Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ПО ИТОГОВОЙ РАБОТЕ

Специальность 09.02.07 «Информационные системы и программирование»

Дисциплина «Разработка программных модулей»

Преподаватель:
Зенкин А.М.
«14» июня 2021г.
Оценка:

Выполнили:
студенты группы Y2334
Панаёт В. Т., Ганькин В. А.,
Панаёт Р. Т., Котлярова С. Ю.

Санкт-Петербург
2020/2021

ЦЕЛЬ РАБОТЫ

Изучить и проанализировать модули прошивки потенциостата.

ХОД РАБОТЫ

potentiostat

```
#ifndef POTENTIOSTAT_H
#define POTENTIOSTAT_H

#include "ps_system_state.h"

#endif
```

Исполняет содержимое файла "ps_system_state.h", если был открыт впервые.

ps_analog_subsystem

```
#ifndef PS_ANALOG_SUBSYSTEM_H
#define PS_ANALOG_SUBSYSTEM_H

#include "ps_hardware_defs.h"
#include "ps_pin_map.h"
#include "ps_gains.h"
#include "ps_volt_range.h"
#include "ps_curr_range.h"
#include "ps_return_status.h"

namespace ps
{

    class AnalogSubsystem
    {
        public:

            static const uint16_t DefaultAnalogWriteResolution = 12;
            static const uint16_t DefaultAnalogReadResolution = 16;
            static const uint16_t DefaultAnalogReadAveraging = 16;
            static const uint8_t  DefaultAnalogReference = INTERNAL;

            static const uint16_t MaxValueAin = uint16_t((uint32_t(1) << DefaultA
nalogReadResolution) -1);
            static const uint16_t MaxValueDac = uint16_t((uint32_t(1) << DefaultA
nalogWriteResolution)-1);
            static const uint16_t MidValueDac = MaxValueDac/2;

            AnalogSubsystem();
```

```
            void initialize();

            void setVolt(float value);
            float getVolt() const;

            float getCurr() const;
            float getRefElectVolt() const;

            void setVoltRange(VoltRangeDac range);
            VoltRangeDac getVoltRange() const;
            bool autoVoltRange(float minVolt, float maxVolt);

            void setCurrRange(CurrRange range);
            CurrRange getCurrRange() const;

            ReturnStatus setVoltRangeByName(String voltRangeName);
            String getVoltRangeName() const;

            ReturnStatus setCurrRangeByName(String currRangeName);
            String getCurrRangeName() const;

#if defined HARDWARE_VERSION_0P2
            void setRefElectVoltRange(VoltRangeAdc range);
            VoltRangeAdc getRefElectVoltRange() const;

            ReturnStatus setRefElectVoltRangeByName(String voltRangeName);
            String getRefElectVoltRangeName() const;
            bool autoRefElectVoltRange(float minVolt, float maxVolt);
#endif

        protected:

            uint16_t valueDac_;
            VoltRangeDac voltRange_;
            CurrRange currRange_;

            void setVoltGain(VoltGain value);
            VoltGain getVoltGain() const;

            void setCurrGainPath(CurrGainPath value);
            CurrGainPath getCurrGainPath() const;

            String getVoltGainString() const;
            String getCurrGainPathString() const;

            void setValueDac(uint16_t value);
            uint16_t getValueDac() const;

            uint16_t getTransAmpAin() const;
            uint16_t getRefElectAin() const;
```

```
            VoltRangeAdc refElectVoltRange_;
#if defined HARDWARE_VERSION_0P2
            void setRefElectVoltGain(VoltGain value);
            VoltGain getRefElectVoltGain() const;
#endif

    };

} // namespace ps


#endif
```

Подсистема для работы с вольтажом электродов, током трансимпедансного усилителя и отдельно для работы с вольтажом электрода сравнения (только для второй версии железа). Под железо второй версии расширен функционал. Под каждое железо своя настройка пинов по умолчанию. Настройка по умолчанию происходит при первой инициализации.

Методы

Для пользователя:

1. void initialize() – инициализация, установка значений по умолчанию.
2. void setVolt(float *value*)установление значения исходящего напряжения на электрод сравнения
3. float getVolt() const – получить текущее значение
4. float getCurr() const – получить измерение тока с рабочего электрожа. Проверить на потенциостате.
5. float getRefElectVolt() const - получить измерение напряжения электрода сравнения
6. void setVoltRange(VoltRangeDac *range*) – установить промежуток исходящего напряжения для электрода сравнения. Изменяет значение усиления напряжения.
7. VoltRangeDac getVoltRange() const – возвращает текущий промежуток напряжения.
8. bool autoVoltRange(float *minVolt*, float *maxVolt*) – автоматическое установление промежутка напряжения. Алгоритм:
9. void setCurrRange(CurrRange *range*) – установить диапазон тока для трансимпедансного усилителя.
10. CurrRange getCurrRange() const – получить текущий диапазон тока для трансимпедансного усилителя.
11. ReturnStatus setVoltRangeByName(String *voltRangeName*) - установить диапазон напряжения по его имени.
12. String getVoltRangeName() const – получить имя диапазона напряжения

13. ReturnStatus setCurrRangeByName(String *currRangeName*) - установить диапазон тока по его имени.
14. String getCurrRangeName() const – получить имя диапазона тока;
15. void setRefElectVoltRange(VoltRangeAdc *range*) - установить диапазон напряжения на аналоговом входе электрода для электрода сравнения;
16. VoltRangeAdc getRefElectVoltRange() const - получить диапазон напряжения на аналоговом входе электрода для электрода сравнения;
17. ReturnStatus setRefElectVoltRangeByName(String *voltRangeName*) - установить диапазон напряжения на аналоговом входе электрода по его имени.
18. String getRefElectVoltRangeName() const--получить имя диапазона напряжения на аналоговом входе электрода для электрода сравнения;
19. bool autoRefElectVoltRange (float *minVolt*, float *maxVolt*) – автоматическая установка диапазона напряжения для электрода сравнения;

Внутренние:

1. void setVoltGain(VoltGain *value*) - установить коэф. усиления для значения напряжения электрода сравнения
2. VoltGain getVoltGain() const – считать усиление. Специфично для версии железа и варианта вольтажа
3. void setCurrGainPath(CurrGainPath *value*) - установить путь усиления. Описание путей усиления ниже.
4. CurrGainPath getCurrGainPath() const - получить текущий путь усиления
5. String getVoltGainString() const - строковое представление
6. String getCurrGainPathString() const - строковое представление
7. void setValueDac(uint16_t *value*) – установить выходное напряжение с ЦАП
8. uint16_t getValueDac() const – получить выходное напряжение с ЦАП
9. uint16_t getTransAmpAin() const - считывание аналогового входа, связанного с трансимпедансным усилителем
10. uint16_t getRefElectAin() const - считывание аналогового входа, связанного с электродом сравнения
11. void setRefElectVoltGain(VoltGain *value*) – устанавливает усиление для электрода сравнения. Только для второй версии железа.
12. VoltGain getRefElectVoltGain() const – получение текущего усиления. Считывает значения электрода сравнения и проверяет их.

ps_base_test

```
#ifndef PS_BASE_TEST_H
#define PS_BASE_TEST_H

#include <Arduino.h>
#include "ps_sample.h"
#include "ps_constants.h"
#include "ps_return_status.h"

#include "third-party/ArduinoJson/ArduinoJson.h"

namespace ps
{

    class BaseTest
    {

        public:

            BaseTest();

            virtual bool isDone(uint64_t t) const;
            virtual uint64_t getDoneTime() const;
            virtual void reset();

            virtual float getValue(uint64_t t) const;
            virtual float getMaxValue() const;
            virtual float getMinValue() const;

            virtual void setQuietTime(uint64_t quietTime);
            virtual uint64_t getQuietTime() const;

            virtual void setQuietValue(float value);
            virtual void setQuietValueToStart();
            virtual float getQuietValue() const;

            virtual void setSamplePeriod(uint64_t samplePeriod);
            virtual uint64_t getSamplePeriod() const;

            virtual void setName(String name);
            virtual String getName();

            virtual void setSampleMethod(SampleMethod sampleMethod);
            virtual SampleMethod getSampleMethod() const;
            virtual bool updateSample(Sample sampleRaw, Sample &sampleTest);

            virtual void getParam(JsonObject &jsonDat);
            virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDa
t);
```

```
            virtual bool isMuxCompatible();
            virtual void setMuxCompatible(bool value);

        protected:

            uint64_t quietTime_ = 0;
            float quietValue_ = 0.0;
            uint64_t samplePeriod_ = 0;
            uint32_t sampleModulus_ = 0;
            String name_ = String("base");
            SampleMethod sampleMethod_ = SampleGeneric;
            bool muxCompatible_ = false;

            JsonObject &getParamJsonObject(JsonObject &json, ReturnStatus &status
);
            void setQuietValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDa
tPrm, ReturnStatus &status);
            void setQuietTimeFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDat
Prm, ReturnStatus &status);


    };

}

#endif
```

Модуль для создания основы теста. Имеет четыре важных параметра - время молчания, значение вольт на время молчания, метод выборки и время между выборками.

Методы

Для пользователя:

virtual bool isDone(uint64_t *t*) const - узнать исполнен ли тест

virtual uint64_t getDoneTime() const - узнать время исполнения теста

virtual void reset() - сброс теста

Получение значений:

virtual float getValue(uint64_t *t*) const;

virtual float getMaxValue() const;

virtual float getMinValue() const;

Установление и получение времени молчания

virtual void setQuietTime(uint64_t *quietTime*);

virtual uint64_t getQuietTime() const;

Установление и получение значения на время молчания

virtual void setQuietValue(float *value*);

virtual void setQuietValueToStart() - в начале следующего теста

virtual float getQuietValue() const;

Установление и получение периода времени между выборками

virtual void setSamplePeriod(uint64_t *samplePeriod*);

virtual uint64_t getSamplePeriod() const;

Установление и получение имени теста

virtual void setName(String *name*);

virtual String getName();

Установление и получение выборки данных при тесте

virtual void setSampleMethod(SampleMethod *sampleMethod*);

virtual SampleMethod getSampleMethod() const;

Редактировать данные

virtual bool updateSample(Sample *sampleRaw*, Sample &*sampleTest*) - метод не работает. М. б. дописать?

Работа с JSON файлами

virtual void getParam(JsonObject &*jsonDat*) - получить текущие параметры () в JSON файле

virtual ReturnStatus setParam(JsonObject &*jsonMsg*, JsonObject &*jsonDat*) - установление параметров из JSON файла

Совместим ли тест с мультиплексором?

virtual bool isMuxCompatible();

virtual void setMuxCompatible(bool *value*); установить да или нет

Внутренние:

1. JsonObject &getParamJsonObject(JsonObject &json, ReturnStatus &status) - получить параметр с JSON файла. Обработаны исключения когда подан не JSON файл и когда нет параметров в файле.
2. void setQuietValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status) - установить значение для тест из JSON файла. Обработаны исключения, когда тип не float.

3. void setQuietTimeFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status) - установить время на тест из JSON файла. Обработаны исключения, когда тип не unsigned long (время не может быть отрицательным).

ps_circular_buffer

```
#ifndef PS_CIRCULAR_BUFFER_H
#define PS_CIRCULAR_BUFFER_H

#include <Arduino.h>
#include "third-party/Array/Array.h"

namespace ps
{
    template<typename T, size_t MAX_SIZE>
    class CircularBuffer
    {

        public:

            CircularBuffer();
            T& front();
            T& back();
            T& operator[](const size_t index);
            T  operator[](const size_t index) const;
            void push_back(const T &value);
            void push_front(const T &value);
            void pop_front();
            void pop_back();
            void clear();
            bool empty() const;
            bool full() const;
            size_t size() const;
            size_t max_size() const;
            size_t pos_front() const;
            size_t pos_back() const;

        protected:

            Array<T,MAX_SIZE+1> data_;
            volatile size_t  pos_front_ = 0;
            volatile size_t  pos_back_  = 0;

    };
```

Реализация циркулярного буфера для messege_receiver.

Циркулярный буфер — это структура данных, использующая единственный буфер фиксированного размера, как будто бы после последнего элемента

сразу же снова идет первый. Такая структура легко предоставляет возможность буферизации потоков данных. Перезаписывает наиболее старые данные на новые.

ps_command_table

```
#ifndef PS_COMMAND_TABLE_H
#define PS_COMMAND_TABLE_H

#include <Arduino.h>
#include "ps_return_status.h"
#include "ps_keyvalue_command.h"
#include "ps_constants.h"

#include "third-party/ArduinoJson/ArduinoJson.h"
#include "third-party/Array/Array.h"

namespace ps
{

    template<typename T, size_t MAX_SIZE>
    class CommandTable
    {

        public:

            CommandTable(T *client=nullptr);

            void clear();
            void clearTable();

            size_t size();
            size_t maxSize();

            void setClient(T *client);
            void registerMethod(String key, String value, ReturnStatus (T::*metho
d)(JsonObject&,JsonObject&));

            ReturnStatus apply(String key, JsonObject &jsonMsg, JsonObject &jsonD
at);

        protected:

            T *client_;
            Array<KeyValueCommand<T>,MAX_SIZE> table_;

    };
```

реализация командной строки.

Методы

1. Table – массив с зарегистрированными методами.
2. clearTable – очистить табло
3. clear – очистить табло и клиента, создавшего табло
4. size – размер табло
5. maxSize – максимальный размер табло (установлен в коде)
6. registerMethod – зарегистрировать метод в табло
7. apply – получить результат команды. Прописаны следующие исключение: клиент не найден, команда не найдена.

ps_constant_test

```
#ifndef PS_CONSTANT_TEST_H
#define PS_CONSTANT_TEST_H

#include "ps_base_test.h"

namespace ps
{

    class ConstantTest : public BaseTest
    {

        public:

            ConstantTest();

            void setDuration(uint64_t duration);
            uint64_t getDuration() const;

            void setValue(float value);
            float getValue();

            virtual float getValue(uint64_t t) const override;

            virtual bool isDone(uint64_t t) const override;
            virtual uint64_t getDoneTime() const override;
            virtual float getMaxValue() const override;
            virtual float getMinValue() const override;
            virtual void getParam(JsonObject &jsonDat) override;
            virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDa
t) override;

        protected:

            uint64_t duration_ = 5000000;
            float value_ = 1.0;
```

```
            void setDurationFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatP
rm, ReturnStatus &status);
            void setValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm,
 ReturnStatus &status);
    };

} // namespace ps

#endif
```

тест, где потенциал между рабочим и эталонным электродами остается постоянным. Как и другие испытания, испытание постоянным напряжением включает в себя период молчания, в течение которого выходное напряжение удерживается, и постоянное значение (напряжение молчания) в течение фиксированной продолжительности до начала испытания.

Параметры:

Параметр	Ед. изм	Тип данных
время молчания	мс.	uint_64
напряжение во время молчания	В.	float
напряжение	мс.	float
продолжительность теста	В.	uint_64

Для пользователя:

ConstantTest() - конструктор, в котором имя теста ставят constant и включают совместимость с мультиплексором.

Установить и получить продолжительность теста

- void setDuration(uint64_t duration);
- uint64_t getDuration() const;

Установить и получить исходящее напряжение

- void setValue(float value);
- float getValue();
- virtual float getValue(uint64_t t) const override; - получить quiet_value

Готов ли тест и за сколько

- virtual bool isDone(uint64_t t) const override; - готов ли тест
- virtual uint64_t getDoneTime() const override; - время выполнения теста

Максимальное и минимальное значение напряжения

- virtual float getMaxValue() const override;
- virtual float getMinValue() const override;

JSON

- virtual void getParam(JsonObject &jsonDat) override - Получить тек. параметры в JSON файл
- virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDat) override; - установить параметры из JSON файла

Внутренние:

Установить продолжительность и значение из JSON файла

1. void setDurationFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);
2. void setValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);

ps_constants

– константы, используемые в зависимости от версии железа, входного тока и напряжения

ps_curr_range

```
#ifndef PS_CURR_RANGE_H
#define PS_CURR_RANGE_H

#include <Arduino.h>
#include "ps_range.h"
#include "ps_gains.h"

namespace ps
{

    class CurrRange : public Range<CurrGainPath,uint16_t>
    {
        public:
            CurrRange() : Range<CurrGainPath,uint16_t>()  {};
```

```
            CurrRange(String name, float minValue, float maxValue, CurrGainPath c
urrGainPath, uint16_t maxInt)
                : Range<CurrGainPath,uint16_t>(name, minValue, maxValue, currGain
Path, maxInt) {};

    };

}

#endif
```

Range по текущему CurrGainPath

ps_device_id_eeprom

```
#ifndef PS_DEVICE_ID_EEPROM_H
#define PS_DEVICE_ID_EEPROM_H

#include "ps_constants.h"
#include "ps_return_status.h"

#include "third-party/ArduinoJson/ArduinoJson.h"

namespace ps
{

    class DeviceId_EEPROM
    {

        public:

            DeviceId_EEPROM(uint32_t address=EEPROM_DeviceIdAddress);
            ReturnStatus set(JsonObject &jsonMsg, JsonObject &jsonDat);
            void get(JsonObject &jsonDat);

        protected:

            uint32_t address_;

    };

} // namespace ps

#endif
```

сохранение в памяти id устройства

ps_electrode_switch

```
#ifndef PS_ELECTRODE_SWITCH_H
#define PS_ELECTRODE_SWITCH_H
```

```
#include "ps_hardware_defs.h"

#if defined HARDWARE_VERSION_0P2

namespace ps
{
    class ElectrodeSwitch
    {
        public:
            ElectrodeSwitch();
            void initialize();

            bool connected(int pin);
            void setConnected(int pin, bool value);

            bool ctrConnected();
            void setCtrConnected(bool value);

            bool refConnected();
            void setRefConnected(bool value);

            bool wrkConnected();
            void setWrkConnected(bool value);

            bool allConnected();
            void setAllConnected(bool value);

    };

}
#endif // if defined HARDWARE_VERSION
#endif // ifndef PS_ELECTRODE_SWITCH_H
```

переключатель и проверка состояния «соединён» электродов. Работает только на второй версии железа.

ps_filter

```
#ifndef PS_FILTER_H
#define PS_FILTER_H

namespace ps
{

    struct LowPassParam
    {
        float cutoffFreq;   //  cutoff frequency (-3dB) for fileter cascade
        float initialValue;
        unsigned int order;
    };
```

```
    // Cascaded first order low-pass filters
    class LowPass
    {
        public:

            static const unsigned int MaxOrder_ = 5;
            static const unsigned int MinOrder_ = 1;

            LowPass(float cutoff_freq=1.0, unsigned int order=1, float value=0.0)
;
            LowPass(LowPassParam param);

            void setParam(float cutoff_freq, unsigned int order, float value);
            void setParam(LowPassParam param);
            LowPassParam param();

            float cutoffFreq();
            void setCutoffFreq(float cutoffFreq);

            float initialValue();
            void setInitialValue(float initialValue);

            float order();
            void setOrder(unsigned int order);
            float singleStageRC();
            float singleStageCutoffFreq();

            void update(float value, float dt);
            void reset();
            float value() const;

        protected:

            LowPassParam param_;
            volatile float state_[MaxOrder_+1];
            float rc_;
            float elemCutoffFreq_;
            void initializeState();
    };

} // namespace filter

#endif
```

модуль реализуют фильтр нижних частот.

ps_gains

```
#ifndef PS_GAINS_H
#define PS_GAINS_H
```

```
#include <Arduino.h>
#include "ps_hardware_defs.h"

namespace ps
{

#if defined(VOLTAGE_VARIANT_AD8250) || defined(VOLTAGE_VARIANT_10V)

    enum VoltGain // Analog output voltage scaling factor
    {
        VoltGain1X  = 0,   // [-1V,  +1V]
        VoltGain2X  = 1,   // [-2V,  +2V]
        VoltGain5X  = 2,   // [-5V,  +5V]
        VoltGain10X = 3,   // [-10V, +10V]
        NumVoltGain = 4
    };

    const String VoltGainStringArray[NumVoltGain] =
    {
        String("VoltGain1X"),
        String("VoltGain2X"),
        String("VoltGain5X"),
        String("VoltGain10X")
    };

#elif defined VOLTAGE_VARIANT_AD8251

    enum VoltGain // Analog output voltage scaling factor
    {
        VoltGain1X  = 0,   // [-1V,  +1V]
        VoltGain2X  = 1,   // [-2V,  +2V]
        VoltGain4X  = 2,   // [-4V,  +4V]
        VoltGain8X  = 3,   // [-8V,  +8V]
        VoltGain10X = 4,   // [-10V, +10V] // used for reference input
        NumVoltGain = 5
    };

    const String VoltGainStringArray[NumVoltGain] =
    {
        String("VoltGain1X"),
        String("VoltGain2X"),
        String("VoltGain4X"),
        String("VoltGain8X"),
        String("VoltGain10X")
    };

#else
#   error "VOLTAGE_VARIANT must be specified"
#endif

    enum CurrGainPath // TransImpedance Amplifiler Current gain path
```

```
    {
        CurrGainPathIn1 = 0,  // [-1uA,    +1uA]    w/ default resistors
        CurrGainPathIn2 = 1,  // [-10uA,   +10uA]
        CurrGainPathIn3 = 2,  // [-100uA,  +100uA]
        CurrGainPathIn4 = 3,  // [-1000uA, +1000uA]
        CurrGainPathErr = 4,  // Incorrect path setting
        NumCurrGainPath = 5
    };


    const String CurrGainPathStringArray[NumCurrGainPath] =
    {
        String("CurrGainPathIn1"),
        String("CurrGainPathIn2"),
        String("CurrGainPathIn3"),
        String("CurrGainPathIn4"),
        String("CurrGainPathErr")
    };

    String voltGainToString(VoltGain value);

    String currGainPathToString(CurrGainPath value);

}
#endif
```

методы для строковой презентации коэффициентов усиления. Используется в аналоговой подсистеме.

ps_multistep_test

```
#ifndef PS_MULTISTEP_TEST_H
#define PS_MULTISTEP_TEST_H
#include "ps_base_test.h"
#include "ps_constants.h"
#include "ps_time_utils.h"
#include "third-party/Array/Array.h"
#include "third-party/ArduinoJson/ArduinoJson.h"

namespace ps
{

    template<size_t MAX_SIZE>
    class MultiStepTest : public BaseTest
    {

        public:

            //static const String StepArrayKey;

            MultiStepTest(size_t numStep=5);
```

```
            virtual void setStepValue(size_t n, float value);
            virtual float getStepValue(size_t n) const;

            virtual void setStepDuration(size_t n, uint64_t duration);
            virtual uint64_t getStepDuration(size_t n) const;
            virtual uint64_t getDuration() const;

            virtual void setNumStep(size_t numStep);
            virtual size_t getNumStep() const;
            virtual size_t getMaxNumStep() const;

            virtual bool isDone(uint64_t t) const override;
            virtual uint64_t getDoneTime() const override;
            virtual void reset() override;

            virtual float getValue(uint64_t t) const override;
            virtual float getMaxValue() const override;
            virtual float getMinValue() const override;

            virtual void getParam(JsonObject &jsonDat) override;
            virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDa
t) override;

        protected:

            Array<float, MAX_SIZE> valueArray_;
            Array<uint64_t, MAX_SIZE> durationArray_;
            size_t numStep_;

            void setValueAndDurationFromJson(JsonObject &jsonMsgPrm, JsonObject &
jsonDatPrm, ReturnStatus &status);

    };
```

тест, где каждое изменение напряжения задается шагом.

Параметры:

Параметр	Ед. изм	Тип данных
время молчания	мс	uint_64
напряжение молчания	В	float
шаг* - (продолжительность, напряжение)	(мс, В)	(uint_64, float)

*макс кол-во шагов = 50

Методы:

Для пользователя:

MultiStepTest(size_t numStep=5) - конструктор, в котором имя теста ставят multiStep, готовят массивы значений и продолжительности, устанавливают количество шагов (по умолчанию 5) и включают совместимость с мультиплексором.

Установить или получить значение в шаге

- virtual void setStepValue(size_t n, float value)
- virtual float getStepValue(size_t n) const

Установить или получить продолжительность в шаге

- virtual void setStepDuration(size_t n, uint64_t duration);
- virtual uint64_t getStepDuration(size_t n) const;

Получить продолжительность теста

- virtual uint64_t getDuration() const;

Установить или получить кол-во шагов

- virtual void setNumStep(size_t numStep);
- virtual size_t getNumStep() const;

Получить максимально возможное кол-во шагов

- virtual size_t getMaxNumStep() const;

Готов ли тест и за сколько

- virtual bool isDone(uint64_t t) const override; - готов ли тест
- virtual uint64_t getDoneTime() const override; - время исполнения теста

Сброс

- virtual void reset() override

Получить текущее значение

- virtual float getValue(uint64_t t) const override

Максимальное и минимальное значение

- virtual float getMaxValue() const override;
- virtual float getMinValue() const override;

JSON

- virtual void getParam(JsonObject &jsonDat) override - получить параметры в JSON файл
- virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDat) override - установить параметры из JSON файла - value and duration

Внутренние:

Установить значение и продолжительность из JSON файла.

Обработаны след. исключения:

- для кол-ва шагов:
  - не найдено кол-во шагов;
  - на вход подан не массив;
  - массив слишком большой
- для value and duration:
  - продолжительность не int;
  - значение не float;
  - размер массива не 2;
  - на вход подан не массив.

1. void setValueAndDurationFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);

ps_periodic_test

```
#ifndef PS_PERIODIC_TEST_H
#define PS_PERIODIC_TEST_H

#include "ps_base_test.h"
#include "ps_constants.h"

namespace ps
{

    class PeriodicTest : public BaseTest
    {
        public:

            static constexpr float DefaultAmplitude = 1.0;
            static constexpr float DefaultOffset = 0.0;
            static constexpr float DefaultShift = 0.0;
            static constexpr uint64_t DefaultPeriod = UINT64_C(1000000);
```

```
            static constexpr uint32_t DefaultNumCycles = UINT32_C(10);

            PeriodicTest();

            virtual void setAmplitude(float amplitude);
            virtual float getAmplitude() const;

            virtual void setOffset(float offset);
            virtual float getOffset() const;

            virtual void setPeriod(uint64_t period);
            virtual uint64_t getPeriod() const;

            virtual void setNumCycles(uint32_t numCycles);
            virtual uint32_t getNumCycles() const;

            virtual void setShift(float lag);
            virtual float getShift() const;

            virtual uint32_t getCycleCount(uint64_t t) const;

            virtual bool isDone(uint64_t t) const override;
            virtual uint64_t getDoneTime() const override;
            virtual float getValue(uint64_t t) const override;
            virtual float getMaxValue() const override;
            virtual float getMinValue() const override;
            virtual void getParam(JsonObject &jsonDat) override;
            virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDa
t) override;

        protected:

            float amplitude_ = DefaultAmplitude;     // 12-bit Dac int
            float offset_ = DefaultOffset;           // 12-bit Dac int
            uint64_t period_ = DefaultPeriod;        // Waveform period (us)
            uint32_t numCycles_ = DefaultNumCycles;  // Number of cycles to perfo
rm

            float shift_ = DefaultShift;             // Waveform shift as fractio
n of period [0,1]
            uint64_t shiftInUs_ = 0;                 // Waveform shift in us;

            void setAmplitudeFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDat
Prm, ReturnStatus &status);
            void setOffsetFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm
, ReturnStatus &status);
            void setPeriodFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm
, ReturnStatus &status);
            void setNumCyclesFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDat
Prm, ReturnStatus &status);
```

```
            void setShiftFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm,
 ReturnStatus &status);

            void updateShiftInUs();




    };


}

#endif
```

Параметры:

Параметр	Ед. изм	Тип данных
время молчания	мс	uint_64
напряжение молчания	В	float
амплитуда	В	float
смещение по y	В	float
период	мс	uint_64
кол-во циклов	шт.	uint_32
фазовый сдвиг (смещение по x)		float

Методы

Для пользователя:

PeriodicTest::PeriodicTest() - конструктор, в котором имя теста = periodic и рассчитывается перемещение в секундах.

Установить или получить амплитуду

- virtual void setAmplitude(float amplitude);
- virtual float getAmplitude() const;

Установить или получить смещение (относительно оси ординат)

- virtual void setOffset(float offset);
- virtual float getOffset() const;

Установить или получить период

- virtual void setPeriod(uint64_t period);
- virtual uint64_t getPeriod() const;

Установить или получить кол-во итераций

- virtual void setNumCycles(uint32_t numCycles);
- virtual uint32_t getNumCycles() const;

Установить или получить смещение относительно оси абсцисс

- virtual void setShift(float lag);
- virtual float getShift() const;

Текущая итерация

- virtual uint32_t getCycleCount(uint64_t t) const;

Готов ли тест и за сколько

- virtual bool isDone(uint64_t t) const override
- virtual uint64_t getDoneTime() const override
- virtual float getValue(uint64_t t) const override; - получение текущего значения. Создан для обязательной инициализации в классах-наследниках.

Мин и Макс знач. напряжения

- virtual float getMaxValue() const override;
- virtual float getMinValue() const override;

Скинуть параметры в JSON файл

- virtual void getParam(JsonObject &jsonDat) override;

Установить параметры из JSON файла

- virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDat) override;

Внутренние:

Установить амплитуду, смещение, период, кол-во циклов, перемещение из JSON файла. Прописаны исключения на тип и для перемещения на значение [между 0 и 1]

1. void setAmplitudeFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);
2. void setOffsetFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);
3. void setPeriodFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);
4. void setNumCyclesFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);
5. void setShiftFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);
6. void updateShiftInUs() - смещение по x в секундах

ps_cyclic_test.h

```
#ifndef PS_CYCLIC_TEST_H
#define PS_CYCLIC_TEST_H

#include <Arduino.h>
#include "ps_periodic_test.h"

namespace ps
{

    class CyclicTest : public PeriodicTest
    {
        public:

            CyclicTest();
            virtual float getValue(uint64_t t) const override;

    };

} // namespace ps

#endif
```

Наследник periodic_test. Тест, где потенциал между рабочим и опорным электродами циклически наращивается вверх и вниз кусочно-линейным способом - в треугольной форме волны.

Параметры:

Параметр	Ед. изм	Тип данных
время молчания	мс	uint_64
напряжение молчания	В	float

амплитуда	В	float
смещение по y	В	float
период	мс	uint_64
кол-во циклов	шт.	uint_32
фазовый сдвиг (смещение по x)		float

Методы
Для пользователя:

CyclicTest() - - конструктор, в котором имя теста = cyclic и multiplexer вкл.

virtual float getValue(uint64_t t) const override - получение текущего значения.

ps_pin_map

```
#ifndef PS_PIN_MAP_H
#define PS_PIN_MAP_H
#include <Arduino.h>
#include "ps_hardware_defs.h"

namespace ps
{

#if defined HARDWARE_VERSION_0P1
    const int AD8250_GAIN_A0 = 0;
    const int AD8250_GAIN_A1 = 1;
    const int TIA_SW1_IN1 = 2;
    const int TIA_SW1_IN2 = 5;
    const int TIA_SW1_IN3 = 6;
    const int TIA_SW1_IN4 = 7;
    const int TIA_SW2_IN1 = 8;
    const int TIA_SW2_IN2 = 9;
    const int TIA_SW2_IN3 = 22;
    const int TIA_SW2_IN4 = 23;
#elif defined HARDWARE_VERSION_0P2
    const int DAC_GAIN_A0 = 0;
    const int DAC_GAIN_A1 = 1;
    const int TIA_GAIN_A0 = 5;
    const int TIA_GAIN_A1 = 6;
    const int REF_GAIN_A0 = 20;
    const int REF_GAIN_A1 = 21;
```

```
    const int SW_CTR_ELECT = 7;
    const int SW_REF_ELECT = 8;
    const int SW_WRK_ELECT = 9;
#else
#   error "HARDWARE_VERSION must be specified"
#endif
    const int DAC_UNI_PIN = A14;
    const int TIA_OUT_UNI_PIN = A1;
    const int REF_ELECT_UNI_PIN = A2;

    // Multiplexer switch pins
    const int MUX_WRK1_TO_TIA = 24;
    const int MUX_WRK1_TO_GND = 28;
    const int MUX_WRK2_TO_TIA = 25;
    const int MUX_WRK2_TO_GND = 29;
    const int MUX_WRK3_TO_TIA = 26;
    const int MUX_WRK3_TO_GND = 30;
    const int MUX_WRK4_TO_TIA = 27;
    const int MUX_WRK4_TO_GND = 31;
    const int MUX_WRK5_TO_TIA = 4;
    const int MUX_WRK5_TO_GND = 11;
    const int MUX_WRK6_TO_TIA = 19;
    const int MUX_WRK6_TO_GND = 13;
    const int MUX_WRK7_TO_TIA = 18;
    const int MUX_WRK7_TO_GND = 10;
    const int MUX_CTR_CONN = 3;
    const int MUX_REF_CONN = 12;

}

#endif
```

нумерование пинов для изменения их состояния в коде.

ps_hardware_defs

```
//защита от повторного включения файла
#ifndef PS_HARDWARE_DEFS_H
#define PS_HARDWARE_DEFS_H

    #if 1
        // Hardware version 0.2
        // ------------------------------------------------------------------------
-
        #define HARDWARE_VERSION_0P2
        #define VOLTAGE_VARIANT_10V
        #define CURRENT_VARIANT_MICRO_AMP   // select from (NANO, MICRO, MILL)
    #else
        // Hardware version 0.1
```

```
        // ------------------------------------------------------------------------
-
        #define HARDWARE_VERSION_0P1
        #define VOLTAGE_VARIANT_AD8250      // select from (AD8250 or AD8251)
        #define CURRENT_VARIANT_MICRO_AMP   // select from (NANO, MICRO or MILL)
    #endif

#endif
```

Описание файла: Прописана защита от повторного включения файла и версия hardware(оборудование)

ps_keyvalue_command

```
#ifndef PS_KEYVALUE_COMMAND_H
#define PS_KEYVALUE_COMMAND_H

#include "ps_constants.h"

namespace ps
{

    template<typename T>
    class KeyValueCommand
    {
        public:
            KeyValueCommand() {};
            KeyValueCommand(String key, String value, ReturnStatus (T::*method)(J
sonObject&,JsonObject&));

            String key();
            void setKey(String key);

            String value();
            void setValue(String value);

            void setMethod(ReturnStatus (T::*method)(JsonObject&,JsonObject&));
            ReturnStatus applyMethod(T* client, JsonObject &jsonMsg, JsonObject &
jsonDat);

        protected:
            String key_;
            String value_;
            ReturnStatus (T::*method_)(JsonObject&,JsonObject&) = nullptr;
    };
```

Описание файла: класс,через который происходит ввод ключевых команд в Командую строку(Command Table).
Имеет три атрибута:

String key_;

String value_;

ReturnStatus (T::*method_)(JsonObject&,JsonObject&) = nullptr;

Для всех них прописаны геттеры и сеттеры.

ps_linearsweep_test

```
#ifndef PS_LINEARSWEEP_TEST_H
#define PS_LINEARSWEEP_TEST_H

#include "ps_base_test.h"

namespace ps
{

    class LinearSweepTest : public BaseTest
    {

        public:

            LinearSweepTest();

            void setStartValue(float value);
            float getStartValue() const;

            void setFinalValue(float value);
            float getFinalValue() const;

            void setDuration(uint64_t duration);
            uint64_t getDuration() const;
            virtual bool isDone(uint64_t t) const override;
            virtual uint64_t getDoneTime() const override;
            virtual float getValue(uint64_t t) const override;
            virtual float getMaxValue() const override;
            virtual float getMinValue() const override;
            virtual void getParam(JsonObject &jsonDat) override;
            virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDa
t) override;
        protected:
            float startValue_ = -0.5;
            float finalValue_ =  0.5;
            uint64_t duration_ = 2000000;

            void setStartValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDa
tPrm, ReturnStatus &status);
            void setFinalValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDa
tPrm, ReturnStatus &status);
```

```
            void setDurationFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatP
rm, ReturnStatus &status);

    };

} // namespace ps

#endif
```

Тест линейной развертки

Параметры:

Параметр	Ед. изм	Тип данных
Стартовое значение	В.	float
Финальное значение	В.	float
продолжительность теста	Сек.	uint_64

Методы:

Для пользователя:

Установить или получить стартового значения

- void setStartValue(float value);
- float getStartValue() const;

Установить или получить конечного значения

- void setFinalValue(float value);
- float getFinalValue() const;

Получить или установить продолжительность теста

- void setDuration(uint64_t duration);
- uint64_t getDuration() const;

Установить или получить кол-во шагов

- virtual void setNumStep(size_t numStep);
- virtual size_t getNumStep() const;

Получить максимально возможное кол-во шагов

- virtual size_t getMaxNumStep() const;

Готов ли тест и за сколько

- virtual bool isDone(uint64_t t) const override; - готов ли тест
- virtual uint64_t getDoneTime() const override; - время исполнения теста

Получить текущее значение

- virtual float getValue(uint64_t t) const override

Максимальное и минимальное значение

- virtual float getMaxValue() const override;
- virtual float getMinValue() const override;

JSON

- virtual void getParam(JsonObject &jsonDat) override - получить параметры в JSON файл
- virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDat) override - установить параметры из JSON файла - value and duration

Внутренние:

Установить значение и продолжительность из JSON файла.

void setStartValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);

void setFinalValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);

void setDurationFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status);

Обработаны след. исключения:

- для кол-ва шагов:
  - не найдено кол-во шагов;
  - на вход подан не массив;
  - массив слишком большой
- для value and duration:
  - продолжительность не int;
  - значение не float;

- размер массива не 2;
- на вход подан не массив.

ps_lookup_table

```
#ifndef PS_LOOKUP_TABLE_H
#define PS_LOOKUP_TABLE_H

#include "third-party/Array/Array.h"

namespace ps
{

    template<typename T, size_t SIZE>
    class LookupTable
    {
        public:

            LookupTable();
            T& operator[](const size_t ind);
            T  operator[](const size_t ind) const;
            T getValue(T pos) const;  // pos >= 0  and pos < SIZE - 1

        protected:

            Array<T,SIZE> data_;
    };
```

Описание файла: Файл, в котором объявлен класс LookupTable. Этот класс используется для того чтобы просматривать данные класса Array. В нём есть перегруженный оператор [] для удобного обращения, так же это можно сделать методом getValue, передав туда индекс.

ps_message_parser

```
#ifndef PS_MESSAGE_PARSER_H
#define PS_MESSAGE_PARSER_H
#include <Arduino.h>
#include "ps_constants.h"
#include "third-party/ArduinoJson/ArduinoJson.h"

namespace ps
{
    class MessageParser
    {
```

```
        public:
            MessageParser();
            JsonObject& parse(String &message, StaticJsonBuffer<JsonMessageBuffer
Size> &jsonBuffer);
    };
} // namespace ps

#endif
```

Описание файла: переводит сообщение типа String в JSON объект

ps_message_receiver

```
#ifndef PS_MESSAGE_RECEIVER_H
#define PS_MESSAGE_RECEIVER_H

#include <Arduino.h>
#include "ps_constants.h"
#include "ps_circular_buffer.h"

namespace ps
{

    class MessageReceiver
    {

        public:

            MessageReceiver();
            void reset();
            void readData();
            String next();
            bool available() const;
            uint32_t getMessageCnt() const;
            uint32_t getTotalMessageCnt() const;

        protected:

            CircularBuffer<char,SerialBufferSize> serialBuffer_;
            bool overflow_ = false;
            uint32_t messageCnt_ = 0;
            uint32_t totalMessageCnt_ = 0;


    };

} // namespace ps

#endif
```

Описание файла:
Класс Message receiver отвечает за принятие сообщения. Состоит из ранее упомянутого циркулярного буфера, флага заполнения и счетчика принятых сообщений без переполнения и счётчика всех принятых сообщений.

В нем есть след методы:

void reset(); - обнуляет кол-во принятых сообщений без переполнения и снимает флаг переполнения

void readData(); - посимвольно принимает информацию

String next(); - копирует данные из буфера в строку

bool available() const; - проверка доступности

uint32_t getMessageCnt() const; - возвращает счётчик принятых сообщений без переполнения

uint32_t getTotalMessageCnt() const; - возвращает счётчик всех принятых сообщений.

ps_message_sender

```
#ifndef PS_MESSAGE_SENDER_H
#define PS_MESSAGE_SENDER_H

#include <Arduino.h>
#include "ps_constants.h"
#include "ps_sample.h"
#include "ps_return_status.h"

#include "third-party/ArduinoJson/ArduinoJson.h"

namespace ps
{

    class MessageSender
    {

        public:

            MessageSender();
            void sendSample(Sample sample);
            void sendSampleEnd();
            void sendCommandResponse(ReturnStatus status, JsonObject &jsonDat);

    };
```

```
} // namespace ps


#endif
```

Описание файла: Класс Message sender отвечает за запись данных в JSON объект.

В нем есть след методы:

void sendSample(Sample sample); - данные результата теста для отправки

void sendCommandResponse(ReturnStatus status, JsonObject &jsonDat); -

ps_multiplexer

```
#ifndef PS_MULTIPLEXER_H
#define PS_MULTIPLEXER_H

#include "ps_pin_map.h"
#include "ps_constants.h"
#include "ps_return_status.h"
#include "third-party/Array/Array.h"

namespace ps
{

    class Multiplexer
    {
        public:

            static const int MuxSwitchPin[NumMuxPin];
            static const int MuxToTiaPin[NumMuxChan];
            static const int MuxToGndPin[NumMuxChan];
            static const int NotConnected = -1;

            Multiplexer();

            void setupSwitchPins();
            void clearSwitchPins();

            void connectCtrElect();
            void disconnectCtrElect();

            void connectRefElect();
            void disconnectRefElect();
```

```
        void connectWrkElect(int electNum);
        void disconnectWrkElect();

        int currentWrkElect();
        bool isConnectedWrk();
        bool isConnectedCtr();
        bool isConnectedRef();

        void connectFirstEnabledWrkElect();
        void connectNextEnabledWrkElect();

        void start();
        void stop();
        bool isRunning();

        void enableWrkElect(int electNum);
        void disableWrkElect(int electNum);

        void enableAllWrkElect();
        void disableAllWrkElect();

        void setEnabledWrkElect(Array<int,NumMuxChan> enabledArray);
        Array<int,NumMuxChan> getEnabledWrkElect();
        bool isWrkElectEnabled(int electNum);

        int numEnabledWrkElect();
        int electNumToIndex(int electNum);
        int indexToElectNum(int index);

    protected:

        bool running_ = false;
        volatile int currWrkElect_ = NotConnected;

        int numEnabled_ = 0;
        Array<bool,NumMuxChan> enabledTable_;

        void setAllChanToGnd();
        void initializeEnabledTable(bool value);

        int countNumEnabled();

};
```

Описание файла: Реализация мультиплексора.

Мультипле́ксор — устройство, им0еющее несколько сигнальных входов, один или более управляющих входов и один выход. Мультиплексор позволяет передавать сигнал с одного из нескольких входов на один выход; при этом выбор желаемого входа осуществляется подачей соответствующей

комбинации управляющих сигналов.

Методы:

Публичные:

void setupSwitchPins(); - установка контактов переключателя мультиплексора

void clearSwitchPins(); - очистка контактов переключателя мультиплексора

void connectCtrElect(); - подключение вспомогательного электрода к трансимпедантному усилителю

void disconnectCtrElect(); - отключение вспомогательного электрода от трансимпедансного усилителя

void connectRefElect(); - подключение сравнительного(референсный) электрода к трансимпедантному усилителю

void disconnectRefElect(); - отключение сравнительного(референсный) электрода от трансимпедансного усилителя

void connectWrkElect(int electNum); - Подключение рабочего электрода к трансимпедантному усилителю

void disconnectWrkElect(); - Отсоединение рабочего электрода от трансимпедансного усилителя

int currentWrkElect(); - возвращает текущий работающий электрод

bool isConnectedWrk(); - подключён ли работающий электрод

bool isConnectedCtr(); - подключён ли вспомогательный электрод

bool isConnectedRef(); - подключён ли сравнительный(референсный) электрод

void connectFirstEnabledWrkElect(); - подключение первого включенного рабочего электрода

void connectNextEnabledWrkElect(); - подключение следующего(после первого) включенного рабочего электрода

void start(); - включение мультиплексора

void stop(); - выключение мультиплексора

bool isRunning(); - работает ли мультиплексор?

void enableWrkElect(int electNum); - включение рабочего электода по номеру электрода

void disableWrkElect(int electNum); - выключение рабочего электорда по номеру электрода

void enableAllWrkElect(); - включение всех рабочих электродов

void disableAllWrkElect(); - выключение всех рабочих электородов

void setEnabledWrkElect(Array<int,NumMuxChan> enabledArray); - включение всех рабочих электродов из массива

Array<int,NumMuxChan> getEnabledWrkElect(); - возвращает массив включенных рабочих электродов

bool isWrkElectEnabled(int electNum); - включён ли рабочий электрод

int numEnabledWrkElect(); - кол-во включенных рабочих электродов

int electNumToIndex(int electNum); - перевод номера электрода в индекс для массива

int indexToElectNum(int index); - перевод индекса в номер электрода

Защищённые:

void setAllChanToGnd(); - установка всех каналов к общему проводу

void initializeEnabledTable(bool value); - иницализация таблицы включённых электродов

int countNumEnabled(); - подсчёт включенных электродов

ps_range.h

```
#ifndef PS_RANGE_H
#define PS_RANGE_H

namespace ps
{

    template<typename GainType, typename IntType>
    class Range
```

```
    {
        public:

            Range() {};
            Range(String name, float minValue, float maxValue, GainType gain, Int
Type maxInt);
            String name() const;
            float minValue() const;
            float maxValue() const;
            GainType gain() const;
            IntType maxInt() const;
            inline IntType valueToInt(float volt) const;
            inline float intToValue(IntType value) const;

        private:

            String name_;
            float minValue_;
            float maxValue_;
            GainType gain_;
            IntType maxInt_;

    };
```

Модуль описывает шаблон класса Range и шаблоны функций:

name() - возвращает имя;

minValue() - возвращает минимальное значение;

maxValue() - возвращает максимальное значение;

gain() - возвращает коэффициент усиления;

maxInt() - возвращает maxInt;

valueToInt() - приводит передаваемое значение volt к int типу + возвращает значение ограниченное 0≤value≤maxInt_:

IntType value = IntType(float(maxInt_)/(maxValue_ - minValue_)*(maxValue_ - volt));

intToValue() - обратная по отношению к valueToInt операция.

ps_return_status

```
#ifndef PS_RETURN_STATUS_H
#define PS_RETURN_STATUS_H
#include <Arduino.h>
namespace ps
```

```
{
    class ReturnStatus
    {

        public:

            ReturnStatus() { };
            void appendToMessage(String value);

            bool success = true;;
            String message;

    };
} // namespace ps
#endif
```

сохраняет статус записывая в переменную message статусы тестов через запятую.

ps_sample.h

```
#ifndef PS_SAMPLE_H
#define PS_SAMPLE_H

namespace ps
{

    class Sample
    {
        public:

            uint64_t t;
            float volt;
            float curr;
            uint8_t chan;
    };

}

#endif
```

ps_sinusoid_test

```
#ifndef PS_SINUSOID_TEST_H
#define PS_SINUSOID_TEST_H

#include "ps_periodic_test.h"
#include "ps_lookup_table.h"
#include <Arduino.h>

namespace ps
```

```
{

    class SinusoidTest : public PeriodicTest
    {
        public:

            static constexpr uint32_t LookupTableSize = 300;

            SinusoidTest();
            virtual void setAmplitude(float amplitude) override;
            virtual void setOffset(float offset) override;
            virtual float getValue(uint64_t t) const override;
            void updateLookupTable();

        protected:

            LookupTable<float,LookupTableSize> lookupTable_;


    };

} // namespace ps

#endif
```

Наследник periodic_test. Этот тест проверяет, является ли ток в потенциостате синусоидальным.

Параметр	Ед. изм	Тип данных
время молчания	мс	uint_64
напряжение молчания	В	float
амплитуда	В	float
смещение по y	В	float
период	мс	uint_64
кол-во циклов	шт.	uint_32
фазовый сдвиг (смещение по x)	В	float

Методы

SinusoidTest(); - конструктор, в котором устанавливается название теста (sinusoid) и включается поддержка мультиплексера.

virtual void setAmplitude(float amplitude) override; - сеттер(установка значения) для амплитуды

virtual void setOffset(float offset) override; - сеттер(установка значения) для смещения по y

virtual float getValue(uint64_t t) const override; - получение значения в период времени

ps_squarewave_test

Тест прямоугольной вольтамперометрии.

```
#ifndef PS_SQUAREWAVE_TEST_H
#define PS_SQUAREWAVE_TEST_H
#include "ps_base_test.h"

namespace ps
{
    class SquareWaveTest : public BaseTest
    {
        public:

            SquareWaveTest();

            void setStartValue(float value);
            float getStartValue();

            void setFinalValue(float value);
            float getFinalValue();

            void setStepValue(float value);
            float getStepValue();

            void setAmplitude(float value);
            float getAmplitude();

            void setWindow(float value);
            float getWindow();

            virtual bool isDone(uint64_t t) const override;
            virtual uint64_t getDoneTime() const override;
            virtual void reset();

            virtual float getValue(uint64_t t) const override;
            virtual float getStairValue(uint64_t t) const;

            virtual float getMaxValue() const override;
```

```
            virtual float getMinValue() const override;

            virtual void setSamplePeriod(uint64_t samplePeriod) override;

            virtual bool updateSample(Sample sampleRaw, Sample &sampleTest);

            virtual void getParam(JsonObject &jsonDat) override;
            virtual ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDa
t) override;

        protected:

            float startValue_ = -0.5;
            float finalValue_ = 0.5;
            float stepValue_ = 0.005;
            float amplitude_ = 0.025;
            float window_ = 0.2;

            float maxValue_ = 0.0;
            float minValue_ = 0.0;
            float stepSign_ = 1.0;

            uint64_t doneTime_ = 0;
            uint64_t halfSamplePeriod_ = 0;
            uint64_t windowLenUs_ = 0;

            bool isFirst_ = true;
            uint64_t testCnt_ = 0;

            uint64_t numForward_ = 0;
            uint64_t numReverse_ = 0;

            float currForward_ = 0.0;
            float currReverse_ = 0.0;

            void updateDoneTime();
            void updateMaxMinValues();
            void updateWindowLenUs();
            void updateStepSign();

            void setStartValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDa
tPrm, ReturnStatus &status);
            void setFinalValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDa
tPrm, ReturnStatus &status);
            void setStepValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDat
Prm, ReturnStatus &status);
            void setAmplitudeFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDat
Prm, ReturnStatus &status);
            void setWindowFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm
, ReturnStatus &status);
    };
```

```
} // namespace ps

#endif
```

Параметры:

Параметр	Ед. изм	Тип данных
Стартовое значение	В.	float
Финальное значение	В.	float
Шаг	(мс, В)	float
Амплитуда	В.	float
Окно	%	float
Продолжительность теста	Сек.	uint_64

Методы:

SquareWaveTest() - конструктор, в котором устанавливается название теста (squareWave) и отключается поддержка мультиплексера.

setStartValue(float value) - устанавливает стартовое значение;

getStartValue() - получает стартовое значение;

setFinalValue(float value), getFinalValue() - установить/получить финальное значение;

setStepValue(float value), getStepValue() - установить/получить значение шага;

setAmplitude(float value), getAmplitude() - установить/получить амплитуду;

setWindow(float value), getWindow() - установить/получить окно;

getMaxValue() const, getMinValue() const - получить максимальное/минимальное значение;

setSamplePeriod(uint64_t SamplePeriod) - установить период выборки;

isDone(uint64_t t) const - провера выполненности теста;

getDoneTime() const - получить время выполнения;

reset() - сброс всех значений к нулю;

getValue(uint64_t t) const - получение значений;

getStairValue(uint64_t t) const - получение значения ступени;

updateSample(Sample sampleRaw, Sample &sampleTest) - обновить образец;

getParam(JsonObject &jsonDat), setParam(JsonObject &jsonMsg, JsonObject &jsonDat) - получить/установить значения из json файла;

updateDoneTime() - время обновления данных;

updateMaxMinValues() - обновить минимальное и максимальное значения;

updateWindowLenUs() - обновление значения окна;

updateStepSign() - изменяет знак шага(+/-);

setStartValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status), setFinalValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status) - устанавливает начальное и финальное значение из json файла;

setStepValueFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status) - устанавливает значение шага из json файла;

setAmplitudeFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status) - устанавливает амплитуду из json файла;

setWindowFromJson(JsonObject &jsonMsgPrm, JsonObject &jsonDatPrm, ReturnStatus &status) - устанавливает окно из json файла;

ps_system_state.cpp

```
#ifndef PS_SYSTEM_STATE_H
#define PS_SYSTEM_STATE_H

#include <Arduino.h>
#include "ps_hardware_defs.h"
#include "ps_constants.h"
#include "ps_return_status.h"
#include "ps_analog_subsystem.h"
#include "ps_electrode_switch.h"
#include "ps_message_receiver.h"
#include "ps_message_sender.h"
#include "ps_message_parser.h"
#include "ps_command_table.h"
```

```
#include "ps_circular_buffer.h"
#include "ps_voltammetry.h"
#include "ps_sample.h"
#include "ps_filter.h"
#include "ps_multiplexer.h"
#include "third-party/Array/Array.h"
#define ARDUINOJSON_USE_DOUBLE 0
#include "third-party/ArduinoJson/ArduinoJson.h"

namespace ps
{

    class SystemState
    {

        public:

            SystemState();
            void initialize();

            void processMessages();
            void updateMessageData();
            void serviceDataBuffer();

            ReturnStatus onCommandRunTest(JsonObject &jsonMsg, JsonObject &jsonDa
t);
            ReturnStatus onCommandStopTest(JsonObject &jsonMsg, JsonObject &jsonD
at);
            ReturnStatus onCommandGetVolt(JsonObject &jsonMsg, JsonObject &jsonDa
t);
            ReturnStatus onCommandSetVolt(JsonObject &jsonMsg, JsonObject &jsonDa
t);
            ReturnStatus onCommandGetCurr(JsonObject &jsonMsg, JsonObject &jsonDa
t);
            ReturnStatus onCommandGetRefVolt(JsonObject &jsonMsg, JsonObject &jso
nDat);
            ReturnStatus onCommandSetTestParam(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandGetTestParam(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandSetVoltRange(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandGetVoltRange(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandSetCurrRange(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandGetCurrRange(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandSetDeviceId(JsonObject &jsonMsg, JsonObject &js
onDat);
```

```
            ReturnStatus onCommandGetDeviceId(JsonObject &jsonMsg, JsonObject &js
onDat);
            ReturnStatus onCommandSetSamplePeriod(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetSamplePeriod(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetTestDoneTime(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetTestNames(JsonObject &jsonMsg, JsonObject &j
sonDat);
            ReturnStatus onCommandGetVersion(JsonObject &jsonMsg, JsonObject &jso
nDat);
            ReturnStatus onCommandGetVariant(JsonObject &jsonMsg, JsonObject &jso
nDat);
            ReturnStatus onCommandSetMuxEnabled(JsonObject &jsonMsg, JsonObject &
jsonDat);
            ReturnStatus onCommandGetMuxEnabled(JsonObject &jsonMsg, JsonObject &
jsonDat);
            ReturnStatus onCommandSetEnabledMuxChan(JsonObject &jsonMsg, JsonObje
ct &jsonDat);
            ReturnStatus onCommandGetEnabledMuxChan(JsonObject &jsonMsg, JsonObje
ct &jsonDat);
            ReturnStatus onCommandGetMuxTestNames(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandSetMuxRefElectConn(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
            ReturnStatus onCommandGetMuxRefElectConn(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
            ReturnStatus onCommandSetMuxCtrElectConn(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
            ReturnStatus onCommandGetMuxCtrElectConn(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
            ReturnStatus onCommandSetMuxWrkElectConn(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
            ReturnStatus onCommandGetMuxWrkElectConn(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
            ReturnStatus onCommandDisconnAllMuxElect(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
#if defined HARDWARE_VERSION_0P2
            ReturnStatus onCommandSetRefElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetRefElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandSetCtrElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetCtrElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandSetWrkElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetWrkElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
```

```
            ReturnStatus onCommandSetAllElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandGetAllElectConn(JsonObject &jsonMsg, JsonObject
 &jsonDat);
            ReturnStatus onCommandSetElectAutoConn(JsonObject &jsonMsg, JsonObjec
t &jsonDat);
            ReturnStatus onCommandGetElectAutoConn(JsonObject &jsonMsg, JsonObjec
t &jsonDat);
            ReturnStatus onCommandSetRefElectVoltRange(JsonObject &jsonMsg, JsonO
bject &jsonDat);
            ReturnStatus onCommandGetRefElectVoltRange(JsonObject &jsonMsg, JsonO
bject &jsonDat);
            ReturnStatus onCommandGetHardwareVersion(JsonObject &jsonMsg, JsonObj
ect &jsonDat);
#endif
            void startTest();
            void stopTest();

            void setSamplePeriod(uint32_t samplePeriod);
            uint32_t getSamplePeriod();

            void setTestTimerCallback(void(*func)());
            void updateTestOnTimer();
        protected:

            volatile bool testInProgress_;
            volatile bool lastSampleFlag_;
            AnalogSubsystem analogSubsystem_;
            Multiplexer multiplexer_;
#if defined HARDWARE_VERSION_0P2
            ElectrodeSwitch electrodeSwitch_;
            bool electrodeAutoConnect_;
#endif
            MessageReceiver messageReceiver_;
            MessageParser messageParser_;
            MessageSender messageSender_;
            CommandTable<SystemState,CommandTableMaxSize> commandTable_;

            CircularBuffer<Sample,DataBufferSize> dataBuffer_;
            Voltammetry voltammetry_;

            IntervalTimer testTimer_;
            void (*testTimerCallback_)() = dummyTimerCallback;
            volatile uint64_t timerCnt_;
            uint32_t samplePeriod_;
            uint32_t sampleModulus_;

            Array<LowPass,NumMuxChan> currLowPass_;
            float lowPassDtSec_;
            BaseTest *test_;
```

```
            static void dummyTimerCallback() {};
            void updateSampleModulus();
    };


} // namespace ps

#endif
```

Первым делом настраивает состояние системы по умолчанию – тестов нет, таймер не запущен, значения фильтра нижних частот тоже по умолчанию(частота среза – 200), период на образец – 0.1с.

Далее в командное табло регистрируются методы по формуле обращения «command <команда>» и устанавливается текущий клиент. Их много, позволяют настраивать все элементы описанные выше.

После включается аналоговая система с выходящим значением напряжения 0 и обнуляется циркулярный буфер приемщика сообщений.

Если версия устройства вторая, то производится автоматическая настройка электродов на вывод. Устанавливать состояние «присоединен» надо вручную в любом случае.

Методы system_state. Начнём с базовых, они доступны для любой версии железа.

Отдельно остановимся на тестах:

Запустить или остановить тест в безопасном режиме, т.к. входящие данные теста и если они неправильного синтаксиса, то тест не запустится. (это относится ко всем методам с set). Если мультиплексор включён, то тест должен его поддерживать, иначе теста не будет. Если на мультиплексоре заняты рабочие электроды, то тест также не запустится. При успешном прохождении проверок происходит подключение электродов к мультиплексору.

· ReturnStatus **onCommandRunTest**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandStopTest**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Установить или получить значение напряжения

· ReturnStatus **onCommandGetVolt**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandSetVolt**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Получить ток или получить напряжение на электроде сравнения

· ReturnStatus **onCommandGetCurr**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetRefVolt**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

·

· ReturnStatus **onCommandSetTestParam**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetTestParam**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Установить или получить промежуток изменения напряжения

· ReturnStatus **onCommandSetVoltRange**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetVoltRange**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Установить или получить промежуток изменения тока

· ReturnStatus **onCommandSetCurrRange**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetCurrRange**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Установить или получить код устройства

· ReturnStatus **onCommandSetDeviceId**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetDeviceId**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Установить или получить время между выборками с командной строки

· ReturnStatus **onCommandSetSamplePeriod**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetSamplePeriod**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Получить время выполнения теста, имена всех тестов, версию прошивки и вариант железа

· ReturnStatus **onCommandGetTestDoneTime**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetTestNames**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetVersion**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetVariant**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Вкл/выкл мультиплексор или проверить его состояние

· ReturnStatus **onCommandSetMuxEnabled**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetMuxEnabled**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Сделать доступным канал мультиплексора (тем самым в вкл. некоторое кол-во рабочих электродов. Кол-во зависит от канала) или проверить его состояние по индексу

· ReturnStatus **onCommandSetEnabledMuxChan**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetEnabledMuxChan**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Получить имена тестов на мультиплексоре

· ReturnStatus **onCommandGetMuxTestNames**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Изменить состояние «присоединён» или получить текущее состояние электрода сравнения, вспомогательного электрода или рабочего электрода на мультеплексоре.

· ReturnStatus **onCommandSetMuxRefElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetMuxRefElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandSetMuxCtrElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetMuxCtrElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandSetMuxWrkElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetMuxWrkElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Разъединить все электроды мультиплексора.

· ReturnStatus **onCommandDisconnAllMuxElect**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

Установить или получить время между выборками

void **setSamplePeriod**(uint32_t samplePeriod);

uint32_t **getSamplePeriod**();

Буфер для хранения образцов

void SystemState::**serviceDataBuffer**()

Чтение информации с буфера

void SystemState::**updateMessageData**()

Вывод сообщений во время обработки JSON

void SystemState::**processMessages**()

установить колбэк таймера

void **setTestTimerCallback**(void(*func)());

void **updateTestOnTimer**() - изменение значений теста во время его исполнения

void **startTest**() – начинает тест

void **stopTest**() – останавливает тест.

Версия железа 2

· Изменить состояние «присоединён» или получить текущее состояние электрода сравнения, вспомогательного электрода или рабочего электрода.

· ReturnStatus **onCommandSetRefElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetRefElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandSetCtrElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetCtrElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandSetWrkElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetWrkElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Все электроды в состояние «присоединён» или получить состояние всех

· ReturnStatus **onCommandSetAllElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetAllElectConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Автоматическое подключение всех электродов

· ReturnStatus **onCommandSetElectAutoConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetElectAutoConn**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Установить или получить промежуток изменения напряжения для электрода сравнения

· ReturnStatus **onCommandSetRefElectVoltRange**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· ReturnStatus **onCommandGetRefElectVoltRange**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

· Получить версию железа

· ReturnStatus **onCommandGetHardwareVersion**(JsonObject &*jsonMsg*, JsonObject &*jsonDat*);

ps_time_utils.h

```
#ifndef PS_TIME_UTILS_H
#define PS_TIME_UTILS_H

namespace ps
{

    inline uint32_t convertUsToMs(uint64_t t)
    {
        return uint32_t(t/UINT64_C(1000));
    }


    inline uint64_t convertMsToUs(uint32_t t)
    {
        return uint64_t(t)*UINT64_C(1000);
    }


} // namespace ps

#endif
```

Вспомогательные методы для преобразования секунд в миллисекунды и обратно

ps_volt_range

```
#ifndef PS_VOLT_RANGE_H
#define PS_VOLT_RANGE_H

#include <Arduino.h>
#include "ps_range.h"
#include "ps_gains.h"

namespace ps
{
    class VoltRange : public Range<VoltGain,uint16_t>
    {
        public:
            VoltRange() : Range<VoltGain,uint16_t>()  {};
```

```
            VoltRange(String name, float minValue, float maxValue, VoltGain voltG
ain, uint16_t maxInt)
                : Range<VoltGain,uint16_t>(name, minValue, maxValue, voltGain, ma
xInt) {};

    };

    class VoltRangeDac : public VoltRange
    {
        //using VoltRange::VoltRange;

        public:
            VoltRangeDac() : VoltRange()  {};

            VoltRangeDac(String name, float minValue, float maxValue, VoltGain vo
ltGain, uint16_t maxInt)
                : VoltRange(name, minValue, maxValue, voltGain, maxInt) {};
    };

    class VoltRangeAdc : public VoltRange
    {
        //using VoltRange::VoltRange;

        public:
            VoltRangeAdc() : VoltRange()  {};

            VoltRangeAdc(String name, float minValue, float maxValue, VoltGain vo
ltGain, uint16_t maxInt)
                : VoltRange(name, minValue, maxValue, voltGain, maxInt) {};
    };

}

#endif
```

Конструкторы VoltRange, VoltRangeDac, VoltRangeAdc на основе шаблона класса, описанного в классе ps_range.h

ps_voltammetry

```
#ifndef PS_VOLTAMMETRY_H
#define PS_VOLTAMMETRY_H

#include "ps_constants.h"
#include "ps_return_status.h"
#include "ps_base_test.h"
#include "ps_cyclic_test.h"
#include "ps_sinusoid_test.h"
#include "ps_constant_test.h"
```

```
#include "ps_linearsweep_test.h"
#include "ps_multistep_test.h"
#include "ps_squarewave_test.h"

#include "third-party/ArduinoJson/ArduinoJson.h"
#include "third-party/Array/Array.h"

namespace ps
{

    class Voltammetry
    {

        public:

            Voltammetry();

            BaseTest *getTest(String name);
            ReturnStatus getTest(JsonObject &jsonMsg, JsonObject &jsonDat, BaseTe
st* &testPtr);
            ReturnStatus getParam(JsonObject &jsonMsg, JsonObject &jsonDat);
            ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDat);
            ReturnStatus getTestDoneTime(JsonObject &jsonMsg, JsonObject &jsonDat
);
            ReturnStatus getTestNames(JsonObject &jsonMsg, JsonObject &jsonDat);
            ReturnStatus getMuxTestNames(JsonObject &jsonMsg, JsonObject &jsonDat
);

            void setSamplePeriod(uint64_t samplePeriod);

            BaseTest baseTest;
            CyclicTest cyclicTest;
            SinusoidTest sinusoidTest;
            ConstantTest constantTest;
            SquareWaveTest squareWaveTest;
            LinearSweepTest linearSweepTest;
            MultiStepTest<2> chronoampTest;
            MultiStepTest<MultiStepMaxSize> multiStepTest;

            static const String TestKey;

        protected:

            Array<BaseTest*,AvailableTestsMaxSize> availableTests_;

    };

}

#endif
```

В .h файле описывается класс Voltametry, с public методами:

*getTest(String name) - указатель на имя теста;

ReturnStatus getTest(JsonObject &jsonMsg, JsonObject &jsonDat, BaseTest* &testPtr) - сохраняет в json тест;

ReturnStatus getParam(JsonObject &jsonMsg, JsonObject &jsonDat) - сохраняет в json параметры теста;

ReturnStatus setParam(JsonObject &jsonMsg, JsonObject &jsonDat) - устанавливает параметры из json файла;

ReturnStatus getTestDoneTime(JsonObject &jsonMsg, JsonObject &jsonDat) - сохраняет в json файл время выполнения теста;

ReturnStatus getTestNames(JsonObject &jsonMsg, JsonObject &jsonDat) - сохраняет в json файл имена тестов;

ReturnStatus getMuxTestNames(JsonObject &jsonMsg, JsonObject &jsonDat) - сохраняет в json файл мультиплексорные имена тестов;

void setSamplePeriod(uint64_t samplePeriod) - устанавливает период времени между выборками;

Затем создаются все тесты описанные в файлах ранее:

BaseTest baseTest;

CyclicTest cyclicTest;

SinusoidTest sinusoidTest;

ConstantTest constantTest;

SquareWaveTest squareWaveTest;

LinearSweepTest linearSweepTest;

MultiStepTest<2> chronoampTest;

MultiStepTest<MultiStepMaxSize> multiStepTest;

Создается переменная: static const String TestKey;

protected методы:

Array<BaseTest*,AvailableTestsMaxSize> availableTests_;

В .cpp файле описывается как работают перечисленные выше методы.

ВЫВОД

Были изучены и проанализированы модули прошивки потенциостата.