УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2021г.

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

Модулей
“PCIe–708UD2”
“mPCIe–708UD2”

(ОС WINDOWS)
(Windows 7/8/10)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.24201-01 33 01-ЛУ

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

2021

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

Из	*Под*	*Дат*

Литера

Утвержден

RU.МСКЮ.24301-01 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР А708» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А708»

Модулей
“PCIe–708UD2”
“mPCIe–708UD2”

(ОС WINDOWS)
(Windows 7/8/10)

Руководство программиста

RU.МСКЮ.24201-01 33 01

Листов-35

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 А708» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ А708». В частности, рассмотрены функциональное назначение и область применения, условия выполнения.

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ 6
1 НАЗНАЧЕНИЕ ПРОГРАММЫ 7
1.1 ДРАЙВЕР A708 7
1.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708 7
2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ 8
2.1 ДРАЙВЕР A708 8
2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708 8
3 ХАРАКТЕРИСТИКА ПРОГРАММЫ 9
3.1 ДРАЙВЕР A708 9
3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708 9
4 ОБРАЩЕНИЕ К ПРОГРАММЕ 10
4.1 ДРАЙВЕР A708 10
4.1.1 Запись в регистр - IOCTL_WRITE_BAR 10
4.1.2 Запись битовой маски в регистр - IOCTL_WRITE_BAR_MASK 10
4.1.3 Чтение из регистра - IOCTL_READ_BAR 11
4.1.4 Чтение из ДМА - IOCTL_READ_DMA 11
4.1.5 Чтение из ДМА - IOCTL_READ_DMA_708 12
4.1.6 Сброс ДМА - IOCTL_CLEAN_DMA 12
4.1.7 Получить информацию о плате - IOCTL_VERSION 13
4.1.8 Запись блока в RAM - IOCTL_WRITE_RAM 14
4.1.9 Чтение блока из RAM - IOCTL_READ_RAM 15
4.1.10 Вкл/выкл слежения за прерываниями - IOCTL_TRACKING_INT 16
4.1.11 Проверка возникновения прерываний РК - IOCTL_CHECK_TRACKING_INT 17
4.1.12 Режим «передача по запросу» - IOCTL_START_AS_RK_IN 17
4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708 18
4.2.1 Получение списка устройств –a708_enumerate 18
4.2.2 Освобождение памяти – a708_free 19
4.2.3 Открытие символьного устройства a708_open 19
4.2.4 Чтение регистра из памяти a708_read_reg 19
4.2.5 Запись регистра – a708_write_reg 20

4.2.6 Модификация бит регистра –a708_modify_reg 20
4.2.7 Чтение блоков DMA –a708_read_dma 21
4.2.8 Сброс указателей DMA –a708_clear_dma 21
4.2.9 Получение информации об устройстве –a708_get_device_info 22
4.2.10 УправлениеDMA устройства – a708_manage_dma 22
4.2.11 Управление DMA канала – a708_manageChannelDma 22
4.2.12 Записать в RAMпередатчика – a708_writeRam 23
4.2.13 Прочитать из RAM передатчика – a708_readRam 23
4.2.14 Слежение за входными РК– a708_manageTrackingInterrupt 24
4.2.15 Проверка прерывания РК – a708_checkTrackingInterrupt 24
4.2.16 Старт/стоп передатчика –a708_txStartStop 24
4.2.17 Однократный запуск передатчика – a708_txStartOnce 25
4.2.18 Установка режима работы передатчика –a708_txSetMode 25
4.2.19 Запись в FIFO передатчика –a708_writeTxFifo 25
4.2.20 Управление метками фильтрации –a708_setLabelConfReg 25
4.2.21 Установка скорости работы канала –a708_setSpeed 26
4.2.22 Разрешение работы канала –a708_manageEnBit 26
4.2.23 Проверка разрешения работы канала –a708_isEnBit 26
4.2.24 Управление битами чётности –a708_parityManage 27
4.2.25 Управление работой фильтра приёмника –a708_rxFiltration 27
4.2.26 Управление фильтром приёмника –a708_rxManageFiltrationLabel ... 27
4.2.27 Сброс меток фильтров приёмника –a708_rxClearFiltrationLabel 28
4.2.28 Управление битами SDI–a708_rxManageRcvSdi 28
4.2.29 Деинициализация канала –a708_deinit 28
4.2.30 Управление порядком хода бит метки – a708_manageReverce 28
4.2.31 Установка времени паузы – a708_txGapBits 29
4.2.32 Управление дискретностью таймера RRT – a708_txRrD 29
4.2.33 Управление таймером RRT – a708_txSkipRrt 29
4.2.34 Управление выходными РК – a708_manageScOut 29
4.2.35 Сброс разрешения прерывания входных РК – a708_clearScIntMask . 30
4.2.36 Установка разрешения прерывания a708_manageScIntMask 30
4.2.37 Запуск передатчика с RRT – a708_txStartWithRrt 30

4.2.38 Управление передачей по запросу для передатчика – a708_txSetRequestTransferByRkIn 31
4.2.39 Управление маской прерываний – a708_manageInterruptMask 31
4.2.40 Управление приёмом по готовности приёмника – a708_rxSetReceiveReadyByRkIn 31
4.2.41 Сброс RAM передатчика – a**708**_ramCls 31
4.2.42 Управление режимом работы приёмника канала Arinc708 – a708_rx708_manage 32
4.2.43 Управление режимом работы передатчика канала Arinc708 – a708_tx708_setMode 32
4.2.44 Управление режимом работы ShortDMA приёмника канала Arinc708 – a708_rx708_shortDma 32
4.2.45 Чтение буфера DMA канала Arinc708 – a708_readDmaA708 33
4.2.46 Запись блока данных передатчика – Arinc708 a708_tx708_writeDataBlock 33
4.2.47 Чтение блока данных передатчика – Arinc708 a708_tx708_readDataBlock 33
4.2.48 Запуск передачи – Arinc708 a708_tx708_start 34
4.2.49 Проверка состояния буфера передачи Arinc708 – a708_tx708_hasSendedDataBlock 34
4.2.50 Остановка передачи – Arinc708 a708_tx708_stop 34
4.2.51 Установка таймера – Arinc708 a708_tx708_writeTimer 34

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 семейства Windows (7, 8, 8.1, 10). Драйвер устанавливается в систему стандартными средствами ОС Windows.

2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708

Библиотека взаимодействия предназначена для функционирования в ОС семейства Windows (7, 8, 8.1, 10) 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 семейства Windows, разработан на языке C 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_ARINC708` (`3bd2b180-d211-4d88-8f46-b73845cf0430`), что позволяет обнаружить все устройства в системе с помощью функции SetupDiGetClassDevs (см. документацию setupapi).

Также возможно обратиться к устройствам напрямую через символьное имя «\Device\nmARINC708%d», где «%d» - порядковый номер устройства.

Взаимодействие с каналами происходит посредством IOCTL-команд.

3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708

Библиотека взаимодействия разработана на языке C++.

Для использования библиотеки необходимо подключить к разрабатываемому проекту заголовочный файл «libarinc708.h» и скомпилированную библиотеку «libarinc708.lib».

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ДРАЙВЕР A708

Взаимодействие с каналами происходит посредством ioctl-команд, описание команд и типы данных представлены в файле «ioctl.h.

4.1.1 Запись в регистр - IOCTL_WRITE_BAR

Позволяет записать значение в заданный регистр управления.

Описание параметров команды приведено на рисунке 4.1.1.

```
/** запись памяти контроллера */
#define IOCTL_WRITE_BAR                                 CTL_CODE(FILE_DEVICE_CONTROLLER,
0x101, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief чтение/запись регистров
Typedef struct _BAR_REG_REQ{
       UINT32 regAddr;      /// адрес регистра
       UINT32 data;         /// данные
} BAR_REG_REQ, *PBAR_REG_REQ;
```

Рисунок 4.1.1 – Листинг команды

4.1.2 Запись битовой маски в регистр - IOCTL_WRITE_BAR_MASK

Позволяет записать значение заданных бит в заданный регистр управления.

Описание параметров команды приведено на рисунке 4.1.2.

```
/** запись памяти контроллера с маской */
#define IOCTL_WRITE_BAR_MASK                    CTL_CODE(FILE_DEVICE_CONTROLLER, 0x102,
METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief запись регистра по маске
Typedef struct _BAR_REG_REQ_MASK{
       UINT32 regAddr;      /// адрес регистра
       UINT32 data;         /// данные
       UINT32 mask;         /// маска
} BAR_REG_REQ_MASK, *PBAR_REG_REQ_MASK;
```

Рисунок 4.1.2 – Листинг команды

4.1.3 Чтение из регистра - IOCTL_READ_BAR

Позволяет прочитать значение из заданного регистра управления.

Описание параметров команды приведено на рисунке 4.1.3.

```
/** чтение памяти контроллера */
#define IOCTL_READ_BAR                                  CTL_CODE(FILE_DEVICE_CONTROLLER,
0x100, METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief чтение/запись регистров
Typedef struct _BAR_REG_REQ{
        UINT32 regAddr;      /// адрес регистра
        UINT32 data;         /// данные
} BAR_REG_REQ, *PBAR_REG_REQ;
```

Рисунок 4.1.3 – Листинг команды

4.1.4 Чтение из ДМА - IOCTL_READ_DMA

Позволяет прочитать данные из ДМА каналаARINC-708.

Описание параметров команды приведено на рисунке 4.1.4.

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

/// \brief ioctl ответ с блоками DMA
Typedef struct _DMA_CONTAINER{
        UINT8 channel;                                  /// внутренний номер канала
        UINT32 count;                                   /// количество блоков
        DMA_BLOCK blocks[DMA_COUNT_BLOCKS];             /// массив блоков
} DMA_CONTAINER, *PDMA_CONTAINER;

/// \brief ioctl запрос блоков DMA
Typedef struct _DMA_REQ{
        UINT8 channel;                                  /// внутренний номер канала
        UINT32 count;                                   /// желаемое количество блоков
} DMA_REQ, *PDMA_REQ;
```

Рисунок 4.1.4 – Листинг команды

4.1.5 Чтение из ДМА - IOCTL_READ_DMA_708

Позволяет прочитать данные из ДМА канала ARINC-708.

Описание параметров команды приведено на рисунке 4.1.5.

```
/** чтение DMA */
#define IOCTL_READ_DMA_708                                  CTL_CODE(FILE_DEVICE_CONTROLLER,
0x202, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief ioctl ответ с блоками DMA
Typedef struct _DMA_CONTAINER_708{
       UINT32 count;                                        /// количество блоков
       DMA_BLOCK_708 blocks[DMA_COUNT_BLOCKS_708];          /// массив блоков
} DMA_CONTAINER_708, *PDMA_CONTAINER_708;

/// \brief ioctl запрос блоков DMA
Typedef struct _DMA_REQ{
       UINT8 channel;                                             /// внутренний номер канала
       UINT32 count;                                        /// желаемое количество блоков
} DMA_REQ, *PDMA_REQ;
```

Рисунок 4.1.5 – Листинг команды

4.1.6 Сброс ДМА - IOCTL_CLEAN_DMA

Позволяет обнулить DMA_INDEX и программные указатели чтения/записи.

Описание параметров команды приведено на рисунке4.1.6.

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

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.

```
/** получение версии устройства и драйвера */
#define IOCTL_VERSION                                   CTL_CODE(FILE_DEVICE_CONTROLLER,
0x300, METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief версия драйвера и устройства
Typedef struct _VERSION_REQ{
      UINT32 device_id;                 /// id устройства
      UINT32 vendor_id;                 /// id производителя
      UINT32 revision_id;               /// ревизия устройства

      UINT64 mbar_addr;                 /// адрес BAR (память)
      UINT64 pbar_addr;                 /// адрес BAR (порт)
      UINT32 irq;                       /// вектор прерываний
      UINT64 dma_size;                  /// размер буфера DMA
      LARGE_INTEGER dma_addr_device;    /// адрес DMA для устройства
      UINT64 dma_addr_driver;           /// адрес DMA для драйвера

      UINT32 driver_version;            /// версия драйвера
      UINT32 driver_build_date;         /// дата сборки драйвера

      UINT8 rx_channel_count;           /// количество каналов-приёмников
      UINT8 tx_channel_count;           /// количество каналов-передатчиков
      UINT8 a708_channel_count;         // количество каналов 708
      UINT8 sc_tx_count;                /// количество приёмников рк
      UINT8 sc_rx_count;                /// количество передатчиков рк
      UINT32 g_channel_offset;          /// смещение номера первого канала устройства
} VERSION_REQ, *PVERSION_REQ;
```

Рисунок 4.1.7 – Листинг команды

4.1.8 Запись блока в RAM - IOCTL_WRITE_RAM

Позволяет записать блок данных в RAM передатчика ARINC-708.

Описание параметров команды приведено на рисунке 4.1.10.

```
/** запись памяти канала */
#define IOCTL_WRITE_RAM                                 CTL_CODE(FILE_DEVICE_CONTROLLER,
0x104, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief размер блока
#define RAM_REQ_MAX 256

#define RAM_BLOCK_TYPE_DATA             1
#define RAM_BLOCK_TYPE_DESCRIPTOR       2

/// \brief чтение/запись блока RAM передатчика
Typedef struct _RAM_REQ{
      UINT8 channel;                    /// внутренний номер канала
      UINT32 type;                      /// тип инструкций (DATA - 1, DESCRIPTOR - 2)
      UINT32offset;                     /// смещение относительно начала буфера
      UINT32length;                     /// количество блоков данных
      UINT32dwords[RAM_REQ_MAX];        /// блоки данных
} RAM_REQ, *PRAM_REQ;
```

Рисунок 4.1.10 – Листинг команды

4.1.9 Чтение блока из RAM - IOCTL_READ_RAM

Позволяет прочитать блок данных из RAM передатчика ARINC-708.

Описание параметров команды приведено на рисунке 4.1.11.

```
/** чтение памяти канала */
#define IOCTL_READ_RAM                                  CTL_CODE(FILE_DEVICE_CONTROLLER,
0x103, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief размер блока
#define RAM_REQ_MAX 256

#define RAM_BLOCK_TYPE_DATA             1
#define RAM_BLOCK_TYPE_DESCRIPTOR       2

/// \brief чтение/запись блока RAM передатчика
Typedef struct _RAM_REQ{
      UINT8 channel;                    /// внутренний номер канала
      UINT32 type;                      /// тип инструкций (DATA - 1, DESCRIPTOR - 2)
      UINT32offset;                     /// смещение относительно начала буфера
      UINT32length;                     /// количество блоков данных
      UINT32dwords[RAM_REQ_MAX];        /// блоки данных
} RAM_REQ, *PRAM_REQ;
```

Рисунок 4.1.11 – Листинг команды

4.1.10 Вкл/выкл слежения за прерываниями - IOCTL_TRACKING_INT

Позволяет включить – выключить функцию слежения за прерываниями.

Описание параметров команды приведено на рисунке 4.1.14.

```
#define IOCTL_TRACKING_INT              CTL_CODE(FILE_DEVICE_CONTROLLER, 0x304,
METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief управление прерываниями
Typedef struct _T_INT_REQ{
      UINT8 channel;                                  /// внутренний номер канала
      UINT32 manage;                                  /// вкл/выкл слежение
      UINT64startFreeTimer;                           /// начальное значение freetimer
      UINT64finishFreeTimer;                          /// конечное значение freetimer
      UINT32rkInAppearCount[T_CHECK_TRACKING_INT_CNT];     /// количество возникновений РК
} T_INT_REQ, *PT_INT_REQ;
```

Рисунок 4.1.14 – Листинг команды

4.1.11 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.

```
#define IOCTL_CHECK_TRACKING_INT        CTL_CODE(FILE_DEVICE_CONTROLLER, 0x305,
METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief управление прерываниями
Typedef struct _T_INT_REQ{
       UINT8 channel;                                  /// внутренний номер канала
       UINT32 manage;                                  /// вкл/выкл слежение
       UINT64startFreeTimer;                           /// начальное значение freetimer
       UINT64finishFreeTimer;                          /// конечное значение freetimer
       UINT32rkInAppearCount[T_CHECK_TRACKING_INT_CNT];     /// количество возникновений РК
} T_INT_REQ, *PT_INT_REQ;
```

Рисунок 4.1.15 – Листинг команды

4.1.12 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.

```
#define IOCTL_START_AS_RK_IN                    CTL_CODE(FILE_DEVICE_CONTROLLER, 0x306,
METHOD_BUFFERED, FILE_ANY_ACCESS)
/// \brief управление прерываниями
Typedef struct _T_INT_REQ{
       UINT8 channel;                                  /// внутренний номер канала
       UINT32 manage;                                  /// вкл/выкл слежение
       UINT64startFreeTimer;                           /// начальное значение freetimer
       UINT64finishFreeTimer;                          /// конечное значение freetimer
       UINT32rkInAppearCount[T_CHECK_TRACKING_INT_CNT];     /// количество возникновений РК
} T_INT_REQ, *PT_INT_REQ;
```

Рисунок 4.1.16– Листинг команды

4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ A708

Библиотека предназначена для функционирования в ОС семейства Windows и встраивания в прикладные приложения. В файле "libarinc708.h" представлены сервисные функции библиотеки взаимодействия.

Все функции библиотеки кроме a708_enumerate и a708_free возвращают код ошибки или ERROR_SUCCESS в случае успеха.

4.2.1 Получение списка устройств –a708_enumerate

Функция запрашивает у системы список устройств, реализующих интерфейс A708, опрашивает каждое устройство и формирует списки устройств и каналов.

```
/// \brief channel info
typedef struct sA708_CHANNEL_INFO {
	WCHAR* device_name;		/// имя файла устройства
	UINT32 g_channel;		/// глобальный номер канала
	UINT32 i_channel;		/// номер канала данного типа
	UINT32 b_channel;		/// номер канала на плате
	UINT32 g_board;			/// номер платы
	UINT32 channel_type;		/// тип канала (T_CHANNEL_RX или T_CHANNEL_TX)
	HANDLE arinc;			/// хэндл устройства
} A708_CHANNEL_INFO, A708_DEVICE_INFO, *PA708_CHANNEL_INFO;

typedef struct A708_DEVICE_LIST {
	UINT32 count;			/// количество каналов
	UINT32 pcount;			/// количество устройств

	A708_CHANNEL_INFO* device; /// список каналов
	A708_CHANNEL_INFO* pdev;   /// список устройств
} A708_DEVICE_LIST, * PA708_DEVICE_LIST;/**
* @brief получение списка устройств
*
* Функция получит у системы все устройства реализующие интерфейс с ID
GUID_DEVINTERFACE_ARINC708,
* откроет каждое, получит информацию о нём и сформирует два списка - устройств и
каналов. Структуру
* A708_DEVICE_LIST необходимо будет освободить с помощью a708_free.
*
*
* @return указатель на структуру A708_DEVICE_LIST
*/
A708_DEVICE_LIST* a708_enumerate();
...
A708_DEVICE_LIST* devices;
devices = a708_enumerate(); //получить список устройств
```

Рисунок 4.2.1 - Параметры команды a708_enumerate

4.2.2 Освобождение памяти – a708_free

Корректное освобождение памяти, занятой списком устройств и каналов.

```
VOID a708_free(A708_DEVICE_LIST* devices);

...

A708_free(devices); //освободить список устройств
```

Рисунок 4.2.2 - Параметры команды a708_free

4.2.3 Открытие символьного устройства a708_open

Открытие символьного устройства.

```
/**
* @brief открытие символьного устройства
*
* Функция попытается открыть описанное структурой устройство и запишет хэндл обратно в структуру A708_CHANNEL_INFO.
* После завершения использования конкретного канала или устройства хэндл необходимо закрыть с помощью CloseHandle.
*
* @param указатель на структуру A708_CHANNEL_INFO
*
* @return хэндл устройства или NULL, если произошла ошибка
*/
HANDLE a708_open(A708_CHANNEL_INFO* info);
...
A708_open(devices->device + 0);
CloseHandle(devices->device[0].arinc);
```

Рисунок 4.2.3 - Параметры команды a708_open

4.2.4 Чтение регистра из памяти a708_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 a708_read_reg(A708_CHANNEL_INFO* arinc, UINT32addr, UINT32* pbuf);
...
UINT32 buf;
a708_read_reg(devices->device + 0, 0x1010, &buf); //прочитать регистр 0x1010 (INTERRUPT_MASK) в переменную buf
```

Рисунок 4.2.4 - Параметры команды a708_read_reg

4.2.5 Запись регистра – a708_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 a708_write_reg(A708_CHANNEL_INFO* arinc, UINT32addr, UINT32buf);
...
a708_write_reg(devices->device + 0, 0x1010, 0x0); //записать 0x0 врегистр 0x1010
(INTERRUPT_MASK)
```

Рисунок 4.2.5 - Параметры команды a708_write_reg

4.2.6 Модификация бит регистра –a708_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 a708_modify_reg(A708_CHANNEL_INFO* arinc, UINT32addr, UINT32buf, UINT32mask);
...
a708_modify_reg(devices->device + 0, 0x1010, 1 << 8 , 1 << 8); //установить 8 бит
регистра INTERRUPT_MASK в 1
```

Рисунок 4.2.6 - Параметры команды a708_modify_reg

4.2.7 Чтение блоков DMA –a708_read_dma

Функция запросит не более pcount блоков из буфера dma. В случае успеха в переменную dmacount 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 a708_read_dma(A708_CHANNEL_INFO* arinc, UINT32* pcount, DMA_BLOCK** dmabuf);
...
DMA_BLOCK *dma = new DMA_BLOCK[DMA_COUNT_BLOCKKS];
UINT32 dmacount = 256;
a708_read_dma(devices->device + i, &dmacount, &dma);
for(inti=0;i<dmacount;i++){ ...} //обработкаблоков
delete[] dma;
```

Рисунок 4.2.7 - Параметры команды a708_read_dma

4.2.8 Сброс указателей DMA –a708_clear_dma

Сброс указателей 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 a708_clear_dma(A708_CHANNEL_INFO* arinc);
...
a708_clear_dma(devices->pdev + i);
```

Рисунок 4.2.8 - Параметры команды a708_read_dma

4.2.9 Получение информации об устройстве –a708_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 a708_get_device_info(A708_CHANNEL_INFO* arinc, VERSION_REQ* version);
...
VERSION_REQ ver;
a708_get_device_info(devices->pdev + i, &ver);
```

Рисунок 4.2.9 - Параметры команды a708_get_device_info

4.2.10 УправлениеDMA устройства – a708_manage_dma

Включение или выключение DMA всего модуля.

```
/**
* @brief управление DMA устройства
*
* @paramarinc указатель на структуру устройства
* @param value A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_manageDma(A708_CHANNEL_INFO* arinc, UINT32value);
...
a708_manageDma(devices->pdev + 0, 1); //включить DMA
```

Рисунок 4.2.10 - Параметры команды a708_manage_dma

4.2.11 Управление DMA канала – a708_manageChannelDma

Управление DMA конкретного канала.

```
/**
* @brief управление DMA канала
*
* @paramarinc указатель на структуру канала
* @param value A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_manageChannelDma(A708_CHANNEL_INFO* arinc, UINT32value);
...
a708_manageDma(devices->device + 5, 1); //включить DMA на 5 канале
```

Рисунок 4.2.11 - Параметры команды a708_manageChannelDma

4.2.12Записать в RAMпередатчика – a708_writeRam

Записьблоков RAM передатчика.

```
Typedef struct _RAM_REQ{
	UINT8 channel;			/// внутренний номер канала
	UINT32 type;			/// тип инструкций (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 a708_writeRam(A708_CHANNEL_INFO* arinc, RAM_REQ* container);
...
RAM_REQ *desc = newRAM_REQ;
loadMem(desc, "mem/desk_test_consol.mem");
a708_writeRam(devices->device + 0, desc);
```

Рисунок 4.2.12 - Параметры команды a708_writeRam

4.2.13Прочитать из RAM передатчика – a708_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 a708_readRam(A708_CHANNEL_INFO* arinc, RAM_REQ* container);
...
RAM_REQ *desc = newRAM_REQ;
a708_writeRam(devices->device + 0, desc);
```

Рисунок 4.2.13 - Параметры команды a708_readRam

4.2.14 Слежение за входными РК– a708_manageTrackingInterrupt

Управление слежением за входными РК.

```
/// \brief управление слежением за прерываниями
Typedef struct _T_INT_REQ{
	UINT8channel;			/// внутренний номер канала
	UINT32manage;			/// вкл/выкл слежение
	UINT64 startFreeTimer;		/// начальное значение freetimer
	UINT64 finishFreeTimer;		/// конечное значение freetimer
	UINT32 rkInAppearCount[T_CHECK_TRACKING_INT_CNT];	/// количество
возникновений РК
} 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 a708_manageTrackingInterrupt(A708_CHANNEL_INFO* arinc, T_INT_REQ* info);
...
T_INT_REQtreq;
treq.manage = T_TRACKING_INT_MAN_ON;
a708_manageTrackingInterrupt(devices->pdev + i, &treq);
```

Рисунок 4.2.14 - Параметры команды a708_manageTrackingInterrupt

4.2.15 Проверка прерывания РК – a708_checkTrackingInterrupt

Проверка возникновения прерывания по входным РК.

```
/**
* @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 a708_checkTrackingInterrupt(A708_CHANNEL_INFO* arinc, T_INT_REQ* info);
...
T_INT_REQtreq;
a708_checkTrackingInterrupt(devices->pdev + i, &treq);
```

Рисунок 4.2.15 - Параметры команды a708_checkTrackingInterrupt

4.2.16 Старт/стоп передатчика –a708_txStartStop

Включение – выключение передатчика.

```
/**
* @brief включение или выключение передачи
*
* @paramarinc указатель на структуру канала
* @param action A708_STOP или A708_START
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_txStartStop(A708_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.16 - Параметры команды a708_txStartStop

4.2.17 Однократный запуск передатчика – a708_txStartOnce

Однократный запуск передатчика.

```
/**
* @brief однократный запуск передатчика
* @paramarinc указатель на структуру канала
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_txStartOnce(A708_CHANNEL_INFO* arinc);
```

Рисунок 4.2.17 - Параметры команды a708_txStartOnce

4.2.18 Установка режима работы передатчика –a708_txSetMode

Установка режима работы передатчика.

```
#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
/**
* @brief установка режима работы передатчика
*
* @paramarinc указатель на структуру канала
* @parammode режим работы
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_txSetMode(A708_CHANNEL_INFO* arinc, UINT32mode);
```

Рисунок 4.2.18 - Параметры команды a708_txSetMode

4.2.19 Запись в FIFO передатчика –a708_writeTxFifo

Запись данных в буфер 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 a708_writeTxFifo(A708_CHANNEL_INFO* arinc, UINT32data);
```

Рисунок 4.2.19 - Параметры команды a708_writeTxFifo

4.2.20 Управление метками фильтрации –a708_setLabelConfReg

Управление метками фильтрации.

```
/**
* @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 a708_setLabelConfReg(A708_CHANNEL_INFO* arinc, UINT32numReg, UINT32mask,
UINT32value);
```

Рисунок 4.2.20 - Параметры команды a708_setLabelConfReg

4.2.21 Установка скорости работы канала –a708_setSpeed

Установка скорости работы канала.

```
#define A708_SPEED_100          0       ///< 100 кбит/с
#define A708_SPEED_12_14        1       ///< 12.5 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
/**
* @brief установка скорости работы канала
*
* @paramarinc указатель на структуру канала
* @param speed скорость (A708_SPEED_...)
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_setSpeed(A708_CHANNEL_INFO* arinc, UINT32speed);
```

Рисунок 4.2.21 - Параметры команды a708_setSpeed

4.2.22 Разрешение работы канала –a708_manageEnBit

Разрешение работы канала.

```
#define A708_STOP           0               ///<стоп
#define A708_START          ~A708_STOP///<старт
/**
* @brief разрешение работы канала
*
* @paramarinc указатель на структуру канала
* @param action A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_manageEnBit(A708_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.22 - Параметры команды a708_manageEnBit

4.2.23 Проверка разрешения работы канала –a708_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 - A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_isEnBit(A708_CHANNEL_INFO* arinc, UINT32* state);
```

Рисунок 4.2.23 - Параметры команды a708_isEnBit

4.2.24 Управление битами чётности –a708_parityManage

Управление битами чётности.

```
/**
* @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 a708_parityManage(A708_CHANNEL_INFO* arinc, UINT32parcheck, UINT32parity);
```

Рисунок 4.2.24 - Параметры команды a708_parityManage

4.2.25 Управление работой фильтра приёмника –a708_rxFiltration

Управление работой фильтра приёмника.

```
#define A708_OFF            0               ///<выкл
#define A708_ON             ~A708_OFF///<вкл
/**
* @brief управление работой фильтра приёмника
*
* @paramarinc указатель на структуру канала
* @param action A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_rxFiltration(A708_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.25 - Параметры команды a708_rxFiltration

4.2.26 Управление фильтром приёмника –a708_rxManageFiltrationLabel

Управление метками фильтра приёмника.

```
#define A708_OFF            0               ///<выкл
#define A708_ON             ~A708_OFF///<вкл
/**
* @brief управление метками фильтра приёмника
*
* @paramarinc указатель на структуру канала
* @param label метка
* @param action A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_rxManageFiltrationLabel(A708_CHANNEL_INFO* arinc, UINT32label,
UINT32action);
```

Рисунок 4.2.26 - Параметры команды a708_rxManageFiltrationLabel

4.2.27 Сброс меток фильтров приёмника –a708_rxClearFiltrationLabel

Сброс меток фильтров приёмника.

```
/**
* @brief сброс меток фильтра приёмника
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_rxClearFiltrationLabel(A708_CHANNEL_INFO* arinc);
```

Рисунок 4.2.27 - Параметры команды a708_rxClearFiltrationLabel

4.2.28 Управление битами SDI–a708_rxManageRcvSdi

Управление битами 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 a708_rxManageRcvSdi(A708_CHANNEL_INFO* arinc, UINT32action, UINT32bit9,
UINT32bit10);
```

Рисунок 4.2.28 - Параметры команды a708_rxManageRcvSdi

4.2.29 Деинициализация канала –a708_deinit

Деинициализация канала.

```
/**
* @brief деинициализация канала
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_deinit(A708_CHANNEL_INFO* arinc);
```

Рисунок 4.2.29 - Параметры команды a708_deinit

4.2.30 Управление порядком хода бит метки – a708_manageReverce

Управление порядком бит для слова данных (бит 28 регистров REG_RX_CONF, REG_TX_CONF).

```
/**
* @brief управление порядком бит для слова данных
*
* @paramarinc указатель на структуру канала
* @param action A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_manageReverce(A708_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.30 - Параметры команды a708_manageReverce

4.2.31 Установка времени паузы – a708_txGapBits

Управление временем паузы, биты 19-25 регистра REG_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 a708_txGapBits(A708_CHANNEL_INFO* arinc, UINT32gapBit);
```

Рисунок 4.2.2 - Параметры команды a708_txGapBits

4.2.32 Управление дискретностью таймера RRT – a708_txRrD

Управление дискретностью таймера RRT.

```
#defineA708_TXRR_10MS       0       ///<txrr 10 ms
#defineA708_TXRR_1MS        1       ///<txrr 1 ms
/**
* @brief управление дискретностью таймера RRT
*
* @paramarinc указатель на структуру канала
* @paramtxrr A708_TXRR_10MS или A708_TXRR_1MS
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_txRrD(A708_CHANNEL_INFO* arinc, UINT32txrr);
```

Рисунок 4.2.32 - Параметры команды a708_txRrD

4.2.33 Управление таймером RRT – a708_txSkipRrt

Управление таймером RRT.

```
/**
* @brief управление таймером RRT
*
* @paramarinc указатель на структуру канала
* @param action A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_txSkipRrt(A708_CHANNEL_INFO* arinc, UINT32action);
```

Рисунок 4.2.33 - Параметры команды a708_txSkipPart

4.2.34 Управление выходными РК – a708_manageScOut

Управление выходными РК.

```
#defineA708_SC_OUT_1            (1)         ///<ВЫХРКв 0
#defineA708_SC_OUT_0            (1<<1)      ///<ВЫХРКв 1
#defineA708_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 a708_manageScOut(A708_CHANNEL_INFO* arinc, UINT32action1, UINT32action2,
UINT32action3, UINT32action4);
```

Рисунок 4.2.34 - Параметры команды a708_manageScOut

4.2.35 Сброс разрешения прерывания входных РК – a708_clearScIntMask

Сброс разрешения прерывания выходных РК.

```
/**
* @brief сброс разрешений выходных РК
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_clearScIntMask(A708_CHANNEL_INFO* arinc);
```

Рисунок 4.2.35 - Параметры команды a708_clearScIntMask

4.2.36 Установка разрешения прерывания a708_manageScIntMask

Установка разрешений прерывания выходных РК.

```
/**
* @brief установка разрешений выходных РК
*
* @paramarinc указатель на структуру канала
* @paramnumRk номерканалаРК
* @paramstateRk A708_SC_RISE или A708_SC_FALL
* @paramactionRk A708_OFF или A708_ON
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_manageScIntMask(A708_CHANNEL_INFO* arinc, UINT32numRk, UINT32stateRk,
UINT32actionRk);
```

Рисунок 4.2.36 - Параметры команды a708_clearScIntMask

4.2.37 Запуск передатчика с RRT – a708_txStartWithRrt

Однократный запуск передатчика с таймером 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 a708_txStartWithRrtOnce(A708_CHANNEL_INFO* arinc, UINT32rrtValue);
```

Рисунок 4.2.37 - Параметры команды a708_txStartWithRrt

4.2.38 Управление передачей по запросу для передатчика – a708_txSetRequestTransferByRkIn

Управление передачей по запросу для передатчика.

```
/**
* @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 a708_txSetRequestTransferByRkIn(A708_CHANNEL_INFO* arinc, UINT32rknum);
```

Рисунок 4.2.38 - Параметры команды a708_txSetRequestTransferByRkIn

4.2.39 Управление маской прерываний – a708_manageInterruptMask

Управление маской прерываний.

```
/**
* @brief Управление маской прерываний
*
* @paramarinc указатель на структуру канала
* @parammask маска
* @paramvalue значение
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_manageInterruptMask(A708_CHANNEL_INFO* arinc, UINT32mask, UINT32value);
```

Рисунок 4.2.39 - Параметры команды a708_manageInterruptMask

4.2.40 Управление приёмом по готовности приёмника – a708_rxSetReceiveReadyByRkIn

Управление приёмом по готовности приёмника.

```
/**
* @brief Управление приёмом по готовности приёмника
*
* @paramarinc указатель на структуру канала
* @paramrknum номер канала РК
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_rxSetReceiveReadyByRkIn(A708_CHANNEL_INFO* arinc, UINT32rknum);
```

Рисунок 4.2.40 - Параметры команды a708_rxSetReceiveReadyByRkIn

4.2.41 Сброс RAM передатчика – a**708**_ramCls

Сброс буфера передатчика.

```
/**
* @brief сброс RAM передатчика
*
* @paramarinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_ramCls(A708_CHANNEL_INFO* arinc);
```

Рисунок 4.2.41 - Параметры командыа708_ramCls

4.2.42 Управление режимом работы приёмника канала Arinc708 – a708_rx708_manage

```
/**
* @brief Управление режимом работы приёмника канала Arinc708
*
* @param arinc указатель на структуру канала
* @param mode A708_ON или A708_OFF
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_rx708_manage(A708_CHANNEL_INFO* arinc, UINT32 mode);
```

Рисунок 4.2.42 - Параметры команды a708_rx708_manage

4.2.43 Управление режимом работы передатчика канала Arinc708 – a708_tx708_setMode

```
/**
* @brief Управление режимом работы передатчика канала Arinc708
*
* @param arinc указатель на структуру канала
* @param mode A708_ON или A708_OFF
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_setMode(A708_CHANNEL_INFO* arinc, UINT32 mode);
```

Рисунок 4.2.43 - Параметры команды a708_tx708_setMode

4.2.44 Управление режимом работы ShortDMA приёмника канала Arinc708 – a708_rx708_shortDma

```
/**
* @brief Управление режимом работы ShortDMA приёмника канала Arinc708
*
* @param arinc указатель на структуру канала
* @param mode A708_ON или A708_OFF
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_rx708_shortDma(A708_CHANNEL_INFO* arinc, UINT32 mode);
```

Рисунок 4.2.44 - Параметры команды a708_rx708_shortDma

4.2.45 Чтение буфера DMA канала Arinc708 – a708_readDmaA708

```
/**
* @brief Чтение буфера DMA канала Arinc708
*
* @param arinc указатель на структуру канала
* @param pcount указатель на максимальное количество блоков (будет записано количество
считанных блоков)
* @param dmabuf указатель на адрес памяти, где расположена структура буфера DMA
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_readDmaA708(A708_CHANNEL_INFO* arinc, UINT32* pcount, DMA_BLOCK_708**
dmabuf);
```

Рисунок 4.2.45 - Параметры команды a708_readDmaA708

4.2.46 Запись блока данных передатчика – Arinc708 a708_tx708_writeDataBlock

```
/**
* @brief Запись блока данных передатчика Arinc708
*
* @param arinc указатель на структуру канала
* @param numBlock номер буфера для записи (начинается с 1)
* @param block указатель на структуру блока с данными
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_writeDataBlock(A708_CHANNEL_INFO* arinc, UINT32 numBlock,
T_A708_DATA_BLOCK* block);
```

Рисунок 4.2.46 - Параметры команды a708_tx708_writeDataBlock

4.2.47 Чтение блока данных передатчика – Arinc708 a708_tx708_readDataBlock

```
/**
* @brief Чтение блока данных передатчика Arinc708
*
* @param arinc указатель на структуру канала
* @param numBlock номер буфера для чтения (начинается с 1)
* @param block указатель на структуру блока куда будут записаны данные
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_readDataBlock(A708_CHANNEL_INFO* arinc, UINT32 numBlock,
T_A708_DATA_BLOCK* block);
```

Рисунок 4.2.47 - Параметры команды a708_tx708_readDataBlock

4.2.48 Запуск передачи – Arinc708 a708_tx708_start

```
/**
* @brief Запуск передачи Arinc708
*
* @param arinc указатель на структуру канала
* @param mode режим передачи, A708_TX708_MODE_SINGLE_SHOT или A708_TX708_MODE_AUTO
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_start(A708_CHANNEL_INFO* arinc, UINT32 mode);
```

Рисунок 4.2.48 - Параметры команды a708_tx708_start

4.2.49 Проверка состояния буфера передачи Arinc708 – a708_tx708_hasSendedDataBlock

```
/**
* @brief Запуск передачи Arinc708
*
* @param arinc указатель на структуру канала
* @param mode режим передачи, A708_TX708_MODE_SINGLE_SHOT или A708_TX708_MODE_AUTO
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_start(A708_CHANNEL_INFO* arinc, UINT32 mode);
```

Рисунок 4.2.49 - Параметры команды a708_tx708_hasSendedDataBlock

4.2.50 Остановка передачи – Arinc708 a708_tx708_stop

```
/**
* @brief Остановка передачи Arinc708
*
* @param arinc указатель на структуру канала
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_stop(A708_CHANNEL_INFO* arinc);
```

Рисунок 4.2.50 - Параметры команды a708_tx708_stop

4.2.51 Установка таймера – Arinc708 a708_tx708_writeTimer

```
/**
* @brief Установка таймера Arinc708
*
* @param arinc указатель на структуру канала
* @param value значение таймера
*
* @return код ошибки или 0 в случае успеха
*/
UINT32 a708_tx708_writeTimer(A708_CHANNEL_INFO* arinc, UINT32 value);
```

Рисунок 4.2.50 - Параметры команды a708_tx708_writeTimer

Лист регистрации

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