УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2021г.

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

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

(OCWINDOWS)
(7, 8, 8.1, 10)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.15201-01 33 01-ЛУ

От	Инженер-программист
_____	_____ _____
«____»_____2021г.	«____»_____2021г.

2021

Из	*Под*	*Дат*

Литера

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

Утвержден

RU.МСКЮ.15201-01 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР А429» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429»

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

(OCWINDOWS)
(7, 8, 8.1, 10)

Руководство программиста

RU.МСКЮ.15201-01 33 01

Листов-32

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 6
1 НАЗНАЧЕНИЕ ПРОГРАММЫ 7
1.1 ДРАЙВЕР A429 7
1.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429 7
2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ 8
2.1 ДРАЙВЕР A429 8
2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429 8
3 ХАРАКТЕРИСТИКА ПРОГРАММЫ 9
3.1 ДРАЙВЕР A429 9
3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429 9
4 ОБРАЩЕНИЕ К ПРОГРАММЕ 10
4.1 ДРАЙВЕР A429 10
4.1.1 Чтение из регистра – IOCTL_READ_BAR 10
4.1.2 Запись в регистр – IOCTL_WRITE_BAR 10
4.1.3 Запись в регистр заданных бит –IOCTL_WRITE_BAR_MASK 11
4.1.4 Чтение RAM передатчика – IOCTL_READ_RAM 12
4.1.5 Запись вRAM передатчика – IOCTL_WRITE_RAM 12
4.1.6 Сбросуказателя DMA – IOCTL_CLEAN_DMA 13
4.1.7 Чтение данных DMA – IOCTL_READ_DMA 13
4.1.8 Получение информации об устройстве IOCTL_VERSION 14
4.1.9 Слежение за прерываниями – IOCTL_TRACKING_INT 15
4.1.10 Проверкапрерываний – IOCTL_CHECK_TRACKING_INT 15
4.1.11 Режим «передача по запросу» –IOCTL_START_AS_RK_IN 15
4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429 16
4.2.1 Получение списка устройств –a429_enumerate 16
4.2.2 Освобождение памяти – a429_free 17
4.2.3 Открытие символьного устройстваa429_open 17
4.2.4 Чтение регистра из памятиa429_read_reg 17
4.2.5 Запись регистра – a429_write_reg 18
4.2.6 Модификация бит регистра –a429_modify_reg 18
4.2.7 Чтение блоков DMA –a429_read_dma 19

4.2.8 Сбросуказателей DMA –a429_clear_dma 19
4.2.9 Получение информации об устройстве –a429_get_device_info 20
4.2.10 УправлениеDMA устройства – a429_manage_dma 20
4.2.11 УправлениеDMAканала – a429_manageChannelDma 20
4.2.12 Записать в RAMпередатчика – a429_writeRam 21
4.2.13 Прочитать из RAMпередатчика – a429_readRam 21
4.2.14 Слежение за входными РК– a429_manageTrackingInterrupt 22
4.2.15 Проверка прерывания РК – a429_checkTrackingInterrupt 22
4.2.16 Старт/стоп передатчика –a429_txStartStop 23
4.2.17 Однократный запуск передатчика – a429_txStartOnce 23
4.2.18 Установка режима работы передатчика –a429_txSetMode 23
4.2.19 Запись в FIFO передатчика –a429_writeTxFifo 24
4.2.20 Управление метками фильтрации –a429_setLabelConfReg 24
4.2.21 Установка скорости работы канала –a429_setSpeed 25
4.2.22 Разрешение работы канала –a429_manageEnBit 25
4.2.23 Проверка разрешения работы канала –a429_isEnBit 25
4.2.24 Управление битами чётности –a429_parityManage 26
4.2.25 Управление работой фильтра приёмника –a429_rxFiltration 26
4.2.26 Управление фильтром приёмника –a429_rxManageFiltrationLabel ... 26
4.2.27 Сброс меток фильтров приёмника –a429_rxClearFiltrationLabel 27
4.2.28 Управление битами SDI–a429_rxManageRcvSdi 27
4.2.29 Деинициализация канала –a429_deinit 27
4.2.30 Управление порядком хода бит метки – a429_manageReverce 28
4.2.31 Установка времени паузы – a429_txGapBits 28
4.2.32 Управление дискретностью таймера RRT – a429_txRrD 28
4.2.33 Управление таймером RRT – a429_txSkipRrt 29
4.2.34 Управление выходными РК – a429_manageScOut 29
4.2.35 Сброс разрешения прерывания входных РК – a429_clearScIntMask . 29
4.2.36 Установка разрешения прерывания a429_manageScIntMask 30
4.2.37 ЗапускпередатчикасRRT – a429_txStartWithRrt 30
4.2.38 Управление передачей по запросу для передатчика – a429_txSetRequestTransferByRkIn 30
4.2.39 Управление маской прерываний – a429_manageInterruptMask 31

4.2.40 Управление приёмом по готовности приёмника – a429_rxSetReceiveReadyByRkIn 31

4.2.41 Сброс RAM передатчика – a429_ramCls 31

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 является модулем ядра и предназначен для функционирования в ОС семейства Windows. Драйвер устанавливается в систему стандартными средствами ОС Windows.

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 А429

Драйвер является модулем ядра и предназначен для функционирования в ОС семейства Windows. Драйвер устанавливается в систему стандартными средствами ОС Windows.

2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429

Библиотека взаимодействия предназначена для функционирования в ОС Windows(7, 8, 8.1, 10)и встраивания в прикладное ПО для инициализации и управления модулями«xxx-429UDxx».

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 ДРАЙВЕР А429

Драйвер является модулем ядра ОС семейства Windows, разработан на языке С и поставляется в виде пакета для установки. Поддерживаются ОС версий 7, 8, 8.1, 10 и архитектуры Intel x86 и AMD64.

Драйвер реализует интерфейс класса `GUID_DEVINTERFACE_ARINC429` (`3bd2b180-d211-4d88-8f46-b73845cf0429`), что позволяет обнаружить все устройства в системе с помощью функции SetupDiGetClassDevs (см. документацию setupapi).

Также возможно обратиться к устройствам напрямую через символьное имя «\Device\nmARINC429%d», где «%d» - порядковый номер устройства.

Взаимодействие с каналами происходит посредством IOCTL-команд.

3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А429

Библиотека взаимодействия разработана на языке C++ в двух исполнениях: статическом и динамическом.

Для использования статической библиотеки необходимо подключить к разрабатываемому проекту заголовочный файл «libarinc429.h» и скомпилированную библиотеку «libarinc429.lib».

Для использования динамической библиотеки необходимо подключить к разрабатываемому проекту заголовочный файл «libarinc429dll.h» и скомпилированный заголовок библиотеки «libarinc429dll.lib». В комплект поставки разрабатываемого ПО необходимо включить скомпилированный файл библиотеки libarinc429_dll.dll.

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ДРАЙВЕР A429

Взаимодействие с драйвером происходит с помощью IOCTL-команд. Описание команд и типы данных представлены в файле ioctl.h.

4.1.1 Чтение из регистра – IOCTL_READ_BAR

Чтение одного регистра размером 32 бита из области памяти устройства.

```
/** чтениепамятиконтроллера */
#defineIOCTL_READ_BAR                                   CTL_CODE(FILE_DEVICE_CONTROLLER,
0x100, METHOD_BUFFERED, FILE_ANY_ACCESS)
/** записьпамятиконтроллера */
#defineIOCTL_WRITE_BAR                                  CTL_CODE(FILE_DEVICE_CONTROLLER,
0x101, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief чтение/записьрегистров
typedefstruct_BAR_REG_REQ{
      UINT32regAddr;     /// адресрегистра
      UINT32data;        /// данные
} BAR_REG_REQ, *PBAR_REG_REQ;
```

Рисунок 4.1.1 – Параметры команд IOCTL_READ_BARи IOCTL_WRITE_BAR

4.1.2 Запись в регистр – IOCTL_WRITE_BAR

Запись одного регистра размером 32 бита в область памяти устройства.

Описание параметров команды приведено на рисунке 4.1.1.

4.1.3 Запись в регистр заданных бит –IOCTL_WRITE_BAR_MASK

Модификация бит регистра памяти устройства.Команда позволяет изменить определённые биты в указанном регистре.

```
/** запись памяти контроллера с маской */
#defineIOCTL_WRITE_BAR_MASK                 CTL_CODE(FILE_DEVICE_CONTROLLER, 0x102,
METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief записьрегистрапомаске
typedefstruct_BAR_REG_REQ_MASK{
      UINT32regAddr;     /// адресрегистра
      UINT32data;        /// данные
      UINT32mask;        /// маска
} BAR_REG_REQ_MASK, *PBAR_REG_REQ_MASK;
```

Рисунок 4.1.3 – Параметры команды IOCTL_WRITE_BAR_MASK

4.1.4 Чтение RAM передатчика – IOCTL_READ_RAM

Команда позволяет считать блоки данныхиз памяти передатчиков.

```
/** чтениепамятиканала */
#defineIOCTL_READ_RAM
      CTL_CODE(FILE_DEVICE_CONTROLLER, 0x103, METHOD_BUFFERED, FILE_ANY_ACCESS)

/** записьпамятиканала */
#defineIOCTL_WRITE_RAM
      CTL_CODE(FILE_DEVICE_CONTROLLER, 0x104, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief размерблока
#defineRAM_REQ_MAX 256

#defineRAM_BLOCK_TYPE_DATA             1
#defineRAM_BLOCK_TYPE_DESCRIPTOR       2

/// \brief чтение/запись блока RAM передатчика
typedefstruct_RAM_REQ{
      UINT8 channel;                                  /// внутреннийномерканала
      UINT32 type;                              /// типинструкций (DATA - 1,
DESCRIPTOR - 2)
      UINT32offset;                                   /// смещение относительно
начала буфера
      UINT32length;                                   /// количество блоков данных
      UINT32dwords[RAM_REQ_MAX];              /// блоки данных
} RAM_REQ, *PRAM_REQ;
```

Рисунок 4.1.4 – Параметры команд IOCTL_READ_RAM и IOCTL_WRITE_RAM

4.1.5 Запись вRAM передатчика – IOCTL_WRITE_RAM

Команда позволяет записать блоки RAM передатчиков.

Описание параметров команды и структура данных приведенына рисунке 4.1.4.

4.1.6 Сбросуказателя DMA – IOCTL_CLEAN_DMA

Сбрасывает указатели чтения и записи DMA для всего устройства.

```
/** сброс DMA */
#defineIOCTL_CLEAN_DMA
	CTL_CODE(FILE_DEVICE_CONTROLLER, 0x200, METHOD_BUFFERED, FILE_ANY_ACCESS)
```

Рисунок 4.1.6 - Параметры команды IOCTL_CLEAN_DMA

4.1.7 Чтение данных DMA – IOCTL_READ_DMA

Позволяет считать блоки DMA конкретного канала.

```
/** чтение DMA */
#defineIOCTL_READ_DMA
	CTL_CODE(FILE_DEVICE_CONTROLLER, 0x201, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \briefioctl ответ с блоками DMA
typedefstruct_DMA_CONTAINER{
	UINT8 channel;                                  /// внутреннийномерканала
	UINT32 count;                                   /// количествоблоков
	DMA_BLOCK blocks[DMA_COUNT_BLOCKS];   /// массивблоков
} DMA_CONTAINER, *PDMA_CONTAINER;

/// \brief ioctlзапросблоков DMA
typedefstruct_DMA_REQ{
	UINT8 channel;                                  /// внутреннийномерканала
	UINT32 count;                                   ///
	желаемоеколичествоблоков
} DMA_REQ, *PDMA_REQ;
```

Рисунок 4.1.7 – Параметры команды IOCTL_READ_DMA

4.1.8 Получение информации об устройстве IOCTL_VERSION

Получение подробной информации об устройстве.

```
/** получение версии устройства и драйвера */
#defineIOCTL_VERSION
      CTL_CODE(FILE_DEVICE_CONTROLLER, 0x300, METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief версиядрайвераиустройства
typedefstruct_VERSION_REQ{
      UINT32device_id;                /// id устройства
      UINT32vendor_id;                /// id производителя
      UINT32revision_id;              /// ревизияустройства

      UINT64mbar_addr;                /// адрес BAR (память)
      UINT64pbar_addr;                /// адрес BAR (порт)
      UINT32irq;                        /// векторпрерываний
      UINT64dma_size;                 /// размербуфера DMA
      LARGE_INTEGERdma_addr_device;       /// адрес DMA дляустройства
      UINT64dma_addr_driver;                  /// адрес DMA длядрйавера

      UINT32driver_version;           /// версиядрайвера
      UINT32driver_build_date; /// датасборкидрайвера

      UINT8rx_channel_count;    /// количество каналов-приёмников
      UINT8tx_channel_count;    /// количество каналов-передатчиков
      UINT8sc_tx_count;         /// количество приёмников рк
      UINT8sc_rx_count;         /// количество передатчиков рк
      UINT32g_channel_offset; /// смещение номера первого канала устройства

} VERSION_REQ, *PVERSION_REQ;
```

Рисунок 4.1.8 - Параметры команды IOCTL_VERSION

4.1.9 Слежение за прерываниями – IOCTL_TRACKING_INT

Позволяет включить/выключить функцию слежения за прерываниями.

```
#defineIOCTL_TRACKING_INT                    CTL_CODE(FILE_DEVICE_CONTROLLER,
0x304, METHOD_BUFFERED, FILE_ANY_ACCESS)

#defineIOCTL_CHECK_TRACKING_INT       CTL_CODE(FILE_DEVICE_CONTROLLER, 0x305,
METHOD_BUFFERED, FILE_ANY_ACCESS)

#defineIOCTL_START_AS_RK_IN                  CTL_CODE(FILE_DEVICE_CONTROLLER,
0x306, METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief управлениеprерываниями
typedefstruct_T_INT_REQ{
      UINT8 channel;
      /// внутреннийномерканала
      UINT32 manage;
      /// вкл/выклслежение
      UINT64startFreeTimer;                                          ///
начальноезначениеfreetimer
      UINT64finishFreeTimer;                                         ///
конечноезначениеfreetimer
      UINT32rkInAppearCount[T_CHECK_TRACKING_INT_CNT];   ///
количествовозникновенийРК
} T_INT_REQ, *PT_INT_REQ;
```

Рисунок 4.1.10 - Параметры команд :IOCTL_TRACKING_INT, IOCTL_CHECK_TRACKING_INT, IOCTL_START_AS_RK_IN

4.1.10 Проверкапрерываний – IOCTL_CHECK_TRACKING_INT

Проверка наличия возникновения прерываний с момента предыдущей проверки при включенной функции слежения за прерываниями.

Описание параметров команды и структура данных приведены на рисунке 4.1.9.

4.1.11 Режим «передача по запросу» –IOCTL_START_AS_RK_IN

Запуск передатчика в режиме "передача по запросу".

Описание параметров команды и структура данных приведены на рисунке 4.1.9.

4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A429

Библиотека предназначена для функционирования в ОС семейства Windows и встраивания в прикладные приложения. В файле "libarinc429.h" представлены сервисные функции библиотеки взаимодействия.

Все функции библиотеки кроме a429_enumerate и a429_free возвращают код ошибки или ERROR_SUCCESSв случае успеха.

4.2.1 Получение списка устройств –a429_enumerate

Функция запрашивает у системы список устройств, реализующих интерфейс A429, опрашивает каждое устройство и формирует списки устройств и каналов.

```
/// \brief channel info
typedefstructA429_CHANNEL_INFO {
	WCHAR* device_name;		/// имя файла устройства
	UINT32g_channel;		/// глобальный номер канала
	UINT32i_channel;		/// номер канала данного типа
	UINT32b_channel;		/// номер канала на плате
	UINT32g_board;			/// номерплаты
	UINT32channel_type;		/// типканала (T_CHANNEL_RX или
T_CHANNEL_TX)
	HANDLEarinc;			/// хэндлустройства
};

typedefstructA429_DEVICE_LIST {
	UINT32count;			/// количество каналов
	UINT32pcount;			/// количество устройств

	A429_CHANNEL_INFO* device;	/// списокканалов
	A429_CHANNEL_INFO* pdev; /// списокустройств
};
/**
* @brief получение списка устройств
*
* Функция получит у системы все устройства реализующие интерфейс с ID
GUID_DEVINTERFACE_ARINC429,
* откроет каждое, получит информацию о нём и сформирует два списка - устройств и
каналов. Структуру
* A429_DEVICE_LIST необходимо будет освободить с помощью a429_free.
*
*
* @return указатель на структуру A429_DEVICE_LIST
*/
A429_DEVICE_LIST* a429_enumerate();
...
A429_DEVICE_LIST* devices;
devices = a429_enumerate(); //получитьсписокустройств
```

Рисунок 4.2.1 - Параметры команды a429_enumerate

4.2.2 Освобождение памяти – a429_free

Корректное освобождение памяти, занятой списком устройств и каналов.

```
VOID a429_free(A429_DEVICE_LIST* devices);

...

a429_free(devices); //освободить список устройств
```

Рисунок 4.2.2 - Параметры команды a429_free

4.2.3 Открытие символьного устройстваa429_open

Открытие символьного устройства.

```
/**
* @brief открытие символьного устройства
*
* Функция попытается открыть описанное структурой устройство и запишет хэндл
обратно в структуру A429_CHANNEL_INFO.
* После завершения использования конкретного канала или устройства хэндл
необходимо закрыть с помощью CloseHandle.
*
* @param указатель на структуру A429_CHANNEL_INFO
*
* @returnхэндл устройства или NULL, если произошла ошибка
*/
HANDLE a429_open(A429_CHANNEL_INFO* info);
...
a429_open(devices->device + 0);
CloseHandle(devices->device[0].arinc);
```

Рисунок 4.2.3 - Параметры команды a429_open

4.2.4 Чтение регистра из памятиa429_read_reg

Чтение регистра из памяти.

```
/**
* @brief Чтение регистра устройства
*
* @paramarinc указатель на структуру устройства
* @paramaddr адрес
* @parampbuf указатель на UINT32, куда будет записано содержимое регистра
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_read_reg(A429_CHANNEL_INFO* arinc, UINT32addr, UINT32* pbuf);
...
UINT32 buf;
a429_read_reg(devices->device + 0, 0x1010, &buf); //прочитатьрегистр 0x1010
(INTERRUPT_MASK) впеременнуюbuf
```

Рисунок 4.2.4 - Параметры команды a429_read_reg

4.2.5 Запись регистра – a429_write_reg

Запись значения регистра в память.

```
/**
* @brief Запись регистра устройства
*
* @paramarinc указатель на структуру устройства
* @paramaddr адрес
* @parambuf содержимое регистра
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_write_reg(A429_CHANNEL_INFO* arinc, UINT32addr, UINT32buf);
...
a429_write_reg(devices->device + 0, 0x1010, 0x0); //записать 0x0 врегистр 0x1010
(INTERRUPT_MASK)
```

Рисунок 4.2.5 - Параметры команды a429_write_reg

4.2.6 Модификация бит регистра –a429_modify_reg

Модификация отдельных бит регистра.

```
/**
* @brief Изменение регистра устройства
*
* @paramarinc указатель на структуру устройства
* @paramaddr адрес
* @parambuf содержимое регистра
* @parammask маска
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_modify_reg(A429_CHANNEL_INFO* arinc, UINT32addr, UINT32buf,
UINT32mask);
...
a429_modify_reg(devices->device + 0, 0x1010, 1 << 8 , 1 << 8); //установить 8
битрегистра INTERRUPT_MASK в 1
```

Рисунок 4.2.6 - Параметры команды a429_modify_reg

4.2.7 Чтение блоков DMA –a429_read_dma

Функция запросит не более pcount блоков из буфера dma. В случае успеха в переменнуюdmacount будет записано количество реально считанных блоков, а сами блоки - в заранее созданный массив dma.

```
/**
* @brief чтение DMA
*
* @paramarinc указатель на структуру канала
* @paramchannel внутренний номер канала
* @parampcount указатель на количество блоков для чтения, после выполнения сюда
будет записано количество считанных блоков
* @paramdmabuf указатель на буфер, куда будут записаны блоки DMA
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_read_dma(A429_CHANNEL_INFO* arinc, UINT32* pcount, DMA_BLOCK**
dmabuf);
...
DMA_BLOCK *dma = new DMA_BLOCK[DMA_COUNT_BLOCKKS];
UINT32 dmacount = 256;
a429_read_dma(devices->device + i, &dmacount, &dma);
for(inti=0;i<dmacount;i++){ ...} //обработкаблоков
delete[] dma;
```

Рисунок 4.2.7 - Параметры команды a429_read_dma

4.2.8 Сбросуказателей DMA –a429_clear_dma

Сбросуказателей DMA устройства.

```
/**
* @brief сброс DMA
*
* Полный сброс DMA для всех каналов устройства
*
* @paramarinc указатель на структуру устройства
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_clear_dma(A429_CHANNEL_INFO* arinc);
...
a429_clear_dma(devices->pdev + i);
```

Рисунок 4.2.8 - Параметры команды a429_read_dma

4.2.9 Получение информации об устройстве –a429_get_device_info

Получение информации об устройстве.

```
/**
* @brief получение данных об устройстве
*
* @paramarinc указатель на структуру устройства
* @paramversion указатель на структуру куда будут записаны данные об устройстве
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_get_device_info(A429_CHANNEL_INFO* arinc, VERSION_REQ* version);
...
VERSION_REQ ver;
a429_get_device_info(devices->pdev + i, &ver);
```

Рисунок 4.2.9 - Параметры команды a429_get_device_info

4.2.10 УправлениеDMA устройства – a429_manage_dma

Включение или выключение DMA всего модуля.

```
/**
* @brief управление DMA устройства
*
* @paramarinc указатель на структуру устройства
* @param value A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageDma(A429_CHANNEL_INFO* arinc, UINT32value);
...
a429_manageDma(devices->pdev + 0, 1); //включить DMA
```

Рисунок 4.2.10 - Параметры команды a429_manage_dma

4.2.11 УправлениеDMAканала – a429_manageChannelDma

Управление DMA конкретного канала.

```
/**
* @brief управление DMA канала
*
* @paramarinc указатель на структуру канала
* @param value A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageChannelDma(A429_CHANNEL_INFO* arinc, UINT32value);
...
a429_manageDma(devices->device + 5, 1); //включить DMA на 5 канале
```

Рисунок 4.2.11 - Параметры команды a429_manageChannelDma

4.2.12 Записать в RAMпередатчика – a429_writeRam

Записьблоков RAM передатчика.

```
typedefstruct_RAM_REQ{
	UINT8 channel;                                  /// внутреннийномерканала
	UINT32 type;                                /// типинструкций (DATA - 1,
DESCRIPTOR - 2)
	UINT32offset;                                   /// смещение относительно
начала буфера
	UINT32length;                                   /// количество блоков данных
	UINT32dwords[RAM_REQ_MAX];                  /// блоки данных
} RAM_REQ, *PRAM_REQ;

/**
* @brief запись RAM передатчика
*
* @paramarinc указатель на структуру канала
* @paramcontainer указатель на структуру RAM_REQ с данными, которые будут записаны
в память
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_writeRam(A429_CHANNEL_INFO* arinc, RAM_REQ* container);
...
RAM_REQ *desc = newRAM_REQ;
loadMem(desc, "mem/desk_test_consol.mem");
a429_writeRam(devices->device + 0, desc);
```

Рисунок 4.2.12 - Параметры команды a429_writeRam

4.2.13 Прочитать из RAMпередатчика – a429_readRam

Чтение блоков RAM передатчика.

```
/**
* @brief запись RAM передатчика
*
* @paramarinc указатель на структуру канала
* @paramcontainer указатель на структуру RAM_REQ, куда будут записаны считанные
данные
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_readRam(A429_CHANNEL_INFO* arinc, RAM_REQ* container);
...
RAM_REQ *desc = newRAM_REQ;
a429_writeRam(devices->device + 0, desc);
```

Рисунок 4.2.13 - Параметры команды a429_readRam

4.2.14 Слежение за входными РК– a429_manageTrackingInterrupt

Управление слежением за входными РК.

```
/// \brief управление слежением за прерываниями
typedefstruct_T_INT_REQ{
	UINT8channel;
	/// внутреннийномерканала
	UINT32manage;
	/// вкл/выклслежение
	UINT64startFreeTimer;                                                    ///
начальноезначениеfreetimer
	UINT64finishFreeTimer;                                                   ///
конечноезначениеfreetimer
	UINT32rkInAppearCount[T_CHECK_TRACKING_INT_CNT];   ///
количествовозникновенийРК
} T_INT_REQ, *PT_INT_REQ;

/**
* @brief Управление слежением за входными РК
*
* @paramarinc указатель на структуру устройства
* @paraminfo указатель на структуру T_INT_REQ
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageTrackingInterrupt(A429_CHANNEL_INFO* arinc, T_INT_REQ* info);
...
T_INT_REQtreq;
treq.manage = T_TRACKING_INT_MAN_ON;
a429_manageTrackingInterrupt(devices->pdev + i, &treq);
```

Рисунок 4.2.14 - Параметры команды a429_manageTrackingInterrupt

4.2.15 Проверка прерывания РК – a429_checkTrackingInterrupt

Проверкавозникновения прерывания повходнымРК.

```
/**
* @brief Управление слежением за входными РК
*
* @paramarinc указатель на структуру устройства
* @paraminfo указатель на структуру T_INT_REQ, куда будет записано состояние РК
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_checkTrackingInterrupt(A429_CHANNEL_INFO* arinc, T_INT_REQ* info);
...
T_INT_REQtreq;
a429_checkTrackingInterrupt(devices->pdev + i, &treq);
```

Рисунок 4.2.15 - Параметры команды a429_checkTrackingInterrupt

4.2.16 Старт/стоп передатчика –a429_txStartStop

Старт/стоп передатчика.

```
/**
* @brief включение или выключение передачи
*
* @paramarinc указатель на структуру канала
* @param action A429_STOP или A429_START
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txStartStop(A429_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.16 - Параметры команды a429_txStartStop

4.2.17 Однократный запуск передатчика – a429_txStartOnce

Однократный запуск передатчика.

```
/**
* @brief однократный запуск передатчика
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txStartOnce(A429_CHANNEL_INFO* arinc);
```

Рисунок 4.2.17 - Параметры команды a429_txStartOnce

4.2.18 Установка режима работы передатчика –a429_txSetMode

Установка режима работы передатчика.

```
#defineA429_MODE_0          0               ///< режим передатчика 0
#defineA429_MODE_1          1               ///< режим передатчика 1
#defineA429_MODE_2          2               ///< режим передатчика 2
#defineA429_MODE_3          3               ///< режим передатчика 3

/**
* @brief установка режима работы передатчика
*
* @paramarinc указатель на структуру канала
* @parammode режим работы
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txSetMode(A429_CHANNEL_INFO* arinc, UINT32mode);
```

Рисунок 4.2.18 - Параметры команды a429_txSetMode

4.2.19 Запись в FIFO передатчика –a429_writeTxFifo

Записьданныхвбуфер FIFO передатчика.

```
/**
* @brief запись данных в буфер FIFO передатчика
*
* @paramarinc указатель на структуру канала
* @paramdata слово данных
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_writeTxFifo(A429_CHANNEL_INFO* arinc, UINT32data);
```

Рисунок 4.2.19 - Параметры команды a429_writeTxFifo

4.2.20 Управление метками фильтрации –a429_setLabelConfReg

Управлениеметкамифильтрации.

```
/**
* @brief Управление метками фильтрации
*
* @paramarinc указатель на структуру канала
* @paramnumReg номер регистра REG_LBL_CONF_PCI_0 - REG_LBL_CONF_PCI_7
* @parammask маска
* @paramvalue значение
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_setLabelConfReg(A429_CHANNEL_INFO* arinc, UINT32numReg, UINT32mask,
UINT32value);
```

Рисунок 4.2.20 - Параметры команды a429_setLabelConfReg

4.2.21 Установка скорости работы канала –a429_setSpeed

Установка скорости работы канала.

```
#defineA429_SPEED_100          0       ///< 100 кбит/с
#defineA429_SPEED_12_14        1       ///< 12.5 кбит/с
#defineA429_SPEED_50           2       ///< 50 кбит/с
#defineA429_SPEED_12           3       ///< 12 кбит/с
#defineA429_SPEED_14_5         4       ///< 14.5 кбит/с
/**
* @brief установка скорости работы канала
*
* @paramarinc указатель на структуру канала
* @param speed скорость (A429_SPEED_...)
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_setSpeed(A429_CHANNEL_INFO* arinc, UINT32speed);
```

Рисунок 4.2.21 - Параметры команды a429_setSpeed

4.2.22 Разрешение работы канала –a429_manageEnBit

Разрешение работы канала.

```
#defineA429_STOP           0                ///<стоп
#defineA429_START          ~A429_STOP///<старт
/**
* @brief разрешение работы канала
*
* @paramarinc указатель на структуру канала
* @param action A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageEnBit(A429_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.22 - Параметры команды a429_manageEnBit

4.2.23 Проверка разрешения работы канала –a429_isEnBit

Проверка разрешения работы канала.

```
/**
* @brief проверка разрешение работы канала
*
* @paramarinc указатель на структуру канала
* @paramstate указатель на переменную, куда будет записано текущее состояние
разрешения работы канала - A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_isEnBit(A429_CHANNEL_INFO* arinc, UINT32* state);
```

Рисунок 4.2.23 - Параметры команды a429_isEnBit

4.2.24 Управление битами чётности –a429_parityManage

Управлениебитамичётности.

```
/**
* @brief управление битами чётности
*
* @paramarinc указатель на структуру канала
* @paramparcheck бит 30 регистра RX_CONF_REG или TX_CONF_REG
* @paramparity бит 29 регистра RX_CONF_REG или TX_CONF_REG
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_parityManage(A429_CHANNEL_INFO* arinc, UINT32parcheck, UINT32parity);
```

Рисунок 4.2.24 - Параметры команды a429_parityManage

4.2.25 Управление работой фильтра приёмника –a429_rxFiltration

Управлениеработойфильтраприёмника.

```
#defineA429_OFF             0                ///<выкл
#defineA429_ON              ~A429_OFF///<вкл
/**
* @brief управление работой фильтра приёмника
*
* @paramarinc указатель на структуру канала
* @param action A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_rxFiltration(A429_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.25 - Параметры команды a429_rxFiltration

4.2.26 Управление фильтром приёмника –a429_rxManageFiltrationLabel

Управлениеметкамифильтраприёмника.

```
#defineA429_OFF             0                ///<выкл
#defineA429_ON              ~A429_OFF///<вкл
/**
* @brief управление метками фильтра приёмника
*
* @paramarinc указатель на структуру канала
* @param label метка
* @param action A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_rxManageFiltrationLabel(A429_CHANNEL_INFO* arinc, UINT32label,
UINT32action);
```

Рисунок 4.2.26 - Параметры команды a429_rxManageFiltrationLabel

4.2.27 Сброс меток фильтров приёмника –a429_rxClearFiltrationLabel

Сбросметокфильтровприёмника.

```
/**
* @brief сброс меток фильтра приёмника
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_rxClearFiltrationLabel(A429_CHANNEL_INFO* arinc);
```

Рисунок 4.2.27 - Параметры команды a429_rxClearFiltrationLabel

4.2.28 Управление битами SDI–a429_rxManageRcvSdi

Управлениебитами 9 и 10 приёмника.

```
/**
* @brief Управление битами 9 и 10 приёмника
*
* @paramarinc указатель на структуру канала
* @paramaction включение или выключение фильтрации поля RCV_SDI
* @param bit9 значение бита 9
* @param bit10 значение бита 10
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_rxManageRcvSdi(A429_CHANNEL_INFO* arinc, UINT32action, UINT32bit9,
UINT32bit10);
```

Рисунок 4.2.28 - Параметры команды a429_rxManageRcvSdi

4.2.29 Деинициализация канала –a429_deinit

Деинициализация канала.

```
/**
* @briefдеинициализация канала
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_deinit(A429_CHANNEL_INFO* arinc);
```

Рисунок 4.2.29 - Параметры команды a429_deinit

4.2.30 Управление порядком хода бит метки – a429_manageReverce

Управлениепорядкомбитдлясловаданных (бит 28 регистровREG_RX_CONF, REG_TX_CONF).

```
/**
* @brief управление порядком бит для слова данных
*
* @paramarinc указатель на структуру канала
* @param action A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageReverce(A429_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.30 - Параметры команды a429_manageReverce

4.2.31 Установка времени паузы – a429_txGapBits

Управлениевременемпаузы, биты 19-25 регистраREG_TX_CONF.

```
/**
* @brief управление временем паузы
*
* @paramarinc указатель на структуру канала
* @paramgapBit время паузы, биты 19-25 регистра REG_TX_CONF
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txGapBits(A429_CHANNEL_INFO* arinc, UINT32gapBit);
```

Рисунок 4.2.2 - Параметры команды a429_txGapBits

4.2.32 Управление дискретностью таймера RRT – a429_txRrD

Управление дискретностью таймера RRT.

```
#defineA429_TXRR_10MS      0       ///<txrr 10 ms
#defineA429_TXRR_1MS       1       ///<txrr 1 ms
/**
* @brief управление дискретностью таймера RRT
*
* @paramarinc указатель на структуру канала
* @paramtxrr A429_TXRR_10MS или A429_TXRR_1MS
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txRrD(A429_CHANNEL_INFO* arinc, UINT32txrr);
```

Рисунок 4.2.32 - Параметры команды a429_txRrD

4.2.33 Управление таймером RRT – a429_txSkipRrt

Управлениетаймером RRT.

```
/**
* @brief управление таймером RRT
*
* @paramarinc указатель на структуру канала
* @param action A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txSkipRrt(A429_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.33 - Параметры команды a429_txSkipPart

4.2.34 Управление выходными РК – a429_manageScOut

Управление выходными РК.

```
#defineA429_SC_OUT_1            (1)         ///<ВЫХРКв 0
#defineA429_SC_OUT_0            (1<<1)      ///<ВЫХРКв 1
#defineA429_SC_OUT_NO_ACT       0           ///<ВЫХРКнетрогать
/**
* @brief управление выходными РК
*
* @paramarinc указатель на структуру канала
* @param action1 управление выходом РК 1
* @param action2 управление выходом РК 2
* @param action3 управление выходом РК 3
* @param action4 управление выходом РК 4
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageScOut(A429_CHANNEL_INFO* arinc, UINT32action1, UINT32action2,
UINT32action3, UINT32action4);
```

Рисунок 4.2.34 - Параметры команды a429_manageScOut

4.2.35 Сброс разрешения прерывания входных РК – a429_clearScIntMask

Сбросразрешения перрываниявыходныхРК.

```
/**
* @brief сброс разрешений выходных РК
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_clearScIntMask(A429_CHANNEL_INFO* arinc);
```

Рисунок 4.2.35 - Параметры команды a429_clearScIntMask

4.2.36 Установка разрешения прерывания a429_manageScIntMask

Установка разрешений прерывания выходных РК.

```
/**
* @brief установка разрешений выходных РК
*
* @paramarinc указатель на структуру канала
* @paramnumRk      номерканалаРК
* @paramstateRk A429_SC_RISE или A429_SC_FALL
* @paramactionRk A429_OFF или A429_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageScIntMask(A429_CHANNEL_INFO* arinc, UINT32numRk, UINT32stateRk,
UINT32actionRk);
```

Рисунок 4.2.36 - Параметры команды a429_clearScIntMask

4.2.37 ЗапускпередатчикасRRT – a429_txStartWithRrt

Однократный запуск передатчика с RRT.

```
/**
* @brief однократный запуск передатчика с RRT
*
* @paramarinc указатель на структуру канала
* @paramrrtValue значение таймера RRT
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txStartWithRrtOnce(A429_CHANNEL_INFO* arinc, UINT32rrtValue);
```

Рисунок 4.2.37 - Параметры команды a429_txStartWithRrt

4.2.38 Управление передачей по запросу для передатчика – a429_txSetRequestTransferByRkIn

Управлениепередачейпозапросудляпередатчика.

```
/**
* @brief Управление передачей по запросу для передатчика
*
* @paramarinc указатель на структуру канала
* @paramrknum номер канала РК
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_txSetRequestTransferByRkIn(A429_CHANNEL_INFO* arinc, UINT32rknum);
```

Рисунок 4.2.38 - Параметры команды a429_txSetRequestTransferByRkIn

4.2.39 Управление маской прерываний – a429_manageInterruptMask

Управлениемаскойпрерываний.

```
/**
* @brief Управление маской прерываний
*
* @paramarinc указатель на структуру канала
* @parammask маска
* @paramvalue значение
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_manageInterruptMask(A429_CHANNEL_INFO* arinc, UINT32mask,
UINT32value);
```

Рисунок 4.2.39 - Параметры команды a429_manageInterruptMask

4.2.40 Управление приёмом по готовности приёмника – a429_rxSetReceiveReadyByRkIn

Управлениеприёмомпоготовностиприёмника.

```
/**
* @brief Управление приёмом по готовности приёмника
*
* @paramarinc указатель на структуру канала
* @paramrknum номер канала РК
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_rxSetReceiveReadyByRkIn(A429_CHANNEL_INFO* arinc, UINT32rknum);
```

Рисунок 4.2.40 - Параметры команды a429_rxSetReceiveReadyByRkIn

4.2.41 Сброс RAM передатчика – a429_ramCls

Сброс RAM передатчика.

```
/**
* @brief сброс RAM передатчика
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a429_ramCls(A429_CHANNEL_INFO* arinc);
```

Рисунок 4.2.41 - Параметры командыа429_ramCls

Лист регистрации

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