Универсальный интерфейс управления трансивером «TCI»
Версия 1.0

Введение

Данный документ ознакомит Вас о том, что такое интерфейс TCI, для чего он нужен и каким образом его использовать. Основной целевой аудиторией данного документа являются программисты, занимающиеся внедрением данного интерфейса в свои программы и устройства.

Интерфейс управления трансивером TCI (от англ. TCI – Transceiver Control Interface) представляет собой сетевой интерфейс управления, передачи данных и синхронизации между трансивером/приемником, аппаратным журналом, контест логгером, программами цифровых видов связи, скиммерами и другим программным обеспечением, а также внешними усилителями мощности, блоками диапазонных фильтров, антенными коммутаторами, контроллерами радиостанции и др. устройствами.

TCI был создан в качестве современной альтернативы устаревшим интерфейсам COM-портов и звуковых кабелей, он использует полнодуплексный web-сокет протокол, который работает поверх соединения TCP и служит для обмена данными между сервером и клиентом, обеспечивая кроссплатформенность. В качестве сервера выступает трансивер, в качестве клиентов — все остальные программы и устройства. Сервер и клиенты могут находиться внутри одного компьютера (SDR-программа-сервер, аппаратный журнал и др. - клиенты) и/или в раздельных физических устройствах, объединенных через локальную сеть (классический трансивер, усилитель мощности, антенный коммутатор, блок ДПФ и т.п.).

Интерфейс TCI содержит основные команды управления трансивером (аналог CAT системы), принимает от клиентов CW макросы и передает их в эфир, выдает квадратурный поток трансивера клиентам, принимает споты от скиммеров и интернет кластеров, принимает/выдает аудио сигнал для работы в цифровых видах связи*.
* Будет реализовано в новой версии интерфейса TCI.

TCI использует расширяемую архитектуру и может быть дополнен новыми функциями и командами, при сохранении работоспособности старых. Таким образом интерфейс TCI может быть расширен и дополнен под конкретные нужды любого производителя программного обеспечения и/или производителя устройств (приемников, трансиверов, усилителей мощности, коммутаторов и др.). Наличие идентификатора устройства позволяет производителям трансиверов и приемников перейти на TCI интерфейс с сохранением обозначения модели устройства. Расширяемость интерфейса TCI позволяет создавать индивидуальный набор команд и функций для каждой модели устройства, при этом сохраняя основной набор команд, присущий всем трансиверам.

Наша компания выступает за всеобщую унификацию обмена данными между устройствами и программами, создав для этого интерфейс TCI. Современные трансиверы и программное обеспечение должны общаться на одном языке — на языке TCI.

Лицензия MIT на использование программы-примера клиента TCI

Данная лицензия разрешает лицам, получившим копию данного программного обеспечения (в дальнейшем именуемыми «Программное Обеспечение»), безвозмездно использовать Программное Обеспечение без ограничений, включая неограниченное право на использование, копирование, изменение, слияние, публикацию, распространение, сублицензирование и/или продажу копий Программного Обеспечения, а также лицам, которым предоставляется данное Программное Обеспечение, при соблюдении следующих условий:

Указанное выше уведомление об авторском праве и данные условия должны быть включены во все копии или значимые части данного Программного Обеспечения.

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА УЩЕРБ ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, В ТОМ ЧИСЛЕ, ПРИ ДЕЙСТВИИ КОНТРАКТА, ДЕЛИКТЕ ИЛИ ИНОЙ СИТУАЦИИ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

Лицензия на использование интерфейса TCI

Данная лицензия разрешает лицам, использующим интерфейс TCI (далее Интерфейс) и сопутствующую документацию к нему, безвозмездно использовать его в любом Программном Обеспечении без каких-либо ограничений и условий, включая неограниченное право на использование, копирование, изменение, слияние, публикацию, распространение, сублицензирование и/или продажу копий Программного Обеспечения с Интерфейсом в его составе, а также лицам, которым предоставляется данное Программное Обеспечение.

ДАННЫЙ ИНТЕРФЕЙС ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА УЩЕРБ ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, В ТОМ ЧИСЛЕ, ПРИ ДЕЙСТВИИ КОНТРАКТА, ДЕЛИКТЕ ИЛИ ИНОЙ СИТУАЦИИ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ИНТЕРФЕЙСА ИЛИ ИНЫХ ДЕЙСТВИЙ С ИНТЕРФЕЙСОМ.

Описание интерфейса

Любая команда представляет ASCII строку, которая содержит название команды и список аргументов соответствующих данной команде. Имеются зарезервированные символы, которые не могут входить в название команды и в аргументы команды.

Список зарезервированных символов: «:», «,», «;».

Структура команды:

1. Название команды;
2. Разделительный символ между названием команды и аргументами «:»;
3. Список аргументов через разделительный знак «,»;
4. Символ конца команды «;».

Если аргументов команда не имеет, то после названия команды ставится символ конца команды. Если команда неправильная, то она игнорируется. Регистр букв не имеет значения.

Программа ExpertSDR2 выполняет роль сервера, который может иметь одновременно несколько клиентских подключений, они будут синхронизироваться между собой сервером. При подключении к ExpertSDR2 клиент получает текущее состояние ExpertSDR2, сначала высылаются команды (только для чтения) и затем параметры для установки состояния такие как частота, модуляция и др.

Когда в программе ExpertSDR2(сервер) происходит изменение параметра, то сервер уведомляет об это всех подключённых клиентов, то есть клиентам не нужно постоянно опрашивать сервер, любое изменение состояния будет отправлено своевременно всем клиентам. Если клиент высылает новое состояние, то сервер его установит себе и также вышлет всем клиентам, то есть сервер выполняет роль синхронизатора. Все клиенты, подключённые к серверу, будут автоматически синхронизированы. Такой способ работы позволяет минимизировать нагрузку на сеть, уменьшая трафик.

Протокол TCI предусматривает работу телеграфом с помощью строковых команд.

Команды делятся на два типа:

1. Макрос;
2. Сообщение

Макрос - это набор символов, не содержащий правил, но подчиняющийся командам изменения скорости и передачи аббревиатуры.

Сообщение - это специальная команда, которая состоит из трёх частей:

1. Текст перед позывным;
2. Позывной;
3. Текст после позывного.

При передаче сообщения есть возможность до отправить позывной после передачи сообщения с неполным позывным, также можно менять скорость внутри сообщения и использовать аббревиатуры.

Если нужно внутри текста поместить аббревиатуру, то строка будет иметь вид:

ANY TEXT |**SK**| OTHER TEXT

Все символы, помещённые между вертикальными скобками, будут передаваться слитно.

Если внутри текста нужно уменьшить скорость, то для этого используется символ «<», для увеличения скорости «>» соответственно. Шаг изменения скорости 5 wpm, например:

ANY TEXT > TEXT+5WPM <TEXT-5WPM >>>TEXT+15WPM

Так как текстовые команды могут содержать запрещённые протоколом символы, то они заменяются на другие символы и на стороне сервера снова преобразуются обратно:

1. Символ «**:**» заменяется на «**^**»;
2. Символ «**,**» заменяется на «**~**»;
3. Символ «**;**» заменяется на «*****».

Команда отправки макроса имеет вид:

cw_macros:arg1,arg2;

arg1 - номер программного трансивера;

arg2 - текст передаваемый перед позывным;

Чтобы передать строку «+5wpmTU -5wpmRA6LH +10wpm599 004 *SK*» команда будет иметь вид:

cw_macros:0,>TU <RA6LH >>599 004 |SK|;

Команда отправки телеграфного сообщения с возможностью до отправки позывного:

cw_msg:arg1,arg2,arg3,arg4;

arg1 - номер программного трансивера;

arg2 - текст передаваемый перед позывным;

arg3 - позывной;

arg4 - текст передаваемый после позывного;

Пример:

Чтобы передать строку «TU RA6LH 599 004» команда будет иметь вид:

cw_msg:0,TU,RA6LH,599 004;

Если нужно два раза повторить позывной «TU RA6LH RA6LH 599 004», то команда будет иметь вид:

cw_msg:0,TU,RA6LH$2,599 004;

Если текст отсутствует перед позывным «RA6LH RA6LH 599 004», то вместо текста записывается специальный символ “_”:

cw_msg:0,_,RA6LH$2,599 004;

Если при отправке сообщения ещё полностью не известен позывной и нужно будет его до отправить, и возможно не один раз, то команда будет выглядеть так: cw_msg:arg1;

Пример:

cw_msg:0,_,RA6$2,599 004;

cw_msg:RA6L;

cw_msg:RA6LH;

Если до отправка позывного пришла после того как была закончена передача позывного, то команда игнорируется.

Также поддерживается очередь телеграфных сообщений, если отправить поочерёдно несколько команд cw_msg то они будут переданы по очереди, в таком случае до отправка позывного будет распространяться на передаваемый позывной.

Команды делятся на три группы:

- Запись/Чтение;
- Только чтение;
- Только запись.

Список команд:

START	Запустить программу	
Установить	*START;*	Аргументы:
Тип	*Чтение / Запись*	
Пример	*START;*	

STOP	Остановить программу	
Установить	*STOP;*	Аргументы:
Тип	*Чтение / Запись*	
Пример	*STOP;*	

DDS	Управление центральной частотой настройки приёмника.	
Установить	DDS:*arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — частота настройки, Гц.
Прочитать	DDS:*arg1;*	
Ответ	DDS:*arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	DDS:*0,7200050;* DDS:*1;*	

IF	Управление частотой настройки фильтра ПЧ в пределах панорамы.	
Установить	*IF:arg1,arg2,arg3;*	Аргументы: *arg1* — номер приёмника. *arg2* — номер канала (A / B). *arg3* — частота настройки относительно центральной частоты, Гц.
Прочитать	*IF:arg1,arg2;*	
Ответ	*IF:arg1,arg2,arg3;*	
Тип	*Чтение / Запись*	
Пример	*IF:0,0,-12000;* *IF:0,0,23000;* *IF:1,0;*	

MODULATION	Управление видом модуляции.	
Установить	*MODULATION:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — вид модуляции (строкой). Список поддерживаемых видов модуляции: AM / SAM / DSB / LSB / USB / CW / NFM / WFM / SPEC / DIGL / DIGU / DRM
Прочитать	*MODULATION:arg1;*	
Ответ	*MODULATION:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*MODULATION:0,LSB;* *MODULATION:0,CW;* *MODULATION:1;*	

После смены частотного диапазона, ExpertSDR2 восстанавливает сохраненные настройки выбранного диапазона, к которым относятся модуляция и ширина фильтра приёмника, поэтому при смене частотного диапазона нужно дождаться прихода команд модуляции и ширины фильтра, если они были изменены, либо выждать защитный интервал 200 мс и после этого отправлять команду смены модуляции или команду смены ширины фильтра если это необходимо.

RX_ENABLE	Включение программных приёмников	
Установить	*RX_ENABLE:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — флаг включения.
Прочитать	*RX_ENABLE:arg1;*	
Ответ	*RX_ENABLE:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*RX_ENABLE:1,true;* *RX_ENABLE:2,false;* *RX_ENABLE:1;*	

VFO_LIMITS	Граничные значения частоты настройки.	
Установить		Аргументы: *arg1* — нижняя граничная частота, Гц. *arg2* — верхняя граничная частота, Гц.
Прочитать	*Высылается при подключении;*	
Ответ	*VFO_LIMITS:arg1,arg2;*	
Тип	*Чтение*	
Пример	*VFO_LIMITS:10000,30000000;*	

IF_LIMITS	Граничные значения частот фильтра ПЧ. (в ESDR2 только для VFOA)	
Установить		Аргументы: *arg1* — нижняя граничная частота, Гц. *arg2* — верхняя граничная частота, Гц.
Прочитать	*Высылается при подключении;*	
Ответ	*IF_LIMITS:arg1,arg2;*	
Тип	*Чтение*	
Пример	*IF_LIMITS:-48000,48000;* *IF_LIMITS:-96000,96000;*	

TRX_COUNT	Количество приёмников (трансиверов) в устройстве.	
Установить		Аргументы: *arg1* — количество приёмников/трансиверов (физических или программных).
Прочитать	*Высылается при подключении;*	
Ответ	*TRX_COUNT:arg1;*	
Тип	*Чтение*	
Пример	*TRX_COUNT:2;* *TRX_COUNT:8;*	

CHANNEL_COUNT	Количество дополнительных каналов приёма в одном приёмнике (A/B/C).	
Установить		Аргументы: *arg1* — количество каналов приёма.
Прочитать	*Высылается при подключении;*	
Ответ	*CHANNEL_COUNT:arg1;*	
Тип	*Чтение*	
Пример	*CHANNEL_COUNT:2;* *CHANNEL_COUNT:3;*	

DEVICE	Название устройства	
Установить		Аргументы: *arg1* — название устройства.
Прочитать	*Высылается при подключении;*	
Ответ	*DEVICE:arg1;*	
Тип	*Чтение*	
Пример	*DEVICE:SunSDR2;* *DEVICE:ColibriDDC;*	

RECEIVE_ONLY	Определяет устройство как приёмник или передатчик.	
Установить		Аргументы: *arg1* — только приём (true), трансивер (false).
Прочитать	*Высылается при подключении;*	
Ответ	*RECEIVE_ONLY:arg1;*	
Тип	*Чтение*	
Пример	*RECEIVE_ONLY:true;* *RECEIVE_ONLY:false;*	

MODULATIONS_LIST	Список поддерживаемых видов модуляции.	
Установить		Аргументы: Вид модуляции передаётся названием.
Прочитать	*Высылается при подключении;*	
Ответ	*MODULATIONS_LIST:arg1, arg2, ... ,argN;*	
Тип	*Чтение*	
Пример	*MODULATIONS_LIST:AM,LSB,USB,FM;* *RECEIVE_ONLY:AM,SAM,LSB,USB,CW,NFM,WFM;*	

TX_ENABLE	Разрешение на использование передатчика.	
Установить		Аргументы: *arg1* — номер приёмника/трансивера. *arg2* — передача разрешена (true)/передача запрещена (false).
Прочитать	*Высылается в процессе работы;*	
Ответ	*TX_ENABLE:arg1, arg2;*	
Тип	*Чтение*	
Пример	*TX_ENABLE:0,true;*	

READY	Высылается после команд инициализации при подключении.	
Установить		
Прочитать	*Высылается при подключении*	
Ответ	*READY;*	
Тип	*Чтение*	
Пример		

TRX	Переключение режимов приём передача.	
Установить	*TRX:arg1,arg2, arg3;*	Аргументы: *arg1* — номер приёмника. *arg2* — флаг включения. *arg3* — источник сигнала (необязательно)(mic - микрофонный сигнал, vac - сигнал из VAC)
Прочитать	*TRX:arg1;*	
Ответ	*TRX:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*TRX:0,true;* *TRX:0,true,mic;* *TRX:0,true,vac;* *TRX:0,false;* *TRX:1;*	
Для обычного использования команда TRX может иметь два аргумента, в таком случае сервер автоматически будет определять источник входного сигнала на передачу, сигнал из VAC будет использоваться передатчиком при выборе вида модуляции DIGL или DIGU и одновременно включённом VAC-е. Во всех остальных случаях будет использоваться сигнал микрофона. Если нужно явно указать источник сигнала, то используется третий аргумент: mic - микрофонный сигнал, vac - сигнал из VAC.		

RIT_ENABLE	Включение расстройки по приёму.	
Установить	*RIT_ENABLE:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — флаг включения.
Прочитать	*RIT_ENABLE:arg1;*	
Ответ	*RIT_ENABLE:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*RIT_ENABLE:0,true;* *RIT_ENABLE:0,false;* *RIT_ENABLE:1;*	

XIT_ENABLE	Включение расстройки по передаче.	
Установить	*XIT_ENABLE:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — флаг включения.
Прочитать	*XIT_ENABLE:arg1;*	
Ответ	*XIT_ENABLE:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*XIT_ENABLE:0,true;* *XIT_ENABLE:0,false;* *XIT_ENABLE:1;*	

SPLIT_ENABLE	Включение режима split.	
Установить	*SPLIT_ENABLE:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — флаг включения.
Прочитать	*SPLIT_ENABLE:arg1;*	
Ответ	*SPLIT_ENABLE:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*SPLIT_ENABLE:0,true;* *SPLIT_ENABLE:0,false;* *SPLIT_ENABLE:1;*	

RIT_OFFSET	Управление частотой расстройки приёмника.	
Установить	*RIT_OFFSET:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — частота расстройки, Гц.
Прочитать	*RIT_OFFSET:arg1;*	
Ответ	*RIT_OFFSET:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*RIT_OFFSET:0,500;* *RIT_OFFSET:0,-200;* *RIT_OFFSET:1;*	

XIT_OFFSET	Управление частотой расстройки передатчика.	
Установить	*XIT_OFFSET:arg1,arg2;*	Аргументы: *arg1* — номер приёмника. *arg2* — частота расстройки, Гц.
Прочитать	*XIT_OFFSET:arg1;*	
Ответ	*XIT_OFFSET:arg1,arg2;*	
Тип	*Чтение / Запись*	
Пример	*XIT_OFFSET:0,500;* *XIT_OFFSET:0,-200;* *XIT_OFFSET:1;*	

RX_CHANNEL_ENABLE	Включение дополнительных каналов приёма.	
Установить	*RX_CHANNEL_ENABLE:arg1,arg2,arg3;*	Аргументы: *arg1* — номер приёмника. *arg2* — номер канала. *arg3* — флаг включения.
Прочитать	*RX_CHANNEL_ENABLE:arg1,arg2;*	
Ответ	*RX_CHANNEL_ENABLE:arg1,arg2,arg3;*	
Тип	*Чтение / Запись*	
Пример	*RX_CHANNEL_ENABLE:0,1,true;* *RX_CHANNEL_ENABLE:0,1,false;* *RX_CHANNEL_ENABLE:1, 1;*	

RX_FILTER_BAND	Управление шириной фильтра основной селекции (фильтр приемника).	
Установить	*RX_FILTERL_BAND:arg1,arg2,arg3;*	Аргументы: *arg1* — номер приёмника. *arg2* — нижняя частота среза, Гц. *arg3* — верхняя частота среза, Гц.
Прочитать	*RX_FILTER_BAND:arg1,arg2;*	
Ответ	*RX_FILTER_BAND:arg1,arg2,arg3;*	
Тип	*Чтение / Запись*	
Пример	*RX_FILTER_BAND:0,30,2700;* *RX_FILTER_BAND:0,-2900,-70;* *RX_FILTER_BAND:1;*	

После смены частотного диапазона, ExpertSDR2 восстанавливает сохраненные настройки выбранного диапазона, к которым относятся модуляция и ширина фильтра приёмника, поэтому при смене частотного диапазона нужно дождаться прихода команд модуляции и ширины фильтра, если они были изменены, либо выждать защитный интервал 200 мс и после этого отправлять команду смены модуляции или команду смены ширины фильтра если это необходимо.

RX_SMETER	Получение уровня сигнала в полосе приёма (в dBm).	
Установить	*RX_SMETER:arg1,arg2,arg3;*	Аргументы: *arg1* — номер приёмника. *arg2* — номер канала. *arg3* — уровень сигнала.
Прочитать	*RX_SMETER:arg1,arg2;*	
Ответ	*RX_SMETER:arg1,arg2,arg3;*	
Тип	*Чтение / Запись*	
Пример	*RX_SMETER:0,0,-72;* *RX_SMETER:0,1,-63;* *RX_SMETER:1,0;*	

CW_MACROS_SPEED	Управление скоростью телеграфирования для макросов.	
Установить	*CW_MACROS_SPEED:arg1;*	Аргументы: *arg1* — скорость телеграфирования, WPM.
Прочитать	*CW_MACROS_SPEED;*	
Ответ	*CW_MACROS_SPEED:arg1;*	
Тип	*Чтение / Запись*	
Пример	*CW_MACROS_SPEED:30;* *CW_MACROS_SPEED:42;* *CW_MACROS_SPEED;*	

CW_MACROS_DELAY	Управление задержкой перед началом телеграфирования после переключения в TX.	
Установить	*CW_MACROS_DELAY:arg1;*	Аргументы: *arg1* — задержка перед началом телеграфирования, мс.
Прочитать	*CW_MACROS_DELAY;*	
Ответ	*CW_MACROS_DELAY:arg1;*	
Тип	*Чтение / Запись*	
Пример	*CW_MACROS_DELAY:100;* *CW_MACROS_DELAY:150;* *CW_MACROS_DELAY;*	

SPOT	Передача спота на отображение в ESDR2.	
Установить	*SPOT:arg1,arg2,arg3,arg4,arg5;*	Аргументы: *arg1* — позывной. *Arg2* — модуляция. *Arg3* — частота, Гц. *Arg4* — цвет ARGB. *Arg5* — дополнительный текст.
Прочитать		
Ответ		
Тип	*Запись*	
Пример	*SPOT:RN6LHF,CW,7100000,16711680,ANY_TEXT;*	
Цвет кодируется 32 битным беззнаковым числом *0x00FF0000* -> *16711680*		

SPOT_DELETE	Удалить спот.	
Установить	*SPOT:arg1;*	Аргументы: *arg1* — позывной.
Прочитать		
Ответ		
Тип	*Запись*	
Пример	*SPOT_DELETE:IT8TY;*	

IQ_SAMPLERATE	Управление частотой дискретизации IQ сигнала.	
Установить	*IQ_SAMPLERATE:arg1;*	Аргументы: *arg1* — частота дискретизации, Гц. Поддерживаемые частоты дискретизации: 48 / 96 / 192 кГц
Прочитать		
Ответ		
Тип	*Чтение / Запись*	
Пример	*IQ_SAMPLERATE:48000;* *IQ_SAMPLERATE:96000;* *IQ_SAMPLERATE:192000;*	

IQ_START	Запустить вывод IQ сигнала.	
Установить	*IQ_START:arg1;*	Аргументы:
Тип	*Чтение / Запись*	
Пример	*IQ_START:0;*	*arg1* — номер приёмника.

IQ_STOP	Остановить вывод IQ сигнала.	
Установить	*IQ_STOP:arg1;*	Аргументы:
Тип	*Чтение / Запись*	
Пример	*IQ_STOP:0;*	*arg1* — номер приёмника.

TX_FOOTSWITCH	Сигнал нажатия на педаль PTT	
Установить	*TX_FOOTCWITCH:arg1,arg2;*	Аргументы:
Тип	*Чтение*	
Пример	*TX_FOOTCWITCH:0,true;* *TX_FOOTCWITCH:0,false;*	*arg1* — номер приёмника. *Arg2* — состояние педали (нажата (true), отжата (false))

Получение IQ сигнала выполняется через бинарное сообщение websocket, структура пакета имеет вид:

```
typedef struct
{
    quint32 receiver;       //!< номер приёмника
    quint32 sampleRate;   //!< частота дискретизации
    quint32 format;       //!< всегда равен 4 (float 32 bit)
    quint32 codec;        //!< алгоритм сжатия (не реализовано), всегда 0
    quint32 crc;          //!< контрольная сумма (не реализовано), всегда 0
    quint32 length;       //!< длина поля данных
    quint32 type;         //!< тип потока данных
    quint32 reserv[9];     //!< зарезервировано
    float   data[4096];    //!< поле данных
}DataStream;
```

Тип потока данных определяется следующим перечислением:

```
typedef enum
{
    IqStream = 0,         //!< Поток IQ сигнала приёмника
    RxAudioStream,     //!< Аудио поток приёмника (не реализовано)
    TxAudioStream,     //!< Аудио поток для передатчика (не реализовано)
    TxChrono,         //!< Поток маркеров времени для передачи аудио сигнала (не реализовано)

}StreamType;
```

Список программ, поддерживающих TCI

- SDC
- LogHX
- SWISSLOG

Заключение

Интерфейс TCI будет постепенно развиваться, в него будут добавляться не реализованные на текущий момент команды и функции. Следите за обновлениями TCI интерфейса.