Описание программного интерфейса контроллера GB/T при взаимодействии по RPC

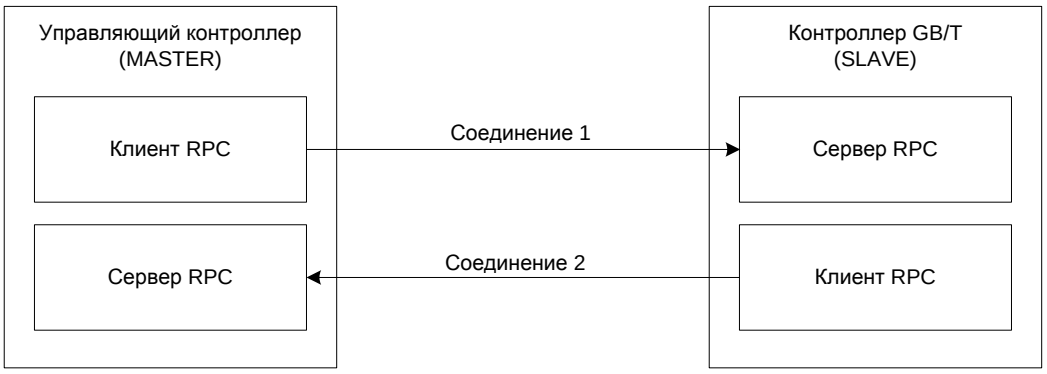
1. Общие сведения

Интерфейс взаимодействия с потребителем – Ethernet. Программный интерфейс взаимодействия основан на использовании удаленного вызова процедур (RPC) по сетевому протоколу TCP IP v4. Используемый формат сообщений msgpack-rpc. Для данного протокола взаимодействия существуют библиотеки с открытым исходным кодом под разные платформы и языки программирования.

Контроллер CCS реализует одновременно сервер RPC (для приема команд от управляющего контроллера) и клиент RPC (для передачи команд на управляющий контроллер).

IP адрес сервера RPC контроллера CCS: 192.168.3.223.

Порт сервера RPC контроллера CCS: 19000.



Обмен информацией между управляющим контроллером и контроллером GB/T осуществляется по схеме Master-Slave, при этом контроллер GB/T выступает в роли Slave. Для инициализации соединения клиент RPC управляющего контроллера подключается к серверу RPC контроллера GB/T и выполняет метод «rpcConnectRequest»:

string rpcConnectRequest(string interfaceId, string remoteAddress, int remotePort, time_t connectionTimeoutMsec, time_t pingPeriodMsec,uint32_t pingCheckCount);

interfaceId – идентификатор интерфейса RPC, должен быть равен "IID_SECC_GBT_1.0";

remoteAddress – собственный IP адрес управляющего контроллера;

remotePort – порт сервера RPC управляющего контроллера;

connectionTimeoutMsec – таймаут соединения TCP, мсек;

pingPeriodMsec – требуемый период передачи пингов, мсек;

pingCheckCount – количество пропущенных пингов, после которого соединение считается разорванным.

После получения вызова «rpcConnectRequest» клиент RPC контроллера GB/T выполняет обратное подключение к серверу RPC управляющего контроллера по указанному IP адресу и порту и начинает с периодичностью pingPeriodMsec вызывать метод «rpcPing» для поддержания соединения:

void rpcPing(int selfConnectionInputState, int selfConnectionOutputState);

selfConnectionInputState – принимаются ли пинги с удаленной стороны (1 – пингов нет, 2 – пинги есть);

selfConnectionOutputState – отправляются ли пинги к удаленной стороне (1 – ошибка отправки пингов, 2 – пинги отправляются).

Клиент RPC управляющего контроллера после установки соединения также должен с периодичностью pingPeriodMsec осуществлять вызов метода «rpcPing» для поддержания соединения и передавать актуальные значения selfConnectionInputState, selfConnectionOutputState.

Если сервер RPC контроллера GB/T после установки соединения не получает пинги в течение времени pingPeriodMsec * pingCheckCount или происходит таймаут соединения TCP, то соединение клиента RPC контроллера CCS к серверу RPC управляющего контроллера разрывается и устанавливается заново.

2. Процедуры программного интерфейса, исполняемые контроллером (RPC сервер)

2.1. Перезагрузка контроллера

void RESET();

2.2. Разрешение зарядной сессии

void AUTHORIZE();

2.3. Остановка зарядной сессии

void USER_STOP();

2.4. Установка лимитов станции для зарядной сессии

void SET_INVERTOR_LIMITS(float maximumPowerLimitW, float maximumVoltageLimitV, float maximumCurrentLimitA, float minimumVoltageLimitV, float minimumCurrentLimitA);

maximumPowerLimitW – максимальная выходная мощность, Вт;

maximumVoltageLimitV – максимальное выходное напряжение, В;

maximumCurrentLimitA – максимальный выходной ток, А;

minimumVoltageLimitV – минимальное выходное напряжение, В;

minimumCurrentLimitA – минимальный выходной ток, А;

2.5. Установка текущего статуса силовой части

void SET_INVERTOR_STATE(bool isPowerOn, bool isInvertorOn, bool isInvertorError, bool isInterfaceError);

isPowerOn – на силовые преобразователи подается питающее напряжение;

isInvertorOn – на выход зарядной станции подается напряжение;

isInvertorError –наличие любой ошибка в силовой части;

isInterfaceError – отсутствие обмена информацией с силовыми преобразователями;

2.6. Установка текущих параметров зарядной сессии

void SET_INVERTOR_PRESENT_PARAMS(float presentVoltageV, float presentCurrentA);

presentVoltageV – текущее напряжение на выходе зарядной станции, В;

presentCurrentA – текущий ток на выходе зарядной станции, А;

2.7. Установка текущих параметров контроля изоляции (;; статус изоляции)

void SET_ISOLATION_STATE(bool isIsolationMonitoring, bool isImdTest, string isolationLevel);

isIsolationMonitoring – активность устройства контроля изоляции;

isImdTest – идет самотестирование устройства контроля изоляции;

isolationLevel – статус изоляции, может принимать следующие значения:

– INVALID – тест изоляции еще не проведен;

– VALID – измеренное сопротивление изоляции в норме;

– WARNING – измеренное сопротивление изоляции ниже уровня предупреждения в соответствии с IEC 61851-23;

– FAULT – измеренное сопротивление изоляции ниже допустимого уровня в соответствии с IEC 61851-23;

– NO_IMD – устройство контроля изоляции отсутствует;

3. Процедуры, вызываемые контроллером (rpc клиент)

Процедуры вызываются контроллером при каждом изменении одного из параметров.

3.1. Передача текущей версии ПО контроллера

void SET_VERSION(string version);

version – текущая версия ПО контроллера;

3.2. Передача текущей стадии зарядной сессии

void SET_SECC_CURRENT_STATE(string currentState);

currentState – текущая стадия зарядной сессии, может принимать следующие значения:

– DISCONNECTED – режим ожидания начала зарядной сессии;

– CONNECTED – зарядный кабель подключен;

– HANDSHAKE – рукопожатие;

– INSULATION_TEST – контроль изоляции зарядного кабеля;

– PARAMETERS_CONFIG – обмен с электромобилем параметрами заряда;

– PRECHARGE – предзаряд (выравнивание напряжения на выходе силовых преобразователей зарядной станции с напряжением на АКБ электромобиля);

– CHARGE – заряд;

– WELDING_DETECTION – проверка “сварки” контактов;

– SESSION_STOP – завершение зарядной сессии;

– ERROR – зарядная сессия завершена с ошибкой;

– STOP – зарядная сессия завершена без ошибок;

3.3. Передача максимальных зарядных лимитов электромобиля

void SET_EV_LIMITS(float maximumVoltageLimitV, float maximumCurrentLimitA);

maximumVoltageLimitV – максимальное допустимое зарядное напряжение, В;

maximumCurrentLimitA – максимальный допустимый ток заряда, А;

3.4. Управление силовыми преобразователями зарядной станции

void SET_EV_TARGET_PARAMS(bool invertorSwitchOn, bool invertorContactorsOn, bool insulationControlOn, float targetVoltageV, float targetCurrentA);

invertorSwitchOn – включение силовых преобразователей;

invertorContactorsOn – включение выходных контакторов;

insulationControlOn – управление работой устройством контроля изоляции;

targetVoltageV – заданное выходное напряжение, В;

targetCurrentA – заданный выходной ток, А;

3.5. Передача параметров электромобиля

void SET_EV_PARAMS(string vin, float batteryNominalEnergy);

vin – VIN номер электромобиля;

batteryNominalEnergy – номинальная емкость батареи электромобиля, кВт*час;

3.6. Передача готовности электромобиля к заряду

void SET_EV_STATE(bool evReady);

evReady – электромобиль готов к заряду;

3.7. Передача состояния электромобиля

void SET_EV_SOC(float evSoc, float estimatedRemainingChargingTime);

evSoc – уровень заряда АКБ электромобиля, %;

estimatedRemainingChargingTime – прогнозируемое электромобилем время до окончания зарядной сессии, с;

3.8. Передача причины завершения зарядной сессии

void SET_ERROR_CODE(uint32_t errorCode, string errorCode_str);

errorCode – код ошибки;

errorCode_str – сообщение об ошибке;

Используемые коды и соответствующие сообщения об ошибках приведены в таблице:

Код ошибки	Сообщение об ошибке	Примечание
0		Нет ошибок
1	cableConnectionError	Обнаружено отсоединение кабеля
2	cableLockError	Обнаружен не запертый замок зарядного кабеля
3	cableTemperatureError	Обнаружено превышение температуры контактов зарядного кабеля
4	canError	Ошибка обмена с электромобилем
5	evError	Ошибка электромобиля во время зарядной сессии
6	seccError	Ошибка зарядной станции во время зарядной сессии