**ЦИФРОВОЙ ПУЛЬТ СВЯЗИ (ЦПС)**

**версия 3.1.0.1**

**Инструкция разработчика ПО**

2024 г.

Содержание

[1 Введение 3](#_Toc168059858)

[2 Зависимости и сборка 4](#_Toc168059859)

[3 Используемое оборудование 5](#_Toc168059860)

[3.1 Разговорные приборы 5](#_Toc168059861)

[3.2 Устройства управления 6](#_Toc168059862)

[4 Архитектура программного интерфейса 6](#_Toc168059863)

[5 Специальные определения типов 8](#_Toc168059864)

[6 Передаваемые и принимаемые структуры 11](#_Toc168059865)

[7 Описание методов API 20](#_Toc168059866)

[7.1 Подключение библиотеки 20](#_Toc168059867)

[7.2 Создание управляющей сессии 20](#_Toc168059868)

[7.3 Использование управляющей сессии 21](#_Toc168059869)

[7.4 Разрушение управляющей сессии 22](#_Toc168059870)

[7.5 Асинхронные события управляющей сессии 23](#_Toc168059871)

# 1 Введение

Данный документ предназначен для разработчиков стороннего программного обеспечения, которое должно выполняться на терминальном оборудовании производства TSIntercom [(tsintercom.ru)](https://tsintercom.ru/). К такому оборудованию относятся:

* Сенсорный IP-терминал связи (TSDT-12), ([12” экран](https://tsintercom.ru/wp-content/uploads/2024/04/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5-%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5-TSDT-12-_-v1.pdf) - техническое описание).
* Встраиваемый сенсорный IP-терминал связи DCP-2, ([8” экран](https://tsintercom.ru/wp-content/uploads/2024/04/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5-%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5-TSDT-8-_-v1.pdf) - техническое описание).

Архитектура вычислительного модуля в обоих случаях – x86(x64). Для уточнения конкретных деталей следует обратиться к техническому описанию данных устройств.

Документ содержит информацию об устройствах, которые поддерживает библиотека, описывает архитектуру и алгоритм использования библиотеки.

Также в документе имеется описание типов, принимаемых/переда­ваемых структур и программного интерфейса (API), позволяющего разработчику прикладного программного обеспечения, оперативно получать информацию от периферийных устройств указанного выше оборудования и осуществлять управление.

Подразумевается использование данного API разработчиками под Windows или Linux. Базовая библиотека предоставляется в исходных кодах, с тем, чтобы имелась возможность её самостоятельной сборки.

# 2 Зависимости и сборка

Описываемое API реализуется кодом динамически загружаемой библиотеки **apiTechStartLib**.

Под **Windows** для работы описываемого API необходимо наличие установленных в системе драйверов для устройств разработки фирмы FTDI. Их необходимо установить с сайта производителя (добавить ссылку), если в комплекте с устройством нет диска с этим программным обеспечением.

Под **Linux** предоставляются исходные коды драйвера, правила и скрипт для udev-подсистемы. Если устройство приобретено с предустановленной ОС, то необходимые драйвера будут уже установлены и настроены. В противном случае необходимо воспользоваться сайтом производителя и скачать архив для установки драйвера.

Порядок установки драйвера:

1. Извлечь содержимое архива в удобную разработчику директорию.

2. Запустить скрипт ft245\_setup.sh.

**Примечание:** в случае обновления ядра ОС необходимо переустановить драйвер запуском скрипта ft245\_setup.sh и последующей перезагрузкой устройства, для корректной работы библиотеки.

Кроме этого, на сайте производителя можно скачать уже собранные библиотеки, описываемые этим документом.

# 3 Используемое оборудование

## 3.1 Разговорные приборы

Разговорные приборы подключаются к оборудованию через стандартные звуковые карты, которые реализованы либо на вычислительном модуле, либо на внешних аудиокодеках. В обоих случаях, для прикладного программного обеспечения необходимо реализовать обработку звука, получаемого со звуковых карт.

|  |  |  |
| --- | --- | --- |
| **Звуковое устройство** | **Разговорные приборы (РП)** | **Описание** |
| Realtek ALC662 Audio Codec | 1. Микрофон.
2. Пара встроен­­ных динамиков.
3. Внешняя активная аудиосистема.
 | Встроенное звуковое устройство в материнс­кую плату. |
| PCM 2912A Audio Codec | 1. 2 аудиоканала канала для подключения динамиков или аналогового магнито­фона (разъем RG45).
 | Встроенное USB-устройство. |
| PCM 2912A Audio Codec | 1. Подсоединяемые внешние трубка или гарнитура.
 | Внешнее USB-устройство. |

## 3.2 Устройства управления

В оборудовании имеется контроллер встроенных приборов управления, который позволяет подключать внешние устройства управления к внутренним или внешним USB-портам.

|  |  |  |
| --- | --- | --- |
| **Устройство управления** | **Приборы управления** | **Описание** |
| Контроллер «Мульти К» | 1. Набор кнопок управления.
2. Набор энкодеров управления.
3. Набор диагностических датчиков.
 | Набор встроенных кнопок управления может состоять из 1-й или 4-х. Набор энкодеров - из одного энкодера с 3-мя позициями регулировок. Энко­дер имеет 3 переключаемые позиции, 2 из которых доступны программно (регули­ровка уровня динамика и микрофона). Диагностические датчики позволяют получить информацию о: статусе питания и диагностические данные о состоянии сенсорной панели. |
| Контроллер «Панель кнопок» | 1. Набор кнопок управления.
 | Подключаемая панель. Набор может содержать разное коли­чество кнопок. Кнопки могут иметь различные характе­рис­тики. |
| Контроллер «Панель энкодеров» | 1. Набор энкодеров управления.
 | Подключаемая панель. Набор может содержать разное коли­чество энкодеров. Каждый эн­кодер может иметь различ­ные характеристики. |
| Контроллер «Панель РП» | 1. Датчик положения трубки.
2. Набор кнопок управления.
 | Подключаемая панель. Опцио­на­льно может содержать дат­чик положения РП. Набор и характеристики кнопок могут быть различные. |

# 4 Архитектура программного интерфейса

Библиотека использует определения, которые описаны в заголовочном файле ([apitechstartlib.h](#_Специальные_определения_типов)).

Библиотека **apiTechStartLib** экспортирует 4 метода:

1. Подключение библиотеки ([apiTechStartConnectLib](#_Подключение_библиотеки))
2. Создание управляющей сессии ([apiTechStartCtlCreate](#_Создание_управляющей_сессии)).
3. Использование управляющей сессии ([apiTechStartCtl](#_Использование_управляющей_сессии)).
4. Разрушение управляющей сессии ([apiTechStartCtlDestroy](#_Разрушение_управляющей_сессии)).

Библиотека использует асинхронные вызовы ([apiTechStartCtlEventRoutine](#_Асинхронные_события_управляющей_сессии)Proc) для оповещения пользователя о событиях в реальном масштабе времени и ([apiTechStartEventRoutineAudio](#_Асинхронное_событие_потока_аудиоданных)) для отправки потока аудиоданных пользователю для обработки.

Как экспортируемые методы, так и обратные вызовы используют **stdcall** соглашение на передачу параметров.

Типичный алгоритм использования API:

1. Подключение библиотеки, используя метод **apiTechStartConnectLib**.
2. Создание управляющей сессии, используя метод **apiTechStartCtlCreate**.
3. Использование управляющей сессии методом **apiTechStartCtl**, для получения текущей конфигурации оборудования и управления теми приборами, которые это позволяют.

Приложение, если ему это необходимо, должно быть готово принимать асинхронные события от библиотеки.

Виды событий:

1. Логирование событий библиотеки и управляющих сессий.
2. Изменение списка доступных устройств.
3. Изменение состояний датчиков приборов управления.
4. Текущая конфигурация устройств при подключении.
5. Изменение состояний диагностических датчиков.
6. Поток аудиоданных от разговорных приборов.

# 5 Специальные определения типов

**typedef unsigned long apiTechStartHandle**

Данный тип является числовым дескриптором сессии управления. Библиотека возвращает это значение при создании управляющей сессии. Этот дескриптор должен быть применен в методах использования и разрушения управляющей сессии.

**typedef unsigned long apiTechStartResult**

Данный тип описывает результат выполнения операции:

|  |  |  |
| --- | --- | --- |
| Название | Значение | Описание |
| **operationError** | 0 | Ошибка выполнения операции |
| **operationSuccess** | 1 | Операция выполнена успешно |
| **operationUnknown** | 2 | Неизвестная команда |

**typedef unsigned short apiTechStartCmd**

Данный тип описывает управляющую команду. Этот описатель должен быть применен в методе использования управляющей сессии.

|  |  |  |
| --- | --- | --- |
| Название | Команда | Описание |
| **commandIgnoreKA** | 0 | Вывод в лог состояния keep alive. Keep alive показывает, что связь между приложением и подключенным устройством стабильна. |
| **commandOpenDevice** | 1 | Открыть устройство |
| **commandSetColorBtn** | 2 | Установка цвета подсветки кнопок управления |
| **commandSetColorEnc** | 3 | Установка цвета шкалы энкодера |
| **commandSetLevelEnc** | 4 | Установка уровня энкодера |
| **commandPlayAudio** | 5 | Воспроизвести аудиофайл |
| **commandStopAudio** | 6 | Остановить воспроизведение аудиофайла |

**typedef unsigned short apiTechStartEvent**

Данный тип описывает асинхронные события. Этот описатель применяется в приемнике асинхронных событий от библиотеки.

|  |  |  |
| --- | --- | --- |
| **Название** | **Код события** | **Описание** |
| **eventLibLogging** | 0 | Логирование событий библиотеки |
| **eventSessionLog** | 1 | Логирование управляющей сессии |
| **eventChangeDevice** | 2 | Изменение списка доступных устройств |
| **eventBtnStateChanged** | 3 | Изменение состояния кнопки |
| **eventNewPTTBtnDetect** | 4 | Подключение кнопки устройства управления и её состояние |
| **eventNewDeviceButton** | 5 | Обнаружение набора с кнопками управления при подключении и его характеристики |
| **eventNewPTTBtnConfig** | 6 | Текущая конфигурация кнопки при подключении |
| **eventNewDeviceEncoder** | 7 | Обнаружение набора энкодеров управления при подключении и его характеристики |
| **eventNewEncoderConfig** | 8 | Текущая конфигурация энкодера при подключении |
| **eventEncoderStateChanged** | 9 | Изменение состояния энкодера |
| **eventEncoderPressed** | 10 | Нажатие кнопки энкодера |
| **eventPowerState** | 11 | Состояние питания |
| **eventDispState** | 12 | Состояние сенсорного дисплея |
| **eventHandsetState** | 13 | Положение трубки |
| **eventAudioData** | 14 | Поток аудиоданных |

**typedef void \*apiTechStartCmdParam**

Данный тип является непрозрачным указателем для передаваемых структур или параметров при отправке управляющей команды. Этот описатель должен быть применен в методе использования управляющей сессии.

**typedef void \*apiTechStartEventParam**

Данный тип является непрозрачным указателем для принимаемых структур или параметров при приеме асинхронных событий от библиотеки. Этот описатель может, если необходимо, быть применен в приемнике асинхронных событий от библиотеки.

**typedef void (\*apiTechStartCtlEventRoutine)(apiTechStartHandle hCtl, apiTechStartEvent event, apiTechStartEventParam \*param)**

Данный тип описывает указатель на функцию обратного вызова, который получает и использует библиотека для отправки асинхронных событий приложению в реальном масштабе времени.

**typedef apiTechStartResult (\*ApiConnect)**

 **(apiTechStartCtlEventRoutine)**

Данный тип описывает указатель на функцию подключения библиотеки, для использования в пользовательском приложении.

**typedef apiTechStartHandle (\*CtlCreate)**

**(apiTechStartCtlEventRoutine, apiTechStartCtlEventRoutine)**

Данный тип описывает указатель на функцию создания управляющей сессии, для использования в пользовательском приложении.

**typedef apiTechStartResult (\*Ctl)**

**(apiTechStartHandle, apiTechStartCmd, apiTechStartCmdParam)**

Данный тип описывает указатель на функцию использования управляющей сессии, для использования в пользовательском приложении.

**typedef apiTechStartResult (\*CtlDestroy)**

**(apiTechStartHandle)**

Данный тип описывает указатель на функцию разрушения управляющей сессии, для использования в пользовательском приложении.

# 6 Передаваемые и принимаемые структуры

struct **DevConfig**

{

 unsigned short **devType**;

 unsigned short **devDip**;

 unsigned short **version**;

 unsigned short **elemCount**;

};

Принимаемая структура по асинхронным событиям с кодами 5 и 7. Описывает характеристики подключаемого устройства управления.

|  |
| --- |
| struct **DevConfig** |
| Поле | Значения | Описание |
| unsigned short **devType** | 1 - 255 | Код типа устройства:3 - контроллер «Мульти К»;4 - контроллер «Панель кнопок»;5 - контроллер «Панель энкодеров»;6 - контроллер «Панель РП». |
| unsigned short **devDip** | 0 - 15 | Номер устройства (DIP-switch) |
| unsigned short **version** | 1 | Версия протокола |
| unsigned short **elemCount** | 1 - 255 | Количество элементов |

struct **SetColor**

{

 unsigned short **devIndex**;

 unsigned short **setRed**;

 unsigned short **setGreen**;

 unsigned short **setBlue**;

 unsigned short **setWhite**;

};

Передаваемая структура по командам с кодами 3 и 4. Описывает устанавливаемые значения поддерживаемых цветов RGB и белого.

|  |
| --- |
| struct **SetColor** |
| Поле | Значения | Описание |
| unsigned short **devIndex** | 0 - 255 | Номер кнопки или энкодераДиапазон для кнопок: 0 - 255Диапазон для энкодеров: 0 - 15Если 0 - все кнопки/энкодеры одним цветом |
| unsigned short **setRed** | 0 - 255 | Значение красного цвета |
| unsigned short **setGreen** | 0 - 255 | Значение зеленого цвета |
| unsigned short **setBlue** | 0 - 255 | Значение синего цвета |
| unsigned short **setWhite** | 0 - 255 | Значение белого цвета |

struct **AudioData**

{

 char **\*data**;

 unsigned short **length**;

};

Принимаемая структура по асинхронному событию с кодом 14. Описывает необработанный пакет аудиоданных с разговорных приборов. Извлечение и обработка данных осуществляется разработчиком.

Передаваемая структура по команде 5. Описывает пакет аудиоданных, который необходимо воспроизвести разработчику.

|  |
| --- |
| struct **AudioData** |
| Поле | Значения | Описание |
| char **\*data** | 0 - 255 | Указатель на пакет необработанных аудиоданных от устройства или на аудиоданные для воспроизведения устройством |
| unsigned long int **length** | 64 для приёма/любое для передачи | Размер пакета аудиоданных.Для приёма данных всегда является равным 64 байтам.Для передачи нет ограничений. |

struct **pwrChState**

{

 unsigned short **ch1**;

 unsigned short **ch2**;

 unsigned short **ch1UV**;

 unsigned short **ch1OV**;

 unsigned short **ch2UV**;

 unsigned short **ch2OV**;

};

Принимаемая структура по асинхронному событию с кодом 11. Описывает состояние каналов питания устройства.

|  |
| --- |
| struct **pwrChState** |
| Поле | Значения | Описание |
| unsigned short **ch1** | 0 - 1 | Состояние питания канала 10 - питание отключено1 - питание подключено |
| unsigned short **ch2** | 0 - 1 | Состояние питания канала 20 - питание отключено1 - питание подключено |
| unsigned short **ch1UV** | 0 - 1 | Ошибка UV канала 10 - норма1 - пониженное напряжение |
| unsigned short **ch1OV** | 0 - 1 | Ошибка OV канала 10 - норма1 - повышенное напряжение |
| unsigned short **ch2UV** | 0 - 1 | Ошибка UV канала 20 - норма1 - пониженное напряжение |
| unsigned short **ch2OV** | 0 - 1 | Ошибка OV канала 20 - норма1 - повышенное напряжение |

struct **pwrDispState**

{

 unsigned short **dispC**;

 unsigned short **dispOn**;

};

Принимаемая структура по асинхронному событию с кодом 12. Описывает состояние сенсорного дисплея.

|  |
| --- |
| struct **pwrDispState** |
| Поле | Значения | Описание |
| unsigned short **dispC** | 0 - 1 | Состояние подключения дисплея к устройству0 - не подключен1 - подключен |
| unsigned short **setOn** | 0 - 1 | Сенсор и подсветка включены0 - выключено1 - включено |

struct **HandsetState**

{

 unsigned short **state**;

};

Принимаемая структура по асинхронному событию с кодом 13. Описывает положение трубки.

|  |
| --- |
| struct **HandsetState** |
| Поле | Значения | Описание |
| unsigned short **state** | 0 - 1 | Положение трубки0 - трубка на базе1 - трубка снята с базы |

struct **PTTConfig**

{

 unsigned short **btnIndex**;

 unsigned short **btnExternal**;

 unsigned short **btnTumb**;

 unsigned short **suppRed**;

 unsigned short **suppGreen**;

 unsigned short **suppBlue**;

 unsigned short **suppWhite**;

};

Принимаемая структура по асинхронному событию с кодом 6. Описывает текущую конфигурацию кнопки при подключении.

|  |
| --- |
| struct **PTTConfig** |
| Поле | Значения | Описание |
| unsigned short **btnIndex** | 1 - 255 | Номер кнопки |
| unsigned short **btnExternal** | 0 - 1 | Встроенная или подключаемая кнопка0 - встроенная1 - подключаемая |
| unsigned short **btnTumb** | 0 - 1 | Имеет альтернативное положение нажатия0 - не имеет1 - имеет |
| unsigned short **suppRed** | 0 - 1 | Имеет красную составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **suppGreen** | 0 - 1 | Имеет зеленую составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **suppBlue** | 0 - 1 | Имеет синюю составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **suppWhite** | 0 - 1 | Имеет белую составляющую подсветки0 - не имеет1 - имеет |

struct **PTTCurrState**

{

 unsigned short **btnIndex**;

 unsigned short **btnPressed**;

 unsigned short **tumbPressed**;

};

Принимаемая структура по асинхронным событиям с кодами 3 и 4. Описывает текущее изменение состояния кнопки или состояние кнопки при подключении.

|  |
| --- |
| struct **PTTCurrState** |
| Поле | Значения | Описание |
| unsigned short **btnIndex** | 1 - 255 | Номер кнопки |
| unsigned short **btnPressed** | 0 - 1 | Нажатие кнопки0 - не нажата1 - нажата |
| unsigned short **tumbPressed** | 0 – 1 | Нажатие кнопки (альтернативное положение)0 - не нажата1 - нажата |

struct **EncConfig**

{

 unsigned short **encIndex**;

 unsigned short **modelRealize**;

 unsigned short **countParam**;

 unsigned short **suppRed**;

 unsigned short **suppGreen**;

 unsigned short **suppBlue**;

 unsigned short **suppWhite**;

 unsigned short **colorCh**;

 unsigned short **btnCh**;

};

Принимаемая структура по асинхронному событию с кодом 8. Описывает текущую конфигурацию энкодера при подключении.

|  |
| --- |
| struct **EncConfig** |
| Поле | Значения | Описание |
| unsigned short **encIndex** | 1 - 255 | Номер энкодера |
| unsigned short **modelRealize** | 0 - 15 | Модель реализации0 - энкодер с N позициями регулировок без программирования цвета шкалы1 - обычный энкодер с программированием цвета шкалы |
| unsigned short **countParam** | 0 - 15 | Количество встроенных параметров (позиций регулировок)0 - для обычных энкодеров1-15 - для энкодеров с N позициями регулировок |
| unsigned short **suppRed** | 0 - 1 | Имеет красную составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **suppGreen** | 0 - 1 | Имеет зеленую составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **suppBlue** | 0 - 1 | Имеет синюю составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **suppWhite** | 0 - 1 | Имеет белую составляющую подсветки0 - не имеет1 - имеет |
| unsigned short **colorCh** | 0 - 1 | Доступно изменение цвета шкалы0 - недоступно1 - доступно |
| unsigned short **btnCh** | 0 - 1 | Событие кнопки энкодера передается0 - не передается1 - передается |

struct **EncState**

{

 unsigned short **encIndex**;

 unsigned short **numParam**;

 unsigned short **minParam**;

 unsigned short **maxParam**;

 unsigned short **valueParam**;

};

Принимаемая структура по асинхронному событию с кодом 9. Передаваемая структура по команде с кодом 5. Описывает текущее изменение состояния энкодера, состояние энкодера при подключении, устанавливаемое состояние энкодера.

|  |
| --- |
| struct **EncState** |
| Поле | Значения | Описание |
| unsigned short **encIndex** | 0 - 15 | Номер энкодера0 - один энкодер с N позициями регулировок0-15 - номер обычного энкодера (0 — все энкодеры) |
| unsigned short **numParam** | 0 - 15 | Номер параметра (позиции регулировки)0 - нет параметров (позиций регулировки)1-15 - номер параметра (позиции регулировки) |
| unsigned short **minParam** | 0 - 23 | Минимальное значение шкалы |
| unsigned short **maxParam** | 0 - 23 | Максимальное значение шкалы |
| unsigned short **valueParam** | min <= v <= max | Значение шкалы |

struct **EncPressed**

{

 unsigned short **encIndex**;

};

Принимаемая структура по асинхронному событию с кодом 10. Описывает нажатие кнопки энкодера. Энкодеры с позициями регулировки не отображают нажатие.

|  |
| --- |
| struct **EncPressed** |
| Поле | Значения | Описание |
| unsigned short **encIndex** | 1 - 15 | Номер энкодера |

# 7 Описание методов API

## 7.1 Подключение библиотеки

**apiTechStartResult apiTechStartConnectLib(**

**apiTechStartCtlEventRoutine lpEventRoutine)**

Подключает библиотеку, давая доступ к функциональности. После подключения библиотека сканирует подключаемые/отключаемые устройства и отправляет список доступных устройств в лог и реализованную пользователем функцию обратного вызова событием **eventChangeDevice**.

|  |  |
| --- | --- |
| Параметр | Описание |
| apiTechStartCtlEventRoutine lpEventRoutine **lpEventRoutine** | Указатель на функцию обратного вызова пользовательского приложения.Для отправки асинхронных событий |

Результат:

Возвращает **operationSuccess** при успешном выполнении операции.

Возвращает **operationError** при ошибке.

## 7.2 Создание управляющей сессии

**ApiTechStartHandle apiTechStartCtlCreate(**

**apiTechStartCtlEventRoutine lpEventRoutine,**

**apiTechStartCtlEventRoutine lpEventRoutineAudio)**

Создает управляющую сессию. Без управляющей сессии невозможно подключиться к устройству. Нумерация сессий начинается с 1.

|  |  |
| --- | --- |
| Параметр | Описание |
| apiTechStartCtlEventRoutine lpEventRoutine **lpEventRoutine** | Указатель на функцию обратного вызова пользовательского приложе­ния. Для приема асинхронных событий |
| apiTechStartCtlEventRoutine lpEventRoutine **lpEventRoutineAudio** | Указатель на функцию обратного вызова пользовательского приложе­ния. Для приема потока аудиоданных с разговорных приборов |

Результат:

Возвращает числовой дескриптор сессии управления. Этот дескриптор применяется в методах использования и разрушения управляющей сессии.

## 7.3 Использование управляющей сессии

**apiTechStartResult apiTechStartCtl(**

**apiTechStartHandle hCtl,apiTechStartCmd aCmd,**

**apiTechStartCmdParam aCmdParam)**

Отправляет команды управляющей сессии, обращаясь через дескриптор. Команды 0 и 1 вместо числового дескриптора сессии передают 0.

|  |  |
| --- | --- |
| Параметр | Описание |
| **apiTechStartHandle hCtl** | Числовой дескриптор управляю­щей сессии, которой отправляется команда |
| **apiTechStartCmd aCmd** | Отправляемая команда управляю­щей сессии |
| **apiTechStartCmdParam aCmdParam** | Отправляемые данные управляю­щей сессии |

Результат:

Возвращает **operationSuccess** при успешном выполнении операции.

Возвращает **operationError** при ошибке.

Описание передаваемых данных при отправке команд:

|  |  |  |  |
| --- | --- | --- | --- |
| Команда | hCtl | Данные | Описание |
| **commandIgnoreKA** | 0 | bool | 0 - выводит в лог состояние keep alive1 - вывод в лог игнорируется |
| **commandOpenDevice** | ≥1 | string | Название устройства, которое необходимо подключить. Для получения списка доступных устройств необходимо реализовать обработку события **eventChangeDevice** |
| **commandSetColorBtn** | struct SetColor\* | Устанавливаемые значения поддер­жи­ваемых цветов RGB и белого устройств управления |
| **commandSetColorEnc** |
| **commandSetLevelEnc** | struct EncState\* | Устанавливаемое состояние энкодера |
| **commandPlayAudio** | struct AudioData\* | Воспроизводимые аудиоданные |
| **commandPlayAudio** | NULL | - |

## 7.4 Разрушение управляющей сессии

**ApiTechStartResult apiTechStartCtlDestroy(**

**apiTechStartHandle hCtl)**

Разрушает управляющую сессию с указанным дескриптором. При уничтожении сессии с подключенным устройством, связь с устройством прерывается.

|  |  |
| --- | --- |
| Параметр | Описание |
| **apiTechStartHandle hCtl** | Числовой дескриптор управляющей сессии, которую нужно разрушить |

Результат:

Возвращает **operationSuccess** при успешном выполнении операции.

Возвращает **operationError** при ошибке.

## 7.5 Асинхронные события управляющей сессии

**void apiTechStartCtlEventRoutineProc(**

**apiTechStartHandle hCtl,**

**apiTechStartEvent event,**

**apiTechStartEventParam param)**

Функция обратного вызова, которая обрабатывает отправляемые библиотекой асинхронные события в реальном времени. Реализуется в приложении разработчиком.

|  |  |
| --- | --- |
| Параметр | Описание |
| **apiTechStartHandle hCtl** | Числовой дескриптор управляющей сессии, от которой пришло событие,0 - событие от библиотеки |
| **apiTechStartEvent event** | Асинхронное событие |
| **apiTechStartEventParam param** | Данные получаемые от возникающих событий |

Описание получаемых данных от возникающих событий:

|  |  |  |  |
| --- | --- | --- | --- |
| Событие | hCtl | Данные | Описание |
| **eventLibLogging** | 0 | string\* | Сообщения от библиотеки |
| **eventSessionLog** | ≥1 | string\* | Лог управляющих сессий |
| **eventChangeDevice** | 0 | list<string>\* | Список доступных устройств |
| **eventBtnStateChanged** | ≥1 | struct PTTCurrState\* | Состояние кнопки |
| **eventNewPTTBtnDetect** |
| **eventNewDeviceButton** | struct DevConfig\* | Характеристики подклю­ченного набора с кнопками |
| **eventNewPTTBtnConfig** | struct PTTConfig\* | Конфигурация подключа­емой кнопки |
| **eventNewDeviceEncoder** | struct DevConfig\* | Характеристики подклю­чен­ного набора с энкодерами |
| **eventNewEncoderConfig** | struct EncConfig\* | Конфигурация подключаемого энкодера |
| **eventEncoderStateChanged** | struct EncState\* | Состояние энкодера |
| **eventEncoderPressed** | struct EncPressed\* | Нажатие кнопки энкодера |
| **eventPowerState** | struct pwrChState\* | Состояние каналов питания устройства |
| **eventDispState** | struct pwrDispState\* | Состояние сенсорного дисплея |
| **eventHandsetState** | struct HandsetState\* | Состояние трубки |

**void** **apiTechStartCtlEventRoutineAudio(**

**apiTechStartHandle hCtl,**

**apiTechStartEvent event,**

**apiTechStartEventParam param)**

Функция обратного вызова, которая обрабатывает отправляемые библиотекой пакеты аудиоданных с разговорных приборов в реальном времени. Реализуется в приложении разработчиком.

Описание получаемых данных от возникающих событий:

|  |  |  |  |
| --- | --- | --- | --- |
| Событие | hCtl | Данные | Описание |
| **eventAudioData** | ≥1 | struct AudioData\* | Пакет аудиоданных от разговорного прибора |