файл и заголовочные фалы библиотеки.

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ДРАЙВЕР A429

Взаимодействие с каналами происходит посредством ioctl-команд, описание команд и типы данных представлены в файле «a429_cmd.h», реализация функций ioctl-команд представлена в файле «a429_ioctl.c».

4.1.1 Запись в регистр - IOCTL_WRITE_REG

Позволяет записать значение в заданный регистр управления.

Описание параметров команды приведено на рисунке 4.1.1.

```
/// \brief команда "запись в регистр"
/// позволяет записать значение в заданный регистр управления
#define IOCTL_WRITE_REG                                 _IOW(IOC_MAGIC, 0,
SADDR_DATA)

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

Рисунок 4.1.1 – Листинг команды

4.1.2 Запись битовой маски в регистр - IOCTL_WRITE_REG_BIT_MASK

Позволяет записать значение заданных бит в заданный регистр управления.

Описание параметров команды приведено на рисунке 4.1.2.

```
/// \brief команда "запись в регистр заданных бит"
/// позволяет изменить отдельные биты в заданном регистре управления
#define IOCTL_WRITE_REG_BIT_MASK                    _IOW(IOC_MAGIC, 1,
SADDR_DATA_BIT_MASK)

/// \brief запись в регистр заданных бит
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 4.1.2 – Листинг команды

4.1.3 Чтение из регистра - IOCTL_READ_REG

Позволяет прочитать значение из заданного регистра управления.

Описание параметров команды приведено на рисунке 4.1.3.

```
/// \brief команда "чтение из регистра"
/// позволяет прочитать значение из заданного регистра управления
#define IOCTL_READ_REG                                  _IOWR(IOC_MAGIC,2,
SADDR_DATA)

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

Рисунок 4.1.3 – Листинг команды

4.1.4 Чтение из ДМА - IOCTL_READ_DMA

Позволяет прочитать данные из ДМА канала.

Описание параметров команды приведено на рисунке 4.1.4.

```
/// \brief команда "чтение из дма"
/// позволяет прочитать данные из дма
#define IOCTL_READ_DMA                                  _IOWR(IOC_MAGIC,3,
T_CONTAINER_DMA)

/// \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 4.1.4 – Листинг команды

4.1.5 Сброс ДМА - IOCTL_CLEAR_REG

Позволяет обнулить DMA_INDEX и программные указатели чтения/записи.

Описание параметров команды приведено на рисунке 4.1.5.

```
/// \brief команда "сброс указателя dma канала"
/// позволяет обнулить DMA_INDEX и программные указатели чтения/записи
#define IOCTL_CLEAR_DMA                                 _IO(IOC_MAGIC,  13)
```

Рисунок 4.1.5 – Листинг команды

4.1.6 Получить информацию о плате - IOCTL_VERSION

Позволяет прочитать подробную информацию о pci-плате.

Описание параметров команды приведено на рисунке 4.1.6.

```
/// \brief команда "чтение подробной информации о плате и драйвере"
/// позволяет считать подробную информацию о pci-плате
#define IOCTL_VERSION                                   _IOR(IOC_MAGIC, 4, VERSION)

/// \brief информация о драйвере и устройстве
typedef struct {
    unsigned int device_id;     ///< идентификатор устройства
    unsigned int vendor_id;     ///< вендор устройства
    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 4.1.6 – Листинг команды

4.1.7 Получить версию и дату - IOCTL_VERSION_DRIVER

Позволяет прочитать версию и дату драйвера.

Описание параметров команды приведено на рисунке 4.1.7.

```
/// \brief команда "чтение версии драйвера"
/// позволяет считать версию и дату драйвера
#define IOCTL_VERSION_DRIVER                            _IOR(IOC_MAGIC, 5, unsigned
int)
```

Рисунок 4.1.7 – Листинг команды

4.1.8 Получить pci-локацию - IOCTL_PCI_LOCATION

Позволяет прочитать информацию о pci-локации платы.

Описание параметров команды приведено на рисунке 4.1.8.

```
/// \brief команда "чтение информации о pci-локации платы"
/// позволяет считать информацию pci
#define IOCTL_PCI_LOCATION                          _IOR(IOC_MAGIC, 6,
PCI_LOCATION)

#define SIZE_OF_PCI_LOC         13

/// \brief информация о локации на шине pci
typedef struct {
    ///
    /// \brief name инфо о локации
    /// например, 0000:00:02.0 -
    /// выделены домен (16 бит), шина (8 бит), устройство (5 бит) и функция (3
бита)
    ///
    char name[SIZE_OF_PCI_LOC];
} PCI_LOCATION;
```

Рисунок 4.1.8 – Листинг команды

4.1.9 Запись блока в RAM - IOCTL_WRITE_TO_RAM

Позволяет записать блок данных в RAM передатчика.

Описание параметров команды приведено на рисунке 4.1.9.

```
/// \brief команда "запись блока в RAM"
#define IOCTL_WRITE_TO_RAM                          _IOW(IOC_MAGIC, 7, RAM_BLOCK)

#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 4.1.9 – Листинг команды

4.1.10 Чтение блока из RAM - IOCTL_READ_FROM_RAM

Позволяет прочитать блок данных из RAM передатчика.

Описание параметров команды приведено на рисунке 4.1.10.

```
/// \brief команда "чтение блока из RAM"
#define IOCTL_READ_FROM_RAM                          _IOWR(IOC_MAGIC,8, RAM_BLOCK)

#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 4.1.10 – Листинг команды

4.1.11 Инфо о канале - IOCTL_INFO_CHANNEL

Позволяет прочитать информацию о канале.

Описание параметров команды приведено на рисунке 4.1.11.

```
/// \brief команда "информация о канале"
#define IOCTL_INFO_CHANNEL                           _IOWR(IOC_MAGIC, 9,
T_INFO_CHANNEL)

#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 4.1.11 – Листинг команды

4.1.12 Инфо о плате - IOCTL_INFO_DEVICE

Позволяет прочитать информацию о плате.

Описание параметров команды приведено на рисунке 4.1.12.

```
/// \brief команда "информация о плате"
#define IOCTL_INFO_DEVICE                              _IOWR(IOC_MAGIC, 10,
T_INFO_DEVICE)

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

Рисунок 4.1.12 – Листинг команды

4.1.13 Вкл/выкл слежения за прерываниями - IOCTL_TRACKING_INT

Позволяет включить – выключить функцию слежения за прерываниями.

Описание параметров команды приведено на рисунке 4.1.13.

```
/// \brief команда "слежение за прерываниями"
#define IOCTL_TRACKING_INT                             _IOWR(IOC_MAGIC, 11,
T_TRACKING_INT)

#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 4.1.13 – Листинг команды

4.1.14 Проверка возникновения прерываний - IOCTL_CHECK_TRACKING_INT

Позволяет проверить наличие возникновения прерываний с предыдущей проверки при включенной функции слежения за прерываниями.

Описание параметров команды приведено на рисунке 4.1.14.

```
/// \brief команда "проверка информации слежения за прерываниями"
#define IOCTL_CHECK_TRACKING_INT                    _IOWR(IOC_MAGIC, 12,
T_CHECK_TRACKING_INT)

/// \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 4.1.14 – Листинг команды

4.1.15 Режим «передача по запросу» - IOCTL_START_AS_RK_IN

Позволяет запустить канал (передатчик) в режиме «передача по запросу».

Описание параметров команды приведено на рисунке 4.1.15.

```
/// \brief команда "запуск канала в режиме передача по запросу"
#define IOCTL_START_AS_RK_IN                        _IOWR(IOC_MAGIC, 14, unsigned
int)

/// \brief входной РК - ничего не выбрано
#define RKIN_NONE      0
/// \brief номер входная РК 1
#define RKIN_NUM_1     1
/// \brief номер входная РК 2
#define RKIN_NUM_2     2
/// \brief номер входная РК 3
#define RKIN_NUM_3     3
/// \brief номер входная РК 4
#define RKIN_NUM_4     4
/// \brief номер входная РК 5
#define RKIN_NUM_5     5
/// \brief номер входная РК 6
#define RKIN_NUM_6     6
/// \brief номер входная РК 7
#define RKIN_NUM_7     7
/// \brief номер входная РК 8
#define RKIN_NUM_8     8
```

Рисунок 4.1.15 – Листинг команды

4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

Библиотека предназначена для функционирования в ОС Linux (Astra Linux) и встраивания в прикладное ПО. В файле «a429_library.h» представлены сервисные функции библиотеки взаимодействия.

4.2.1 Открыть канал – a429_open

Описание функции приведено на рисунке 4.2.1.

```
/// \brief a429_open открыть канал
/// \param _file имя файла символьного устройства канала
/// \return дескриптор открытого канала
#define a429_open(_file)
```

Рисунок 4.2.1– Листинг функции

4.2.2 Открыть канал с опциями – a429_open

Описание функции приведено на рисунке 4.2.2.

```
/// \brief a429_openfl открыть канал
/// \param _file имя файла символьного устройства канала
/// \param _oflags флаги-опции открытия
/// \return дескриптор открытого канала
#define a429_openfl(_file, _oflags)
```

Рисунок 4.2.2– Листинг функции

4.2.3 Закрыть канал – a429_close

Описание функции приведено на рисунке 4.2.3.

```
/// \brief a429_close закрыть канал
/// \param _fd дескриптор открытого канала
#define a429_close(_fd)
```

Рисунок 4.2.3– Листинг функции

4.2.4 Получить инфо о плате – a429_ getDeviceInfo

Описание функции приведено на рисунке 4.2.4.

```
///
/// \brief a429_getDeviceInfo получить информацию о канале и плате
/// \param fd дескриптор канала
/// \param info инфо о канале и плате
/// \return результат операции
///
unsigned int a429_getDeviceInfo(int fd, 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

4.2.5 Получить версию драйвера – a429_ getDriverVersion

Описание функции приведено на рисунке 4.2.5.

```
///
/// \brief a429_getDriverVersion получить версию драйвера
/// \param fd дескриптор канала
/// \param version версия драйвера
/// \return результат операции
///
unsigned int a429_getDriverVersion(int fd, unsigned int* version);
```

Рисунок 4.2.5– Листинг функции

4.2.6 Получить PCI-локацию платы – a429_ getDevicePciInfo

Описание функции приведено на рисунке 4.2.6.

```
///
/// \brief a429_getDevicePciInfo получить информацию о pci локации платы
/// \param fd дескриптор канала
/// \param pciInfo информация о pci локации платы
/// \return результат операции
///
unsigned int a429_getDevicePciInfo(int fd, PCI_LOCATION* pciInfo);

#define SIZE_OF_PCI_LOC         13

/// \brief информация о локации на шине pci
typedef struct {
    ///
    /// \brief name инфо о локации
    /// например, 0000:00:02.0 -
    /// выделены домен (16 бит), шина (8 бит), устройство (5 бит) и функция (3
бита)
    ///
    char name[SIZE_OF_PCI_LOC];
} PCI_LOCATION;
```

Рисунок 4.2.6– Листинг функции

4.2.7 Получить инфо о канале – a429_ getChannelInfo

Описание функции приведено на рисунке 4.2.7.

```
///
/// \brief a429_getChannelInfo получить информацию о канале
/// \param fd дескриптор канала
/// \param chInfo информация о канале
/// \return результат операции
///
unsigned int a429_getChannelInfo(int fd, 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

4.2.8 Получить инфо о каналах платы – a429_ getDevInfoChannels

Описание функции приведено на рисунке 4.2.8.

```
///
/// \brief a429_getDevInfoChannels получить информацию о каналах девайса
/// \param fd дескриптор канала
/// \param devInfoCh информация о каналах девайса
/// \return результат операции
///
unsigned int a429_getDevInfoChannels(int fd, 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

4.2.9 Записать регистр – a429_ writeReg

Описание функции приведено на рисунке 4.2.9.

```
///
/// \brief a429_writeReg записать значение регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int a429_writeReg(int fd, 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

4.2.10 Прочитать регистр – a429_ readReg

Описание функции приведено на рисунке 4.2.10.

```
///
/// \brief a429_readReg прочитать значение регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int a429_readReg(int fd, 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

4.2.11 Записать заданные биты регистра – a429_ modifyReg

Описание функции приведено на рисунке 4.2.11.

```
///
/// \brief a429_modifyReg изменить заданные биты регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int a429_modifyReg(int fd, SADDR_DATA_BIT_MASK* reg);

/// \brief запись в регистр заданных бит
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

4.2.12 Прочитать ДМА – a429_ readDma

Описание функции приведено на рисунке 4.2.12.

```
///
/// \brief a429_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
///
unsigned int a429_readDma(int fd, 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

4.2.13 Записать в RAM передатчика – a429_ writeRam

Описание функции приведено на рисунке 4.2.13.

```
///
/// \brief a429_writeRam запись в RAM передатчика
/// \param fd дескриптор канала
/// \param container контейнер данных ram
/// \return результат операции
///
unsigned int a429_writeRam(int fd, 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

4.2.14 Прочитать из RAM передатчика – a429_ readRam

Описание функции приведено на рисунке 4.2.14.

```
///
/// \brief a429_readRam чтение из RAM передатчика
/// \param fd дескриптор канала
/// \param container контейнер данных ram
/// \return результат операции
///
unsigned int a429_readRam(int fd, 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

4.2.15 Управление слежением за входными РК – a429_ manageTrackingInterrupt

Описание функции приведено на рисунке 4.2.15.

```
///
/// \brief a429_manageTrackingInterrupt управление слежением за входными РК
/// \param fd дескриптор канала
/// \param info инфо об управлении входными РК
/// \return результат операции
///
unsigned int a429_manageTrackingInterrupt(int fd, 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

4.2.16 Проверить инфо о возникших входных РК – a429_ checkTrackingInterrupt

Описание функции приведено на рисунке 4.2.16.

```
///
///
/// \brief a429_checkTrackingInterrupt проверить информацию о входных РК
/// \param fd дескриптор канала
/// \param info инфо о входных РК
/// \return результат операции
///
unsigned int a429_checkTrackingInterrupt(int fd, 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

4.2.17 Сброс ДМА – a429_ clearDma

Описание функции приведено на рисунке 4.2.17.

```
///
/// \brief a429_clearDma сброс дма
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a429_clearDma(int fd);
```

Рисунок 4.2.17– Листинг функции

4.2.18 Управление ДМА – a429_ manageDma

Описание функции приведено на рисунке 4.2.18.

```
///
/// \brief a429_manageDma управление дма платы
/// \param fd дескриптор канала
/// \param value выкл - 0, вкл - ~0
/// \return результат операции
///
unsigned int a429_manageDma(int fd, unsigned int value);
```

Рисунок 4.2.18– Листинг функции

4.2.19 Управление ДМА канала – a429_ manageChannelDma

Описание функции приведено на рисунке 4.2.19.

```
///
/// \brief a429_manageChannelsDma управление разрешением работы дма канала
/// \param fd дескриптор каналak
/// \param value выкл - 0, вкл - ~0
/// \return результат операции
///
unsigned int a429_manageChannelDma(int fd, unsigned int value);
```

Рисунок 4.2.19– Листинг функции

4.2.20 Управление маской прерываний – a429_ setMaskInterrupt

Описание функции приведено на рисунке 4.2.20.

```
///
/// \brief a429_setMaskInterrupt установка маски прерываний
/// \param fd дескриптор канала
/// \param mask маска изменения
/// \param value маска прерываний
/// \return результат операции
///
unsigned int a429_setMaskInterrupt(int fd, unsigned int mask, unsigned int value);
```

Рисунок 4.2.20– Листинг функции

4.2.21 Старт/стоп передатчика – a429_ txStartStop

Описание функции приведено на рисунке 4.2.21.

```
///
/// \brief a429_txStartStop работа передатчика (старт/стоп)
/// \param fd дескриптор канала
/// \param action старт/стоп
/// \return результат операции
///
unsigned int a429_txStartStop(int fd, unsigned char action);

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

Рисунок 4.2.21– Листинг функции

4.2.22 Однократный запуск передатчика – a429_ txStartOnce

Описание функции приведено на рисунке 4.2.22.

```
///
/// \brief a429_txStartOnce однократная работа передатчика
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a429_txStartOnce(int fd);
```

Рисунок 4.2.22– Листинг функции

4.2.23 Выбор режима передатчика – a429_ txSetMode

Описание функции приведено на рисунке 4.2.23.

```
///
/// \brief a429_txSetMode управление режима передатчика
/// \param fd дескриптор канала
/// \param mode режим
/// \return результат операции
///
unsigned int a429_txSetMode(int fd, 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

4.2.24 Запись в FIFO передатчика – a429_ writeTxFifo

Описание функции приведено на рисунке 4.2.24.

```
///
/// \brief a429_writeTxFifo запись в tx fifo
/// \param fd дескриптор канала
/// \param data слово данных
/// \return результат операции
///
unsigned int a429_writeTxFifo(int fd, unsigned int data);
```

Рисунок 4.2.24– Листинг функции

4.2.25 Управление метками фильтрации – a429_ setLabelConfReg

Описание функции приведено на рисунке 4.2.25.

```
///
/// \brief a429_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
///
unsigned int a429_setLabelConfReg(int fd, int numReg, unsigned mask, unsigned int
value);
```

Рисунок 4.2.25– Листинг функции

4.2.26 Управление скоростью канала – a429_ setLabelConfReg

Описание функции приведено на рисунке 4.2.26.

```
///
/// \brief a429_setSpeed установить скорость
/// \param fd дескриптор канала
/// \param speed скорость
/// \return результат операции
///
unsigned int a429_setSpeed(int fd, 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

4.2.27 Разрешение работы канала – a429_ manageEnBit

Описание функции приведено на рисунке 4.2.27.

```
///
/// \brief a429_manageEnBit включение/выключение приёмника/передатчика
/// \param fd дескриптор канала
/// \param action старт/стоп
/// \return результат операции
///
unsigned int a429_manageEnBit(int fd, int action);
```

Рисунок 4.2.27– Листинг функции

4.2.28 Проверка разрешения работы канала – a429_ isEnBit

Описание функции приведено на рисунке 4.2.28.

```
///
/// \brief a429_isEnBit проверить состояние бита 31
/// \param fd дескриптор канала
/// \param state состояние бита
/// \return результат операции
///
unsigned int a429_isEnBit(int fd, int* state);
```

Рисунок 4.2.28– Листинг функции

4.2.29 Управление битом паритета – a429_ parityManage

Описание функции приведено на рисунке 4.2.29.

```
///
/// \brief a429_parityManage управление настройками бита паритета
/// \param fd дескриптор канала
/// \param parcheck вкл/выкл проверку паритета
/// \param parity состояние/режим бита паритета
/// \return результат операции
///
unsigned int a429_parityManage(int fd, int parcheck, int parity);
```

Рисунок 4.2.29– Листинг функции

4.2.30 Управление работой фильтрации приёмника – a429_ rxFiltration

Описание функции приведено на рисунке 4.2.30.

```
///
/// \brief a429_rxFiltration включение/выключение фильтрации приёмника
/// \param fd дескриптор канала
/// \param action вкл/выкл
/// \return результат операции
///
unsigned int a429_rxFiltration(int fd, int action);
```

Рисунок 4.2.30– Листинг функции

4.2.31 Управление метками фильтрации приёмника – a429_ rxManageFiltrationLabel

Описание функции приведено на рисунке 4.2.31.

```
///
/// \brief a429_rxManageFiltrationLabel управление фильтрацией LBL_CONF_REG_PCI_x
/// \param fd дескриптор канала
/// \param label метка (адрес)
/// \param action вкл/выкл
/// \return результат операции
///
unsigned int a429_rxManageFiltrationLabel(int fd, int label, int action);
```

Рисунок 4.2.31– Листинг функции

4.2.32 Сброс меток фильтрации приёмника – a429_ rxClearFiltrationLabel

Описание функции приведено на рисунке 4.2.32.

```
///
/// \brief a429_rxClearFiltrationLabel сброс LBL_CONF_REG_PCI_x
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a429_rxClearFiltrationLabel(int fd);
```

Рисунок 4.2.32– Листинг функции

4.2.33 Управление битом 9 и 10 приёмника – a429_ rxManageRcvSdi

Описание функции приведено на рисунке 4.2.33.

```
///
/// \brief a429_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
///
unsigned int a429_rxManageRcvSdi(int fd, int action, int bit9, int bit10);
```

Рисунок 4.2.33– Листинг функции

4.2.34 Деинициализация канала – a429_ deinit

Описание функции приведено на рисунке 4.2.34.

```
///
/// \brief a429_deinit деинициализация
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a429_deinit(int fd);
```

Рисунок 4.2.34– Листинг функции

4.2.35 Управление порядком хода бит метки – a429_ manageReverce

Описание функции приведено на рисунке 4.2.35.

```
///
/// \brief a429_manageReverce управление порядком бита метки
/// \param fd дескриптор канала
/// \param action 1/0
/// \return результат операции
///
unsigned int a429_manageReverce(int fd, int action);
```

Рисунок 4.2.35– Листинг функции

4.2.36 Становить время паузы – a429_ txGapBits

Описание функции приведено на рисунке 4.2.36.

```
///
/// \brief a429_txGapBits установить время паузы
/// \param fd дескриптор канала
/// \param gapBit пауза
/// \return результат операции
///
unsigned int a429_txGapBits(int fd, unsigned int gapBit);
```

Рисунок 4.2.36– Листинг функции

4.2.37 Управление дискретностью таймера RRT – a429_ txRrD

Описание функции приведено на рисунке 4.2.37.

```
///
/// \brief a429_txRrD установить делитель частоты таймера RRT
/// \param fd дескриптор канала
/// \param txrr делитель частоты таймера RRT (1 или 10 мс)
/// \return результат операции
///
unsigned int a429_txRrD(int fd, 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

4.2.38 Управление таймером RRT – a429_ txSkipRrt

Описание функции приведено на рисунке 4.2.38.

```
///
/// \brief a429_txSkipRrt управление битом skip rrt
/// \param fd дескриптор канала
/// \param action 1/0
/// \return результат операции
///
unsigned int a429_txSkipRrt(int fd, int action);
```

Рисунок 4.2.38– Листинг функции

4.2.39 Управление выходными РК – a429_ manageScOut

Описание функции приведено на рисунке 4.2.39.

```
///
/// \brief a429_manageScOut управление выходными РК
/// \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
///
unsigned int  a429_manageScOut(int fd, 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

4.2.40 Сброс разрешений входных РК – a429_ clearScIntMask

Описание функции приведено на рисунке 4.2.40.

```
///
/// \brief a429_clearScIntMask сброс разрешений
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a429_clearScIntMask(int fd);
```

Рисунок 4.2.40– Листинг функции

4.2.41 Управление фронтом входных РК – a429_ manageScIntMask

Описание функции приведено на рисунке 4.2.41.

```
///
/// \brief a429_manageScIntMask управление фронтом входной РК
/// \param fd дескриптор канала
/// \param numRk номер ВХ РК
/// \param stateRk фронт
/// \param actionRk вкл/выкл
/// \return результат операции
///
unsigned int a429_manageScIntMask(int fd, 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

4.2.42 Запуск передатчика с RRT – a429_ txStartWithRrt

Описание функции приведено на рисунке 4.2.42.

```
///
/// \brief a429_txStartWithRrt запуск с rrt в бесконечный цикл
/// \param fd дескриптор канала
/// \param rrtValue rrt значение
/// \return результат операции
///
unsigned int a429_txStartWithRrt(int fd, unsigned int rrtValue);
```

Рисунок 4.2.42– Листинг функции

4.2.43 Однократный запуск передатчика с RRT – a429_ txStartWithRrtOnce

Описание функции приведено на рисунке 4.2.43.

```
///
/// \brief a429_txStartWithRrtOnce запуск с rrt разово
/// \param fd дескриптор канала
/// \param rrtValue rrt значение
/// \return результат операции
///
unsigned int a429_txStartWithRrtOnce(int fd, unsigned int rrtValue);
```

Рисунок 4.2.43– Листинг функции

4.2.44 Управление передачей по запросу для передатчика – a429_ txSetRequestTransferByRkIn

Описание функции приведено на рисунке 4.2.44.

```
///
/// \brief a429_txSetRequestTransferByRkIn установить передачу по запросу для
передатчика по вх рк
/// \param fd дескриптор канала
/// \param rknum номер РК или отсутствие
/// \return результат операции
///
unsigned int a429_txSetRequestTransferByRkIn(int fd, unsigned int rknum);
```

Рисунок 4.2.44– Листинг функции

4.2.45 Управление маской прерываний – a429_ manageInterruptMask

Описание функции приведено на рисунке 4.2.45.

```
///
/// \brief a429_manageInterruptMask управление маской прерываний
/// \param fd дескриптор канала
/// \param mask маска
/// \param value значение
/// \return результат операции
///
unsigned int a429_manageInterruptMask(int fd, unsigned int mask, unsigned int
value);
```

Рисунок 4.2.45– Листинг функции

4.2.46 Управление приёмом по готовности приёмника – a429_ rxSetReceiveReadyByRkIn

Описание функции приведено на рисунке 4.2.46.

```
///
/// \brief a429_rxSetReceiveReadyByRkIn установить готовность к приёму приёмника
по вх рк
/// \param fd дескриптор канала
/// \param rknum номер РК или отсутствие
/// \return результат операции
///
unsigned int a429_rxSetReceiveReadyByRkIn(int fd, unsigned int rknum);
```

Рисунок 4.2.46– Листинг функции

4.2.47 Сброс RAM – a429_ ramCls

Описание функции приведено на рисунке 4.2.47.

```
///
/// \brief a429_ramCls RAM_CLS
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a429_ramCls(int fd);
```

Рисунок 4.2.47– Листинг функции

5 СООБЩЕНИЯ

5.1 ДРАЙВЕР A429

В процессе работы драйвер сохраняет отладочную информацию, которую можно увидеть с помощью команды dmesg, набранную в терминале ОС Linux (Astra Linux).

5.1.1 Обработка ошибок IOCTL-команд

Описание кодов ошибок приведено на рисунке 5.1.1.

```
/// \brief нет ошибок
#define GOOD                            0
/// \brief ошибка
#define ERROR                           -1000
/// \brief ошибка адреса регистра
#define ERR_BAD_ADDRESS                 -1001
/// \brief ошибка копирования данных в юзерспейс
#define ERR_COPY_TO_USER                -1003
/// \brief ошибка аргумента - typr
#define ERR_BAD_ARG_TYPE                -1011
/// \brief ошибка выделения памяти
/// внутренняя ошибка драйвера
#define ERR_BAD_ALLOC_MEM               -1012
#define ERR_BAD_DMA_INDEX               -1013
#define ERR_BAD_DMA_NBLOCK              -1014
```

Рисунок 5.1.1– Листинг

5.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

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 А Сборка и установка драйвера

Проект драйвера можно собирать с помощью командной строки и утилиты make.

- с помощью командной строки и утилиты make:

открыть терминал в папке с проектом, написать в терминале "make"

для очистки проекта - "make clean"

для установки драйвера - "sudo make install"

для удаления драйвера - "sudo make uninstall"

для останова работающего драйвера - "sudo rmmod a429_driver"

для запуска установленного драйвера - "sudo insmod a429_driver.ko"

для проверки работает ли драйвер в данный момент - "sudo lsmod | grep a429_driver"

ТАКЖЕ перед началом работ следует установить:

"sudo apt-get install libelf-dev"

"sudo apt-get install linux-headers-generic".

ПРИЛОЖЕНИЕ Б Пример программы.

Пример программы с использованием библиотека 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

///
/// \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

```c
#include "ctest1.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();
    int ret = GOOD_CODE;
    if (argc != 2) { // проверка кол-ва аргументов
        usage:
        LOGGER_PRINT("%s\n","Usage: ctest1 <configfilename>");
        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
          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();
    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++) {
        devices[index].mFileDescryptor = a429_open(devices[index].mNameChannel);
        a429_clearDma(devices[index].mFileDescryptor);
        a429_manageDma(devices[index].mFileDescryptor, A429_ON);
        a429_manageInterruptMask(devices[index].mFileDescryptor, 1<<8 | 1<<9 |
1<<10 | 1<<11, 1<<8 | 1<<9 | 1<<10 | 1<<11);
        a429_manageScIntMask(devices[index].mFileDescryptor, A429_IN_RK_1,
A429_SC_FALL, A429_ON);
        a429_manageScIntMask(devices[index].mFileDescryptor, A429_IN_RK_2,
A429_SC_FALL, A429_ON);
        a429_manageScIntMask(devices[index].mFileDescryptor, A429_IN_RK_3,
A429_SC_FALL, A429_ON);
        a429_manageScIntMask(devices[index].mFileDescryptor, A429_IN_RK_4,
A429_SC_FALL, A429_ON);
        T_TRACKING_INT trackInt;
        // включаем слежение за РК ВХ
        trackInt.manage = T_TRACKING_INT_MAN_ON;
        a429_manageTrackingInterrupt(devices[index].mFileDescryptor, &trackInt);
        a429_close(devices[index].mFileDescryptor);
    }
    for (int index = 0; index < countChannels; index++) {
        channels[index].mFileDescryptor = a429_open(channels[index].mNameChannel);
```

```
        a429_deinit(channels[index].mFileDescryptor);
        a429_close(channels[index].mFileDescryptor);
    }
    // инициализация каналов
    for (int index = 0; index < countChannels; index++) {
        counters[index] = 0;
        channels[index].mFileDescryptor = a429_open(channels[index].mNameChannel);
        a429_manageEnBit(channels[index].mFileDescryptor, A429_OFF);
        a429_manageChannelDma(channels[index].mFileDescryptor, A429_ON);
        a429_setSpeed(channels[index].mFileDescryptor, params.mSpeed);
        a429_parityManage(channels[index].mFileDescryptor, A429_ON,
params.mParityBit);
        a429_manageReverce(channels[index].mFileDescryptor, params.mReverce);
        if (channels[index].mInfo.typeChannel == T_INFO_CHANNEL_TYPE_TRANSMITTER)
{
            a429_txSetMode(channels[index].mFileDescryptor, params.mMode);
        }
        a429_manageEnBit(channels[index].mFileDescryptor, A429_ON);
        if (channels[index].mInfo.typeChannel == T_INFO_CHANNEL_TYPE_TRANSMITTER)
{
            a429_txStartStop(channels[index].mFileDescryptor, A429_START);
        }
        a429_close(channels[index].mFileDescryptor);
    }
    LOGGER_PRINT("%s\n", "test is processing...");
    // выполнение алгоритма тестирования
    for(;;) {
        for (int index = 0; index < countDevices; index++) {
            devices[index].mFileDescryptor =
a429_open(devices[index].mNameChannel);
            a429_manageScOut(devices[index].mFileDescryptor, A429_SC_OUT_1,
A429_SC_OUT_1, A429_SC_OUT_1, A429_SC_OUT_1);
            a429_close(devices[index].mFileDescryptor);
        }
        for (int index = 0; index < countChannels; index++) {
            T_CONTAINER_DMA dma;
            dma.count = 0;
            // чтение данных
            channels[index].mFileDescryptor =
a429_open(channels[index].mNameChannel);
            a429_readDma(channels[index].mFileDescryptor, &dma);
            a429_close(channels[index].mFileDescryptor);
            counters[index] += dma.count;
        }
        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++) {
                devices[index].mFileDescryptor =
a429_open(devices[index].mNameChannel);
                a429_manageScOut(devices[index].mFileDescryptor, A429_SC_OUT_0,
A429_SC_OUT_0, A429_SC_OUT_0, A429_SC_OUT_0);
                a429_close(devices[index].mFileDescryptor);
            }
            // 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) {
                    channels[index].mFileDescryptor =
a429_open(channels[index].mNameChannel);
                    // отправка данных
                    for (int i=0; i<params.mCountMessages; i++) {
                        a429_writeTxFifo(channels[index].mFileDescryptor,
0xAAAAAAAA);
                    }
                    a429_close(channels[index].mFileDescryptor);
                }
            }
            for (int index = 0; index < countDevices; index++) {
                devices[index].mFileDescryptor =
a429_open(devices[index].mNameChannel);
                T_CHECK_TRACKING_INT trackCheckInt;
                // проверяем результат ВХ РК
                a429_checkTrackingInterrupt(devices[index].mFileDescryptor,
&trackCheckInt);
                LOGGER_PRINT("Device '%s' devId= %d - checking IN_RK 1= %d, IN_RK
2= %d, IN_RK 3= %d, IN_RK 4= %d\n", \
                             devices[index].mNameChannel, devices[index].mDevId, \

trackCheckInt.rkInAppearCount[T_CHECK_TRACKING_INT_RKIN_1],
trackCheckInt.rkInAppearCount[T_CHECK_TRACKING_INT_RKIN_2], \

trackCheckInt.rkInAppearCount[T_CHECK_TRACKING_INT_RKIN_3],
trackCheckInt.rkInAppearCount[T_CHECK_TRACKING_INT_RKIN_4]);
                a429_close(devices[index].mFileDescryptor);
            }
            for (int index = 0; index < countDevices; index++) {
                devices[index].mFileDescryptor =
a429_open(devices[index].mNameChannel);
                T_TRACKING_INT trackInt;
                trackInt.manage = T_TRACKING_INT_MAN_OFF;
                a429_manageTrackingInterrupt(devices[index].mFileDescryptor,
&trackInt);
                trackInt.manage = T_TRACKING_INT_MAN_ON;
                a429_manageTrackingInterrupt(devices[index].mFileDescryptor,
&trackInt);
                a429_close(devices[index].mFileDescryptor);
            }
            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 <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>
#include "../A429LinuxLibrary/a429_library.h"

/// \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
    int  mFileDescryptor;                               ///< дескриптор файла
    T_INFO_CHANNEL mInfo;                               ///< инфо о канале
};

///
/// \brief The TDevice struct информация о девайсе
///
```

```
struct TDevice {
    char mNameChannel[CHANNEL_NAME_LENGTH];             ///< полное имя канала
    int  mFileDescryptor;                               ///< дескриптор файла
    unsigned int mDevId;                                ///< device id
};

///
/// \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);

/// \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)
```

```
#endif // GLOBAL_H
```

Листинг А.3 – global.h

```
#include "global.h"

FILE* LOGGER_FILE;

int getCountDevices(void)
{
    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)
{
    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 getArrayDevices(struct TDevice** devices, int* length)
{
    *length = getCountDevices();
    *devices = malloc(sizeof(struct TDevice) * *length);
    int index = 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)) {
                snprintf((*devices)[index].mNameChannel, CHANNEL_NAME_LENGTH,
"%s/%s", STR_DEV, direntries->d_name);
                (*devices)[index].mFileDescryptor =
a429_open((*devices)[index].mNameChannel);
                VERSION ver;
                a429_getDeviceInfo((*devices)[index].mFileDescryptor, &ver);
                (*devices)[index].mDevId = ver.device_id;
                a429_close((*devices)[index].mFileDescryptor);
```

```
                // debug
                LOGGER_PRINT("%s - devId %d\n", (*devices)[index].mNameChannel,
(*devices)[index].mDevId);
                index++;
            }
        }
        closedir(dir);
    }
}

void getArrayChannels(struct TChannel **channels, int *length)
{
    *length = getCountChannels();
    *channels = malloc(sizeof(struct TChannel) * *length);
    int index = 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) {
                snprintf((*channels)[index].mNameChannel, CHANNEL_NAME_LENGTH,
"%s/%s", STR_DEV, direntries->d_name);
                (*channels)[index].mFileDescryptor =
a429_open((*channels)[index].mNameChannel);
                a429_getChannelInfo((*channels)[index].mFileDescryptor,
&((*channels)[index].mInfo));
                a429_close((*channels)[index].mFileDescryptor);
                // 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);
                }
                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);
                                        }
```

Листинг А.4 – global.c

Лист регистрации

Номера листов					*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводительного докум. и дата*	*Подп.*	*Дата*
Изм.	*измененных*	*замененных*	*новых*	*аннулированных*					