УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2021 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «ДРАЙВЕР А708» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А708»

Модулей
“PCIe–708UD2”
“mPCIe–708UD2”

(ОС LINUX)
(Astra Linux)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.24101-04 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	

RU.МСКЮ.15101-04 33 01

Утвержден

RU.МСКЮ.24101-04 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР А708» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А708»

Модулей
“PCIe–708UD2”
“mPCIe–708UD2”

(ОС LINUX)
(Astra Linux)

Руководство программиста

RU.МСКЮ.24101-04 33 01

Листов - 58

2021

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

Изм	Под	Дат

Литера

АННОТАЦИЯ

В книге описываются технологические принципы, использованные в программном обеспечении «ДРАЙВЕР А708» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А708». 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 A708

Программное обеспечение «ДРАЙВЕР A708» (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 A708

Программное обеспечение «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708» (далее – библиотека) обеспечивает вспомогательный сервисный функционал при взаимодействии с PCI-устройством xxx-708UD2.

Библиотека обеспечивает выполнение следующих основных задач:

– реализация сервисных функций поканально.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 ДРАЙВЕР A708

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

2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708

Библиотека взаимодействия предназначена для функционирования в ОС Astra Linux (Linux) и встраивания в прикладное ПО для инициализации и управления платами «PCIe-708UD2» и «mPCIe-708UD2».

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 ДРАЙВЕР A708

Драйвер является модулем ядра ОС Astra Linux (Linux), разработан на языке С, после сборки представляет собой исполняемый объектный модуль с именем «a708_driver.ko».

Взаимодействие с каналами ARINC708 и ARINC429 осуществляется через символьные устройства поканально, которые расположение в каталоге «/dev».

Шаблон имени символьного устройства (канала): «a708dev-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.

Каналы: ARINC429 приёмники – 0-3, ARINC429 передатчики – 4-5, ARINC708 приёмопередатчики – 6-7.

Взаимодействие с каналами происходит посредством ioctl-команд.

3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708

Библиотека взаимодействия разработана на языке С.

Для использования библиотеки в проекте необходимо в каталоге библиотеки выполнить команду «make», в результате которой появится файл a708lib.so и подключить к целевому проекту полученный файл и заголовочные фалы библиотеки.

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ДРАЙВЕР A708

Взаимодействие с каналами происходит посредством ioctl-команд, описание команд и типы данных представлены в файле «a708_cmd.h», реализация функций ioctl-команд представлена в файле «a708ud_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 записть 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 ARINC-429.

Описание параметров команды приведено на рисунке 4.1.4.

```
/// \brief команда "чтение из дма канала ARINC-429"
/// позволяет прочитать данные из дма канала ARINC-429
#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;

/// \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 Чтение из ДМА - IOCTL_A708_READ_DMA

Позволяет прочитать данные из ДМА канала ARINC-708.

Описание параметров команды приведено на рисунке 4.1.5.

```
/// \brief команда "чтение из дма A708 канала ARINC-708"
/// позволяет прочитать данные из дма A708 канала ARINC-708
#define IOCTL_A708_READ_DMA                          _IOWR(IOC_MAGIC, 15,
T_CONTAINER_DMA_A708)
/// \brief максимальное кол-во блоков дма в контейнере
#define DMA_COUNT_BLOCKS_A708                        64
/// \brief размер одного блока дма в байтах
#define DMA_RAW_BLOCK_SIZE_A708                      256
/// \brief кол_во слов в блоке A708
#define DMA_COUNT_WORDS_A708                         52

#pragma pack(push, 1)
/// \brief блок данных DMA
typedef struct {
    unsigned int words[DMA_COUNT_WORDS_A708];
} T_BLOCK_DMA_A708;

/// \brief контейнер DMA
typedef struct {
    unsigned int count;                                  ///< кол-во блоков
    T_BLOCK_DMA_A708 blocks[DMA_COUNT_BLOCKS_A708];      ///< блоки дма
} T_CONTAINER_DMA_A708;
#pragma pack(pop)
```

Рисунок 4.1.5 – Листинг команды

4.1.6 Сброс ДМА - IOCTL_CLEAR_DMA

Позволяет обнулить DMA_INDEX и программные указатели чтения/записи.

Описание параметров команды приведено на рисунке 4.1.6.

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

Рисунок 4.1.6 – Листинг команды

4.1.7 Получить информацию о плате - IOCTL_VERSION

Позволяет прочитать подробную информацию о pci-плате.

Описание параметров команды приведено на рисунке 4.1.7.

```
/// \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 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 4.1.7 – Листинг команды

4.1.8 Получить версию и дату - IOCTL_VERSION_DRIVER

Позволяет прочитать версию и дату драйвера.

Описание параметров команды приведено на рисунке 4.1.8.

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

Рисунок 4.1.8 – Листинг команды

4.1.9 Получить pci-локацию - IOCTL_PCI_LOCATION

Позволяет прочитать информацию о pci-локации платы.

Описание параметров команды приведено на рисунке 4.1.9.

```
/// \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 4.1.9 – Листинг команды

4.1.10 Запись блока в RAM - IOCTL_WRITE_TO_RAM

Позволяет записать блок данных в RAM передатчика ARINC-429.

Описание параметров команды приведено на рисунке 4.1.10.

```
/// \brief команда "запись блока в RAM передатчика ARINC-429"
#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 ARINC-429
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 Чтение блока из RAM - IOCTL_READ_FROM_RAM

Позволяет прочитать блок данных из RAM передатчика ARINC-429.

Описание параметров команды приведено на рисунке 4.1.11.

```
/// \brief команда "чтение блока из RAM передатчика ARINC-429"
#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 ARINC-429
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 4.1.11 – Листинг команды

4.1.12 Инфо о канале - IOCTL_INFO_CHANNEL

Позволяет прочитать информацию о канале.

Описание параметров команды приведено на рисунке 4.1.12.

```
/// \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 4.1.12 – Листинг команды

4.1.13 Инфо о плате - IOCTL_INFO_DEVICE

Позволяет прочитать информацию о плате.

Описание параметров команды приведено на рисунке 4.1.13.

```
/// \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 4.1.13 – Листинг команды

4.1.14 Вкл/выкл слежения за прерываниями - IOCTL_TRACKING_INT

Позволяет включить – выключить функцию слежения за прерываниями.

Описание параметров команды приведено на рисунке 4.1.14.

```
/// \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 4.1.14 – Листинг команды

4.1.15 Проверка возникновения прерываний РК - IOCTL_CHECK_TRACKING_INT

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

Описание параметров команды приведено на рисунке 4.1.15.

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

#define T_CHECK_TRACKING_INT_CNT    4

#define T_CHECK_TRACKING_INT_RKIN_1 0
#define T_CHECK_TRACKING_INT_RKIN_2 1
#define T_CHECK_TRACKING_INT_RKIN_3 2
#define T_CHECK_TRACKING_INT_RKIN_4 3

/// \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 4.1.15 – Листинг команды

4.1.16 Режим «передача по запросу» - IOCTL_START_AS_RK_IN

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

Описание параметров команды приведено на рисунке 4.1.16.

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

#define RKIN_NONE      0
#define RKIN_NUM_1     1
#define RKIN_NUM_2     2
#define RKIN_NUM_3     3
#define RKIN_NUM_4     4
```

Рисунок 4.1.16– Листинг команды

4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А708

Библиотека предназначена для функционирования в ОС Linux и встраивания в прикладное ПО. В файле «a708_library.h» представлены сервисные функции библиотеки взаимодействия.

ОБЩИЕ ФУНКЦИИ ДЛЯ РАБОТЫ С КАНАЛАМИ (ОБЩАЯ ФУНКЦИЯ)

4.2.1 Открыть канал – a708_open

Описание функции приведено на рисунке 4.2.1.

```
/// \brief a708_open открыть канал
/// ОБЩАЯ ФУНКЦИЯ
/// \param _file имя файла символьного устройства канала
/// \return дескриптор открытого канала
#define a708_open(_file)
```

Рисунок 4.2.1– Листинг функции

4.2.2 Открыть канал с опциями – a708_openfl

Описание функции приведено на рисунке 4.2.2.

```
/// \brief a708_openfl открыть канал
/// ОБЩАЯ ФУНКЦИЯ
/// \param _file имя файла символьного устройства канала
/// \param _oflags флаги-опции открытия
/// \return дескриптор открытого канала
#define a708_openfl(_file, _oflags)
```

Рисунок 4.2.2– Листинг функции

4.2.3 Закрыть канал – a708_close

Описание функции приведено на рисунке 4.2.3.

```
/// \brief a708_close закрыть канал
/// ОБЩАЯ ФУНКЦИЯ
/// \param _fd дескриптор открытого канала
#define a708_close(_fd)
```

Рисунок 4.2.3– Листинг функции

4.2.4 Получить инфо о плате – a708_getDeviceInfo

Описание функции приведено на рисунке 4.2.4.

```
///
/// \brief a708_getDeviceInfo получить информацию о канале и плате
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param info инфо о канале и плате
/// \return результат операции
///
unsigned int a708_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 – a708_getDriverVersion

Описание функции приведено на рисунке 4.2.5.

```
///
/// \brief a708_getDriverVersion получить версию драйвера
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param version версия драйвера
/// \return результат операции
///
unsigned int a708_getDriverVersion(int fd, unsigned int* version);
```

Рисунок 4.2.5– Листинг функции

4.2.6 Получить PCI-локацию платы – a708_getDevicePciInfo

Описание функции приведено на рисунке 4.2.6.

```
///
/// \brief a708_getDevicePciInfo получить информацию о pci локации платы
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param pciInfo информация о pci локации платы
/// \return результат операции
///
unsigned int a708_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 – a708_getChannelInfo

Описание функции приведено на рисунке 4.2.7.

```
///
/// \brief a708_getChannelInfo получить информацию о канале
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param chInfo информация о канале
/// \return результат операции
///
unsigned int a708_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 Проверка на тип канала A708 – A708_ASSERT_TYPE_A708

Описание функции приведено на рисунке 4.2.8.

```
/// \brief проверка на тип канала A708
/// ОБЩАЯ ФУНКЦИЯ
#define A708_ASSERT_TYPE_A708(fd) \
    if (a708_channelIsA708(fd) == (unsigned int)A708_FALSE) \
        return ERR_BAD_CHANNEL_TYPE
```

Рисунок 4.2.8– Листинг функции

4.2.9 Проверка на тип канала A429 – A708_ASSERT_NOT_TYPE_A708

Описание функции приведено на рисунке 4.2.9.

```
/// \brief проверка на тип канала A708
/// ОБЩАЯ ФУНКЦИЯ
#define A708_ASSERT_NOT_TYPE_A708(fd) \
    if (a708_channelIsA708(fd) == (unsigned int)A708_TRUE) \
        return ERR_BAD_CHANNEL_TYPE
```

Рисунок 4.2.9– Листинг функции

4.2.10 Проверка на тип канала A708 – a708_channelIsA708

Описание функции приведено на рисунке 4.2.10.

```
///
/// \brief a708_channelIsA708 проверка канала на тип A708
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return да/нет (A708_TRUE / A708_FALSE)
///
unsigned int a708_channelIsA708(int fd);
```

Рисунок 4.2.10– Листинг функции

4.2.11 Получить инфо о каналах платы – a708_getDevInfoChannels

Описание функции приведено на рисунке 4.2.11.

```
///
/// \brief a708_getDevInfoChannels получить информацию о каналах девайса
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param devInfoCh информация о каналах девайса
/// \return результат операции
///
unsigned int a708_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 4.2.11– Листинг функции

4.2.12 Записать регистр – a708_writeReg

Описание функции приведено на рисунке 4.2.12.

```
///
/// \brief a708_writeReg записать значение регистра
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int a708_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 4.2.12– Листинг функции

4.2.13 Прочитать регистр – a708_readReg

Описание функции приведено на рисунке 4.2.13.

```
///
/// \brief a708_readReg прочитать значение регистра
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int a708_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 4.2.13– Листинг функции

4.2.14 Записать заданные биты регистра – a708_modifyReg

Описание функции приведено на рисунке 4.2.14.

```
///
/// \brief a708_modifyReg изменить заданные биты регистра
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int a708_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 4.2.14– Листинг функции

4.2.15 Управление маской прерываний – a708_manageInterruptMask

Описание функции приведено на рисунке 4.2.15.

```
///
/// \brief a708_manageInterruptMask управление маской прерываний
/// ОБЩАЯ ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param mask маска
/// \param value значение
/// \return результат операции
///
unsigned int a708_manageInterruptMask(int fd, unsigned int mask, unsigned int
value);
```

Рисунок 4.2.15– Листинг функции

ФУНКЦИИ ДЛЯ РАБОТЫ С РАЗОВЫМИ КОМАНДАМИ (РК ФУНКЦИЯ)

4.2.16 Управление слежением за входными РК – a708_manageTrackingInterrupt

Описание функции приведено на рисунке 4.2.16.

```
///
/// \brief a708_manageTrackingInterrupt управление слежением за входными РК
/// РК ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param info инфо об управлении входными РК
/// \return результат операции
///
unsigned int a708_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 4.2.16– Листинг функции

4.2.17 Проверить инфо о возникших входных РК – a708_checkTrackingInterrupt

Описание функции приведено на рисунке 4.2.17.

```
///
///
/// \brief a708_checkTrackingInterrupt проверить информацию о входных РК
/// РК ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param info инфо о входных РК
/// \return результат операции
///
unsigned int a708_checkTrackingInterrupt(int fd, T_CHECK_TRACKING_INT* info);

#define T_CHECK_TRACKING_INT_CNT    4

#define T_CHECK_TRACKING_INT_RKIN_1 0
#define T_CHECK_TRACKING_INT_RKIN_2 1
#define T_CHECK_TRACKING_INT_RKIN_3 2
#define T_CHECK_TRACKING_INT_RKIN_4 3

/// \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 4.2.17– Листинг функции

4.2.18 Управление выходными РК – a708_manageScOut

Описание функции приведено на рисунке 4.2.18.

```
///
/// \brief a708_manageScOut управление выходными РК
/// РК ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param action1 упр ВЫХ РК1
/// \param action2 упр ВЫХ РК2
/// \param action3 упр ВЫХ РК3
/// \param action4 упр ВЫХ РК4
/// \return результат операции
///
unsigned int  a708_manageScOut(int fd, int action1, int action2, int action3, int
action4);

#define A708_SC_OUT_1           (1)         ///< ВЫХ РК в 0
#define A708_SC_OUT_0           (1<<1)      ///< ВЫХ РК в 1
#define A708_SC_OUT_NO_ACT      0           ///< ВЫХ РК не трогать
```

Рисунок 4.2.18– Листинг функции

4.2.19 Сброс разрешений входных РК – a708_clearScIntMask

Описание функции приведено на рисунке 4.2.19.

```
///
/// \brief a708_clearScIntMask сброс разрешений
/// РК ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_clearScIntMask(int fd);
```

Рисунок 4.2.19– Листинг функции

4.2.20 Управление фронтом входных РК – a708_manageScIntMask

Описание функции приведено на рисунке 4.2.20.

```
///
/// \brief a708_manageScIntMask управление фронтом входной РК
/// РК ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param numRk номер ВХ РК
/// \param stateRk фронт
/// \param actionRk вкл/выкл
/// \return результат операции
///
unsigned int a708_manageScIntMask(int fd, int numRk, int stateRk, int actionRk);

#define A708_IN_RK_1    0       ///< вх рк 1
#define A708_IN_RK_2    1       ///< вх рк 2
#define A708_IN_RK_3    2       ///< вх рк 3
#define A708_IN_RK_4    3       ///< вх рк 4

#define A708_SC_RISE    0                   ///< передний фронт
#define A708_SC_FALL    ~A708_SC_RISE       ///< задний фронт
```

Рисунок 4.2.20– Листинг функции

ФУНКЦИИ ДЛЯ РАБОТЫ С КАНАЛАМИ А429 (А429 ФУНКЦИЯ)

4.2.21 Прочитать ДМА канала ARINC-429 – a708_readDma

Описание функции приведено на рисунке 4.2.21.

```
///
/// \brief a708_readDma чтение доступных данных из дма ARINC-429
/// А429 ФУНКЦИЯ
/// (но не более 256 блоков за раз)
/// \param fd дескриптор канала
/// \param container контейнер данных дма
/// \return результат операции
///
unsigned int a708_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 4.2.21– Листинг функции

4.2.22 Записать в RAM передатчика канала ARINC-429 – a708_writeRam

Описание функции приведено на рисунке 4.2.22.

```
///
/// \brief a708_writeRam запись в RAM передатчика ARINC-429
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param container контейнер данных ram
/// \return результат операции
///
unsigned int a708_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 4.2.22– Листинг функции

4.2.23 Прочитать из RAM передатчика – a708_readRam

Описание функции приведено на рисунке 4.2.23.

```
///
/// \brief a708_readRam чтение из RAM передатчика
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param container контейнер данных ram
/// \return результат операции
///
unsigned int a708_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 4.2.23– Листинг функции

4.2.24 Сброс ДМА – a708_clearDma

Описание функции приведено на рисунке 4.2.24.

```
///
/// \brief a708_clearDma сброс дма
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_clearDma(int fd);
```

Рисунок 4.2.24– Листинг функции

4.2.25 Управление ДМА – a708_manageDma

Описание функции приведено на рисунке 4.2.25.

```
///
/// \brief a708_manageDma управление дма платы
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param value выкл - 0, вкл - ~0
/// \return результат операции
///
unsigned int a708_manageDma(int fd, unsigned int value);
```

Рисунок 4.2.25– Листинг функции

4.2.26 Управление ДМА канала – a708_manageChannelDma

Описание функции приведено на рисунке 4.2.26.

```
///
/// \brief a708_manageChannelsDma управление разрешением работы дма канала
/// A429 ФУНКЦИЯ
/// \param fd дескриптор каналak
/// \param value выкл - 0, вкл - ~0
/// \return результат операции
///
unsigned int a708_manageChannelDma(int fd, unsigned int value);
```

Рисунок 4.2.26– Листинг функции

4.2.27 Старт/стоп передатчика – a708_txStartStop

Описание функции приведено на рисунке 4.2.27.

```
///
/// \brief a708_txStartStop работа передатчика (старт/стоп)
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param action старт/стоп
/// \return результат операции
///
unsigned int a708_txStartStop(int fd, unsigned char action);

#define A708_STOP           0               ///< стоп
#define A708_START          ~A708_STOP      ///< старт
```

Рисунок 4.2.27– Листинг функции

4.2.28 Однократный запуск передатчика – a708_txStartOnce

Описание функции приведено на рисунке 4.2.28.

```
///
/// \brief a708_txStartOnce однократная работа передатчика
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_txStartOnce(int fd);
```

Рисунок 4.2.28– Листинг функции

4.2.29 Выбор режима передатчика – a708_txSetMode

Описание функции приведено на рисунке 4.2.29.

```
///
/// \brief a708_txSetMode управление режима передатчика
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param mode режим
/// \return результат операции
///
unsigned int a708_txSetMode(int fd, unsigned char mode);

#define A708_MODE_0         0               ///< режим передатчика 0
#define A708_MODE_1         1               ///< режим передатчика 1
#define A708_MODE_2         2               ///< режим передатчика 2
#define A708_MODE_3         3               ///< режим передатчика 3
```

Рисунок 4.2.29– Листинг функции

4.2.30 Запись в FIFO передатчика – a708_writeTxFifo

Описание функции приведено на рисунке 4.2.30.

```
///
/// \brief a708_writeTxFifo запись в tx fifo
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param data слово данных
/// \return результат операции
///
unsigned int a708_writeTxFifo(int fd, unsigned int data);
```

Рисунок 4.2.30– Листинг функции

4.2.31 Управление метками фильтрации – a708_setLabelConfReg

Описание функции приведено на рисунке 4.2.31.

```
///
/// \brief a708_setLabelConfReg управление регистрами LBL_CONF_REG
/// A429 ФУНКЦИЯ
/// \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 a708_setLabelConfReg(int fd, int numReg, unsigned mask, unsigned int
value);
```

Рисунок 4.2.31– Листинг функции

4.2.32 Управление скоростью канала – a708_setLabelConfReg

Описание функции приведено на рисунке 4.2.32.

```
///
/// \brief a708_setSpeed установить скорость
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param speed скорость
/// \return результат операции
///
unsigned int a708_setSpeed(int fd, int speed);

#define A708_SPEED_100          0       ///< 100 кбит/с
#define A708_SPEED_12_14        1       ///< 12-14,5 кбит/с
#define A708_SPEED_50           2       ///< 50 кбит/с
#define A708_SPEED_12           3       ///< 12 кбит/с
#define A708_SPEED_14_5         4       ///< 14.5 кбит/с
```

Рисунок 4.2.32– Листинг функции

4.2.33 Разрешение работы канала – a708_manageEnBit

Описание функции приведено на рисунке 4.2.33.

```
///
/// \brief a708_manageEnBit включение/выключение приёмника/передатчика
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param action старт/стоп
/// \return результат операции
///
unsigned int a708_manageEnBit(int fd, int action);
```

Рисунок 4.2.33– Листинг функции

4.2.34 Проверка разрешения работы канала – a708_isEnBit

Описание функции приведено на рисунке 4.2.34.

```
///
/// \brief a708_isEnBit проверить состояние бита 31
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param state состояние бита
/// \return результат операции
///
unsigned int a708_isEnBit(int fd, int* state);
```

Рисунок 4.2.34– Листинг функции

4.2.35 Управление битом паритета – a708_parityManage

Описание функции приведено на рисунке 4.2.35.

```
///
/// \brief a708_parityManage управление настройками бита паритета
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param parcheck вкл/выкл проверку паритета
/// \param parity состояние/режим бита паритета
/// \return результат операции
///
unsigned int a708_parityManage(int fd, int parcheck, int parity);
```

Рисунок 4.2.35– Листинг функции

4.2.36 Управление работой фильтрации приёмника – a708_rxFiltration

Описание функции приведено на рисунке 4.2.36.

```
///
/// \brief a708_rxFiltration включение/выключение фильтрации приёмника
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param action вкл/выкл
/// \return результат операции
///
unsigned int a708_rxFiltration(int fd, int action);
```

Рисунок 4.2.36– Листинг функции

4.2.37 Управление метками фильтрации приёмника – a708_rxManageFiltrationLabel

Описание функции приведено на рисунке 4.2.37.

```
///
/// \brief a708_rxManageFiltrationLabel управление фильтрацией LBL_CONF_REG_PCI_x
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param label метка (адрес)
/// \param action вкл/выкл
/// \return результат операции
///
unsigned int a708_rxManageFiltrationLabel(int fd, int label, int action);
```

Рисунок 4.2.37– Листинг функции

4.2.38 Сброс меток фильтрации приёмника – a708_rxClearFiltrationLabel

Описание функции приведено на рисунке 4.2.38.

```
///
/// \brief a708_rxClearFiltrationLabel сброс LBL_CONF_REG_PCI_x
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_rxClearFiltrationLabel(int fd);
```

Рисунок 4.2.38– Листинг функции

4.2.39 Управление битом 9 и 10 приёмника – a708_rxManageRcvSdi

Описание функции приведено на рисунке 4.2.39.

```
///
/// \brief a708_rxManageRcvSdi управление битом 9 и 10 по ГОСТ 18977-79
/// A429 ФУНКЦИЯ
/// \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 a708_rxManageRcvSdi(int fd, int action, int bit9, int bit10);
```

Рисунок 4.2.39– Листинг функции

4.2.40 Деинициализация канала – a708_deinit

Описание функции приведено на рисунке 4.2.40.

```
///
/// \brief a708_deinit деинициализация
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_deinit(int fd);
```

Рисунок 4.2.40– Листинг функции

4.2.41 Управление порядком хода бит метки – a708_manageReverce

Описание функции приведено на рисунке 4.2.41.

```
///
/// \brief a708_manageReverce управление порядком бита метки
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param action 1/0
/// \return результат операции
///
unsigned int a708_manageReverce(int fd, int action);
```

Рисунок 4.2.41– Листинг функции

4.2.42 Становить время паузы – a708_txGapBits

Описание функции приведено на рисунке 4.2.42.

```
///
/// \brief a708_txGapBits установить время паузы
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param gapBit пауза
/// \return результат операции
///
unsigned int a708_txGapBits(int fd, unsigned int gapBit);
```

Рисунок 4.2.42– Листинг функции

4.2.43 Управление дискретностью таймера RRT – a708_txRrD

Описание функции приведено на рисунке 4.2.43.

```
///
/// \brief a708_txRrD установить делитель частоты таймера RRT
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param txrr делитель частоты таймера RRT (1 или 10 мс)
/// \return результат операции
///
unsigned int a708_txRrD(int fd, unsigned int txrr);

#define A708_TXRR_10MS      0       ///< txrr 10 ms
#define A708_TXRR_1MS       1       ///< txrr 1 ms
```

Рисунок 4.2.43– Листинг функции

4.2.44 Управление таймером RRT – a708_txSkipRrt

Описание функции приведено на рисунке 4.2.44.

```
///
/// \brief a708_txSkipRrt управление битом skip rrt
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param action 1/0
/// \return результат операции
///
unsigned int a708_txSkipRrt(int fd, int action);
```

Рисунок 4.2.44– Листинг функции

4.2.45 Запуск передатчика с RRT – a708_txStartWithRrt

Описание функции приведено на рисунке 4.2.45.

```
///
/// \brief a708_txStartWithRrt запуск с rrt в бесконечный цикл
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param rrtValue rrt значение
/// \return результат операции
///
unsigned int a708_txStartWithRrt(int fd, unsigned int rrtValue);
```

Рисунок 4.2.45– Листинг функции

4.2.46 Управление передачей по запросу для передатчика – a708_txSetRequestTransferByRkIn

Описание функции приведено на рисунке 4.2.46.

```
///
/// \brief a708_txSetRequestTransferByRkIn установить передачу по запросу для
передатчика по вх рк
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param rknum номер РК или отсутствие
/// \return результат операции
///
unsigned int a708_txSetRequestTransferByRkIn(int fd, unsigned int rknum);
```

Рисунок 4.2.46– Листинг функции

4.2.47 Управление приёмом по готовности приёмника – a708_rxSetReceiveReadyByRkIn

Описание функции приведено на рисунке 4.2.47.

```
///
/// \brief a708_rxSetReceiveReadyByRkIn установить готовность к приёму приёмника
по вх рк
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param rknum номер РК или отсутствие
/// \return результат операции
///
unsigned int a708_rxSetReceiveReadyByRkIn(int fd, unsigned int rknum);
```

Рисунок 4.2.47– Листинг функции

4.2.48 Сброс RAM – a708_ramCls

Описание функции приведено на рисунке 4.2.48.

```
///
/// \brief a708_ramCls RAM_CLS
/// A429 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_ramCls(int fd);
```

Рисунок 4.2.48– Листинг функции

ФУНКЦИИ ДЛЯ РАБОТЫ С ПРИЁМО-ПЕРЕДАТЧИКАМИ A708 (A708 ФУНКЦИЯ)

4.2.49 Управление приёмником A708 – a708_rx708_manage

Описание функции приведено на рисунке 4.2.49.

```
///
/// \brief a708_rx708_manage включение-выключение приёмника a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param value вкл - 1, выкл - 0 (A708_ON, A708_OFF)
/// \return результат операции
///
unsigned int a708_rx708_manage(int fd, unsigned int value);
```

Рисунок 4.2.49– Листинг функции

4.2.50 Управление опцией SHORT_DMA приёмника A708 – a708_rx708_shortDma

Описание функции приведено на рисунке 4.2.50.

```
///
/// \brief a708_rx708_shortDma управление short dma приёмника a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param value вкл - 1, выкл - 0 (A708_ON, A708_OFF)
/// \return результат операции
///
unsigned int a708_rx708_shortDma(int fd, unsigned int value);
```

Рисунок 4.2.50– Листинг функции

4.2.51 Управление режимом передатчика A708 – a708_tx708_setMode

Описание функции приведено на рисунке 4.2.51.

```
/// \brief режим передатчика a708 - выключен
#define A708_TX708_MODE_OFF             0
/// \brief режим передатчика a708 - однократная отправка
#define A708_TX708_MODE_SINGLE_SHOT     1
/// \brief режим передатчика a708 - автоматическая отправка
#define A708_TX708_MODE_AUTO            3

///
/// \brief a708_tx708_setMode управление режимом передатчика a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param value режим
/// \return результат операции
///
unsigned int a708_tx708_setMode(int fd, unsigned int value);
```

Рисунок 4.2.51– Листинг функции

4.2.52 Запуск передатчика A708 – a708_tx708_start

Описание функции приведено на рисунке 4.2.52.

```
///
/// \brief a708_tx708_start старт передатчика a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param value режим
/// \return результат операции
///
unsigned int a708_tx708_start(int fd, unsigned int value);
```

Рисунок 4.2.52– Листинг функции

4.2.53 Останов передатчика A708 – a708_tx708_stop

Описание функции приведено на рисунке 4.2.53.

```
///
/// \brief a708_tx708_stop останов передатчика a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int a708_tx708_stop(int fd);
```

Рисунок 4.2.53– Листинг функции

4.2.54 Интервальный таймер A708 – a708_tx708_writeTimer

Описание функции приведено на рисунке 4.2.54.

```
///
/// \brief a708_tx708_writeTimer запись интервального таймера передатчика a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param value значение таймера
/// \return результат операции
///
unsigned int a708_tx708_writeTimer(int fd, unsigned int value);
```

Рисунок 4.2.54– Листинг функции

4.2.55 Запись в буфер на передачу A708 – a708_tx708_writeDataBlock

Описание функции приведено на рисунке 4.2.55.

```
/// \brief блок данных a708
typedef struct {
    unsigned int words[A708_DATA_BLOCK_SIZE_708];       ///< слова данных
    int length;                                         ///< кол-во слов данных
} T_A708_DATA_BLOCK;

///
/// \brief a708_tx708_writeDataBlock запись блока данных в передатчик a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param numBlock номер блока 1 - 8
/// \param dataBlock блок данных a708
/// \return результат операции
///
unsigned int a708_tx708_writeDataBlock(int fd, int numBlock, T_A708_DATA_BLOCK*
dataBlock);
```

Рисунок 4.2.55– Листинг функции

4.2.56 Чтение из буфера на передачу A708 – a708_tx708_readDataBlock

Описание функции приведено на рисунке 4.2.56.

```
/// \brief размер блока данных a708 - 50 слов
#define A708_DATA_BLOCK_SIZE_708    50

/// \brief блок данных a708
typedef struct {
    unsigned int words[A708_DATA_BLOCK_SIZE_708];       ///< слова данных
    int length;                                         ///< кол-во слов данных
} T_A708_DATA_BLOCK;

///
/// \brief a708_tx708_readDataBlock чтение блока данных из передатчика a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param numBlock номер блока 1 - 8
/// \param dataBlock блок данных a708
/// \return результат операции
///
unsigned int a708_tx708_readDataBlock(int fd, int numBlock, T_A708_DATA_BLOCK*
dataBlock);
```

Рисунок 4.2.56– Листинг функции

4.2.57 Проверка отправки из буфера A708 – a708_tx708_hasSendedDataBlock

Описание функции приведено на рисунке 4.2.57.

```
///
/// \brief a708_tx708_hasSendedDataBlock проверка передан ли блок данных из
передатчика a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param numBlock номер блока 1 - 8
/// \param hasSended признак передан ли блок данных
/// \return результат операции
///
unsigned int a708_tx708_hasSendedDataBlock(int fd, int numBlock, unsigned int *
hasSended);
```

Рисунок 4.2.57– Листинг функции

4.2.58 Чтение из ДМА приёмника A708 – a708_readDmaA708

Описание функции приведено на рисунке 4.2.58.

```
///
/// \brief a708_readDmaA708 чтение дма a708
/// A708 ФУНКЦИЯ
/// \param fd дескриптор канала
/// \param container контейнер
/// \return результат операции
///
unsigned int a708_readDmaA708(int fd, T_CONTAINER_DMA_A708* container);
```

Рисунок 4.2.58– Листинг функции

5 СООБЩЕНИЯ

5.1 ДРАЙВЕР A708

В процессе работы драйвер сохраняет отладочную информацию, которую можно увидеть с помощью команды dmesg, набранную в терминале ОС Astra Linux (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
/// \brief ошибка - невалидный дма индекс
#define ERR_BAD_DMA_INDEX               -1013
/// \brief ошибка - невалидный номер дма блока
#define ERR_BAD_DMA_NBLOCK              -1014
/// \brief ошибка - неверный тип канала
#define ERR_BAD_CHANNEL_TYPE            -1015
```

Рисунок 5.1.1– Листинг

5.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708

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 (ОБЯЗАТЕЛЬНОЕ)

Инструкция по сборке и установке 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 a708_driver"

для запуска установленного драйвера - "sudo insmod a708_driver.ko"

для проверки работает ли драйвер в данный момент - "sudo lsmod | grep a708_driver"

ТАКЖЕ перед началом работ следует установить:

"sudo apt-get install libelf-dev"

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

ПРИЛОЖЕНИЕ Б (ИНФОРМАЦИОННОЕ)

Пример программы с использованием библиотека взаимодействия

```
#ifndef CTEST3_H
#define CTEST3_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 TParamCTest3 struct параметры теста 3
///
struct TParamCTest3 {
    int mCh1Mode;           ///< режим канала 1
    int mCh2Mode;           ///< режим канала 2
    int mTimeoutSend;       ///< таймаут отправки
};

///< режим канала 1
#define T_CTEST3_CH1_MODE       "ch1m"
///< режим канала 2
#define T_CTEST3_CH2_MODE       "ch2m"
///< таймаут отправки
#define T_CTEST3_TIMEOUT_T      "tmt"

extern char configFileName[];           ///< имя файла конфигурации, считанного из
аргументов
extern char fullConfigFileName[];       ///< полное имя файла конфигурации в папке
config
extern struct TParamCTest3 params;      ///< параметры теста 3
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 TParamCTest3* params);
///
/// \brief procCTest3 процедура консольного теста 3
/// \param params параметры теста
///
void procCTest3(struct TParamCTest3 params, int countDevices);
///
/// \brief initLogCTest3 настройка логов теста 1
///
void initLogCTest3(void);
///
/// \brief initDefaultParamsCTest3 инициализация параметров ао-умолчанию теста 1
/// \param params параметры теста
///
void initDefaultParamsCTest3(struct TParamCTest3* params);

#endif // CTEST3_H
```

Листинг А.1 – ctest3.h

```c
#include "ctest3.h"

char configFileName[MAX_LEN_CONFIG_FILE];
char fullConfigFileName[MAX_LEN_FULL_CONFIG_FILE];
struct TParamCTest3 params;
struct TChannel* channels;
struct TDevice* devices;
int countChannels = 0;

int main(int argc, char* argv[])
{
    int countDevices = 0;
    initLogCTest3();
    LOGGER_PRINT("%s %s\n","ctest3 - version", VERSION_APP);
    int ret = GOOD_CODE;
    if (argc != 2) { // проверка кол-ва аргументов
        usage:
        LOGGER_PRINT("%s\n","Usage: ctest3 <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);
          initDefaultParamsCTest3(&params);
          ret = readConfigFile(fullConfigFileName, &params); // чтение параметров
из конфигурационного файла
          getArrayDevices(&devices, &countDevices);
          if (countDevices == 0) {
              LOGGER_PRINT("%s\n", "Has not founded any devices for testing");
              goto END;
          }
          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
              procCTest3(params, countDevices);
          }
        }
    }
    END:
    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 TParamCTest3* 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[10];
        int valArg = BAD_CODE;
        sscanf(str, "%s %d", typeArgStr, &valArg);
        if (valArg == BAD_CODE)
            continue;
        if (0 == strncmp (typeArgStr, T_CTEST3_CH1_MODE,
sizeof(T_CTEST3_CH1_MODE))) {
            params->mCh1Mode = valArg;
        } else if (0 == strncmp (typeArgStr, T_CTEST3_CH2_MODE,
sizeof(T_CTEST3_CH2_MODE))) {
            params->mCh2Mode = valArg;
        } else if (0 == strncmp (typeArgStr, T_CTEST3_TIMEOUT_T,
sizeof(T_CTEST3_TIMEOUT_T))) {
            params->mTimeoutSend = valArg;
        }
    }
    fclose(file);
    LOGGER_PRINT("PARAMS: ch1 mode= %d, ch2 mode= %d, timeout send= %d\n\n", \
           params->mCh1Mode, params->mCh2Mode, params->mTimeoutSend);
    return GOOD_CODE;
}

#define PRINT_TIMEOUT_MS    3

///
/// \brief The TStatisticsCTest3 struct текущая статистика теста 3
///
struct TStatisticsCTest3 {
    int mCh1DmaBlocks;
    int mCh1Errors;
};

void clearStat(struct TStatisticsCTest3 * stat) {
    int i =0;
    for (i=0; i<countChannels; i++) {
    stat[i].mCh1DmaBlocks = 0;
    stat[i].mCh1Errors = 0;
    }
}

void printStat(const struct TStatisticsCTest3 * stat) {
    int i =0;
    for (i=0; i<countChannels; i++) {
        channels[i].mFileDescryptor = a708_open(channels[i].mNameChannel);
        a708_getChannelInfo(channels[i].mFileDescryptor, &(channels[i].mInfo));
        if (channels[i].mInfo.typeChannel != T_INFO_CHANNEL_TYPE_A708) {
            a708_close(channels[i].mFileDescryptor);
            continue;
        }
        a708_close(channels[i].mFileDescryptor);
        LOGGER_PRINT("ch-%s: ch_blk= %d, ch_err= %d \n", channels[i].mNameChannel,
stat[i].mCh1DmaBlocks, stat[i].mCh1Errors);
    }


}
```

```
void send(int indexCh, T_A708_DATA_BLOCK* blockA708, T_A708_DATA_BLOCK*
blockA708R, struct TStatisticsCTest3 * stat) {
    T_CONTAINER_DMA_A708 dmaA708;
    int isEqual = 1;
    channels[indexCh].mFileDescryptor = a708_open(channels[indexCh].mNameChannel);
    dmaA708.count = 0;
    a708_readDmaA708(channels[indexCh].mFileDescryptor, &dmaA708);

    stat[indexCh].mCh1DmaBlocks+= dmaA708.count;
    for (int i=0; i<dmaA708.count; i++){
        if (isErrInDmaBlock(dmaA708.blocks[i].words[0]))
            stat[indexCh].mCh1Errors+= 1;
    }

    a708_tx708_writeDataBlock(channels[indexCh].mFileDescryptor, 1, blockA708);
    a708_tx708_readDataBlock(channels[indexCh].mFileDescryptor, 1, blockA708R);
    for (int i=0; i<A708_DATA_BLOCK_SIZE_708; i++){
        if (blockA708->words[i] != blockA708R->words[i]) {
            isEqual = 0;
            break;
        }
    }
    if (isEqual == 0)
        LOGGER_PRINT("A708-%d writed blocks are not equal\n", \
                     indexCh);
    a708_tx708_start(channels[indexCh].mFileDescryptor,
A708_TX708_MODE_SINGLE_SHOT);


    LOGGER_PRINT("ch %d send\n", indexCh);
    a708_close(channels[indexCh].mFileDescryptor);
}

int isErrInDmaBlock(unsigned int word) {
    int isErr = 0;
    if ((word & 1<<16) > 0) isErr = 1;
    if ((word & 1<<17) > 0) isErr = 1;
    if ((word & 1<<18) > 0) isErr = 1;
    if ((word & 1<<19) > 0) isErr = 1;
    return isErr;
}


void recv(int indexCh, struct TStatisticsCTest3* stat) {
    int numBlock;
    T_CONTAINER_DMA_A708 dmaA708;
    channels[indexCh].mFileDescryptor = a708_open(channels[indexCh].mNameChannel);
    a708_readDmaA708(channels[indexCh].mFileDescryptor, &dmaA708);
    LOGGER_PRINT("ch %d recv\n", indexCh);
    if (dmaA708.count == 0)
        return;

        stat[indexCh].mCh1DmaBlocks += dmaA708.count;

    for (numBlock = 0; numBlock < dmaA708.count; numBlock++) {
        if (isErrInDmaBlock(dmaA708.blocks[numBlock].words[ 0 ])) {

                stat[indexCh].mCh1Errors += 1;

        }
    }
```

```
    a708_close(channels[indexCh].mFileDescryptor);
}

void procCTest3(struct TParamCTest3 params, int countDevices) {
    T_A708_DATA_BLOCK blockA708, blockA708R;
    int i = 0;
    struct timeval tvStart, tvCur, tvDiff, tvStartPrint, tvDiffPrint;
    struct TStatisticsCTest3* stat;      // текущая статистика теста
    gettimeofday(&tvStart, NULL);
    LOGGER_PRINT("Has founded %d devices \n\n", countDevices);


    getArrayChannels(&channels, &countChannels);
    stat = (struct TStatisticsCTest3*)malloc(sizeof(struct TStatisticsCTest3) *
countChannels);
    LOGGER_PRINT("Has founded %d channels \n\n", countChannels);
    // инициализация девайсов
    for (int index = 0; index < countDevices; index++) {
        devices[index].mFileDescryptor = a708_open(devices[index].mNameChannel);

        a708_clearDma(devices[index].mFileDescryptor);
        a708_manageDma(devices[index].mFileDescryptor, A708_ON);
        a708_close(devices[index].mFileDescryptor);
    }
    for (int index = 0; index < countChannels; index++) {
        channels[index].mFileDescryptor = a708_open(channels[index].mNameChannel);
        a708_getChannelInfo(channels[index].mFileDescryptor,
&(channels[index].mInfo));
        if (channels[index].mInfo.typeChannel != T_INFO_CHANNEL_TYPE_A708) {
            a708_close(channels[index].mFileDescryptor);
            continue;
        }
        a708_deinit(channels[index].mFileDescryptor);
        a708_close(channels[index].mFileDescryptor);
    }
    // инициализация каналов
    for (int index = 0; index < countChannels; index++) {
        channels[index].mFileDescryptor = a708_open(channels[index].mNameChannel);
        a708_getChannelInfo(channels[index].mFileDescryptor,
&(channels[index].mInfo));
        if (channels[index].mInfo.typeChannel != T_INFO_CHANNEL_TYPE_A708) {
            a708_close(channels[index].mFileDescryptor);
            continue;
        }
        a708_rx708_manage(channels[index].mFileDescryptor, A708_OFF);
        a708_tx708_setMode(channels[index].mFileDescryptor, A708_TX708_MODE_OFF);
        if ((channels[index].mInfo.numberChannel == 1-1) && (params.mCh1Mode ==
1)) {
            a708_tx708_setMode(channels[index].mFileDescryptor,
A708_TX708_MODE_SINGLE_SHOT);
            a708_rx708_manage(channels[index].mFileDescryptor, A708_ON);
        } else if ((channels[index].mInfo.numberChannel == 1-1) &&
(params.mCh1Mode == 2)) {
            a708_tx708_setMode(channels[index].mFileDescryptor,
A708_TX708_MODE_OFF);
            a708_rx708_manage(channels[index].mFileDescryptor, A708_ON);
        }
        if ((channels[index].mInfo.numberChannel == 2-1) && (params.mCh2Mode ==
1)) {
            a708_rx708_manage(channels[index].mFileDescryptor, A708_ON);
```

```
            a708_tx708_setMode(channels[index].mFileDescryptor,
A708_TX708_MODE_SINGLE_SHOT);
        } else if ((channels[index].mInfo.numberChannel == 2-1) &&
(params.mCh2Mode == 2)) {
            a708_tx708_setMode(channels[index].mFileDescryptor,
A708_TX708_MODE_OFF);
            a708_rx708_manage(channels[index].mFileDescryptor, A708_ON);
        }
        a708_rx708_shortDma(channels[index].mFileDescryptor, A708_OFF);
        a708_close(channels[index].mFileDescryptor);
    }
    clearStat(stat);
    for (int i=0; i<A708_DATA_BLOCK_SIZE_708; i++){
        blockA708.words[i] = 0xAAAA;
        blockA708R.words[i] = 0;
    }
    blockA708.length = A708_DATA_BLOCK_SIZE_708;
    blockA708R.length = A708_DATA_BLOCK_SIZE_708;
    gettimeofday(&tvStart, NULL);
    gettimeofday(&tvStartPrint, NULL);

    LOGGER_PRINT("%s\n", "test is processing...");
    // выполнение алгоритма тестирования
    for(;;) {



        gettimeofday(&tvCur, NULL);
        timersub(&tvCur, &tvStart, &tvDiff);    // вычисляем время прошедшее с
предыдущего вывода результатов
        if (tvDiff.tv_sec * 1000000 + tvDiff.tv_usec >= (params.mTimeoutSend
/*mseconds*/  * 1000000)) { //проверка таймаута
            for (int index = 0; index < countChannels; index++) {
                channels[index].mFileDescryptor =
a708_open(channels[index].mNameChannel);
                a708_getChannelInfo(channels[index].mFileDescryptor,
&(channels[index].mInfo));
                if (channels[index].mInfo.typeChannel != T_INFO_CHANNEL_TYPE_A708)
{
                    a708_close(channels[index].mFileDescryptor);
                    continue;
                }
                a708_close(channels[index].mFileDescryptor);
                if (channels[index].mInfo.numberChannel == 1-1)
                if (params.mCh1Mode == 2)
                    recv(index, stat);
                if (channels[index].mInfo.numberChannel == 2-1)
                if (params.mCh2Mode == 2)
                    recv(index, stat);
            }
            for (int index = 0; index < countChannels; index++) {
                channels[index].mFileDescryptor =
a708_open(channels[index].mNameChannel);
                a708_getChannelInfo(channels[index].mFileDescryptor,
&(channels[index].mInfo));
                if (channels[index].mInfo.typeChannel != T_INFO_CHANNEL_TYPE_A708)
{
                    a708_close(channels[index].mFileDescryptor);
                    continue;
                }
                a708_close(channels[index].mFileDescryptor);
```

```
                if (channels[index].mInfo.numberChannel == 1-1)
                if (params.mCh1Mode == 1){
                    send(index, &blockA708, &blockA708R, stat);
                }
                if (channels[index].mInfo.numberChannel == 2-1)
                if (params.mCh2Mode == 1){
                    send(index, &blockA708, &blockA708R, stat);
                }
            }
            tvStart = tvCur;
        }
        gettimeofday(&tvCur, NULL);
        timersub(&tvCur, &tvStartPrint, &tvDiffPrint);
        if (tvDiffPrint.tv_sec * 1000000 + tvDiffPrint.tv_usec >=
(PRINT_TIMEOUT_MS /*mseconds*/ * 1000000)) { //проверка таймаута
            tvStartPrint = tvCur;
            LOGGER_PRINT("-has passed %d mseconds-\n", PRINT_TIMEOUT_MS);

                printStat(stat);
                clearStat(stat);

        }
        LOGGER_FLUSH();
    }
}

void initLogCTest3(void) {
    char filename[LOGGER_FILENAME_LENGTH];
    constructLogFilename(filename, LOGGER_FILENAME_LENGTH, "ctest3");
    LOGGER_OPEN(filename);
}

void initDefaultParamsCTest3(struct TParamCTest3* params) {
    params->mCh1Mode = 1;
    params->mCh2Mode = 2;
    params->mTimeoutSend = 1;
}
```

Листинг А.2 – ctest3.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 "../A708LinuxLibrary/a708_library.h"

#define VERSION_APP                     "25.03.2021"

/// \brief паттерн имени девайса
#define STR_CH0_PATTERN                 "-ch-0"
/// \brief паттерн имени канала
#define STR_DEV_PATTERN                 "a708dev"
/// \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 A429
/// \param channels массив описателей каналов
/// \param length длина массива
///
void getArrayChannels(struct TChannel** channels, int* length);
///
/// \brief getArrayChannelsA708 получить массив описателей каналов A708
/// \param channels массив описателей каналов
/// \param length длина массива
///
void getArrayChannelsA708(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)

# 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)

#endif // GLOBAL_H
```

Листинг А.3 – global.h

```c
#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 =
a708_open((*devices)[index].mNameChannel);
                VERSION ver;
                a708_getDeviceInfo((*devices)[index].mFileDescryptor, &ver);
                (*devices)[index].mDevId = ver.device_id;
                a708_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 =
a708_open((*channels)[index].mNameChannel);
                a708_getChannelInfo((*channels)[index].mFileDescryptor,
&((*channels)[index].mInfo));
                a708_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 if ((*channels)[index].mInfo.typeChannel ==
T_INFO_CHANNEL_TYPE_TRANSMITTER) {
                    LOGGER_PRINT("%s - num %d -
T_INFO_CHANNEL_TYPE_TRANSMITTER\n", (*channels)[index].mNameChannel,
(*channels)[index].mInfo.numberChannel);
                } else                 if ((*channels)[index].mInfo.typeChannel ==
T_INFO_CHANNEL_TYPE_A708) {
                    LOGGER_PRINT("%s - num %d - T_INFO_CHANNEL_TYPE_A708\n",
(*channels)[index].mNameChannel, (*channels)[index].mInfo.numberChannel);
                }
                index++;
            }
        }
        closedir(dir);
    }
}

void getArrayChannelsA708(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 =
a708_open((*channels)[index].mNameChannel);
                a708_getChannelInfo((*channels)[index].mFileDescryptor,
&((*channels)[index].mInfo));
                a708_close((*channels)[index].mFileDescryptor);
                // debug
                if ((*channels)[index].mInfo.typeChannel ==
T_INFO_CHANNEL_TYPE_A708) {
                    LOGGER_PRINT("%s - num %d - T_INFO_CHANNEL_TYPE_A708\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*					