СИТИС | СИВИОНИК®

# Драйв Терминал 1.00

# # 3100

**Руководство пользователя**
**3100 – РП**

**Предварительная редакция № 1.5**
**16.03.2015**

## Аннотация

Настоящий документ является руководством пользователя программы «Драйв: Терминал 1.00».

Руководство пользователя содержит описание интерфейсов программы для различных ОС, принцип работы, функции, возможности и другие сведения, необходимые для работы с программой «Драйв: Терминал 1.00».

Программное обеспечение СИТИС: СИВИОНИК постоянно совершенствуются, в настоящее руководство и в другую техническую документацию могут вноситься изменения и уточнения.

Рекомендуется пользоваться актуальной версией документации, размещенной на сайте WWW.CIVIONIC.RU

## Авторское право

**Оглавление**

# 1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Программа «Драйв Терминал» предназначена для задания команд устройствам и получения информации от устройств, поддерживающих обмен данными по стандарту RS-232, а при использовании адаптеров и переходников – с любым устройством, поддерживающим последовательный протокол обмена информацией.

С помощью программы возможно взаимодействовать как с компьютерами и компьютеризированными устройствами, такими как контроллеры, регистраторы, тахеометры, лабораторные приборы и другое подобное оборудование, так и с отдельными микросхемами стандартов 1WIRE, I2C и другими.

Ввод и отображение команд и информации возможен как в обычном текстовом режиме, так и виде последовательности шестнадцатеричных символов.

Для упрощения ввода последовательности команд поддерживается работа с макросами – пакетами команд, последовательно передаваемых подключенному устройству.

Также возможно задание достаточно сложных алгоритмов взаимодействия с подключенными устройствами и обработки полученных данных – в составе программы реализованы интерпретаторы команд на языках JavaScript, Phyton. Возможно подключение пакета математической обработки SciLab (бесплатный аналог MathLab).

Для обработки данных и передачи информации в программе предусмотрены библиотеки функций для работы с базами данных, электронными таблицами, обработки и преобразования рядов измеренных данных, отправки электронных писем и СМС, звонки на телефонные номера и другие возможности.

Возможно запускать макросы и скрипты в цикле с заданной периодичностью, для регулярного взаимодействия с подключенными устройствами.

# 2. ТЕРМИНЫ И СОКРАЩЕНИЯ

В тексте этого документа используются следующие термины и сокращения:

**ПК** — персональный компьютер.

**ПО** — программное обеспечение, совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

**ОС** — операционная система

**Шестнадцатеричный формат** — система счисления по целочисленному основанию 16. Обычно в качестве шестнадцатеричных цифр используются десятичные цифры от 0 до 9 и латинские буквы от A до F

**Макрос** — набор команд, передаваемых устройству за один раз.

**Последовательный порт** — разъем устройства или компьютера, предназначен для обмена информацией с другими устройствами по стандарту последовательной передачи данных RS-232.

**Скрипт** — программа в виде последовательности инструкций, которые выполняются интерпретатором во время работы программы.

# 3. ПРИНЦИП РАБОТЫ ПРОГРАММЫ

## 3.1. Последовательность действий при работе с программой

1) Запустить программу
2) Открыть проект (при необходимости)
3) Установить соединение с устройством, задав параметры через форму настроек COM-порта.
4) Выбрать режим отправки команд (режим «text» или режим «hex»).
5) Выбрать режим отображения (режим «text» или режим «hex»).
6) Последовательно вводить команды подключенному устройству в строке ввода команд. Отправка команды осуществляется по нажатии на клавишу «Enter» или активации кнопки «отправить»
7) Последовательно запускать макросы и скрипты с помощью советующих интерфейсных элементов
8) Сохранить проект

Файл проекта содержит настройки подключения, описание объектов, некоторые настройки отображения. Описание файла проекта см в разделе 13.

## 3.2. Макросы

Программа «Драйв Терминал» предоставляет возможность пользователю создавать и использовать наборы команд, называемые макросами.

В макросах можно вводить текстовые команды согласно принципу, одна команда – одна строчка. Для выполнения задержки между отправкой команд необходимо использовать специальную команду #sleepX, где X – время задержки в миллисекундах.

Специальные команды в тексте макросов начинаются с символа #.

Если необходимо отправить данные в шестнадцатеричной форме, каждую команду следует начинать с символа «0x». Символы возврата каретки и перевода строки при этом автоматически добавлены не будут, если они нужны, следует завершить строку кодами «0a0d».

При интерпретации данных как текстовых по умолчанию символы возврата катертки и перевода строки добавляются. Если к данной команде их добавлять не следует, необходимо в начале строки поставить символ «#nLfCr».

Пример макроса для работы с регистраторами СИТИС:Сивионик:

Данный макрос переключает режим логирования регистратора, передает команду получения файла с устройства и потом переключает режим логирования обратно

```
log 2
#sleep 200
fls
#sleep 1000
log 3
```

## 3.3. Скрипты

Программа «Драйв Терминал» предоставляет возможность пользователю создавать и использовать скрипты. Скрипт является специальной программой, способной запускаться для выполнения каких-либо действий заданному в скрипте алгоритму.

В программе «Драйв: Терминал» могут использоваться скрипты с использованием языков:

*JavaScript.*

*Phyton*

*SciLab*

Стандартным скриптовым языком, входящим в комплект поставки, является JavaScript. Интерпретаторы других языков поставляются по запросу.

Для обработки и распределения полученных с устройств данных , из скриптов возможно вызывать функции и методы статических классов библиотек, входящих в пакет поставки программы. Описание библиотек функций приведено в соответствующем разделе документации..

Глобальные переменные в скрипте очищаются при каждом новом запуске скрипта (при нажатии кнопки), соответственно, при циклической работе скрипта они существуют до тех пор, пока активна кнопка.

Пример скрипта для работы с регистраторами СИТИС:Сиывионик:

Данный скрипт в цикле обращается к устройству, проверяя не появился ли новый файл измерений, и если появился, то сохраняет его на диск и записывает результаты измерения в базу данных.

```
//при запуске скрипта в цикле открывать БД только один раз
var wasOpened;
if (isNaN(wasOpened))
    wasOpened = false;
if (!wasOpened) {
    dprintln("открытие БД sqlite");
    ds.project = "D:\\MyProject.dpsj";
    openDB(2, "D:\\MyDatabase.db");
```

```
    wasOpened = true;
}
//переменная для хранения номера последнего файла
var fno;
if (isNaN(fno))
    fno = 0;
//отправляем команду запроса номера последнего файла
sendText("fno", 1);
var str = "";
var fno_str = "";
//вычитываем ответ с com-porta
while (isLine())
    str += getLine();
print("ответ на fno: " + str);
//если формат ответа верный: fno>msr_n tpl_n
if (str.indexOf("fno>") != -1) {
    fno_str = str.substr(str.indexOf(">") + 1);
    var ar = fno_str.split(" ");
    //получаем значение первого номера
    var new_no = ar[0];
    //если разбор ответа произошел правильно
    //и полученный номер больше, чем предыдущий
    if (ar.length == 2 && new_no > fno) {
        fno = new_no;
        print("изменен номер последнего файла: " + fno);
        //оправляем команду получения последнего файла
        sendText("fls", 1);
        //пытаемся прочитать файл с устройства
        //и сохранить в папке files
        var res = createMSRFileAndGetName("files");
        if (res != "") {
            print("Файл создан успешно!");
            dprintln("импорт в sqlite");
            importMsrFile(res, 2);
        }
    }
}
else
      print("Неверный формат ответа на команду fno");
```

При разработке скриптов не следует предусматривать бесконечные циклы для ожидания отклика подключенного устройства. Создание бесконечных циклов является крайне нежелательным, так как может вызвать непредвиденные проблемы в работе программы, такие как блокировка интерфейса, утечка памяти и тому подобные.

## 3.4. Специальные команды

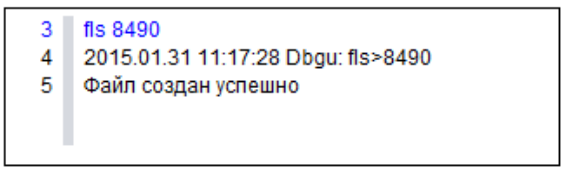
Помимо простой работы в режиме отправки введенных команд и показа отклика устройства на экране, программа «Драйв Терминал» предоставляет возможность специальной обработки отклика подключенных устройств после команд, предназначенных для обмена файлами данных с регистраторами и другими устройствами.

Например:

**fls** - команда получения файлов измерений с регистраторов СИТИС:Сивионик.

После отправления данной команды подключенный к устройству прибор отправит текстовый файл с результатами измерений в виде последовательности символов, оканчивающихся символом конца файла. Программа «Терминал» распознает эту последовательность как файл и сохранит его в указанную пользователем папку.

На рисунке ниже приведен пример использования специальной команды fls и ответа подключенного регистратора.



*Папка для сохранения файлов задается в меню «Настройки» программы, в пункте «Настройки интерфейса».*

## 4. ИНТЕРФЕЙС ДЛЯ КОМПЬЮТЕРОВ И НОУТБОКОВ

Интерфейс для компьютеров и ноутбуков с ОС Windows подходит для устройств с большими мониторами. Для компьютеров и ноутбуков подходит режим отображения «Обычные кнопки» см. п. 4.2.2. Ввод команд, макросов и скриптов осуществляется при помощи клавиатуры и мышки. На рисунке 2 представлена фотография ноутбука с программой «Драйв: Терминал 1.00».

***Рис. 2** Вид программы на ноутбуке*

На рисунке 3 представлен комплекс оборудования - ноутбук с установленной программой и подключенный к ноутбуку регистратор «Краб».

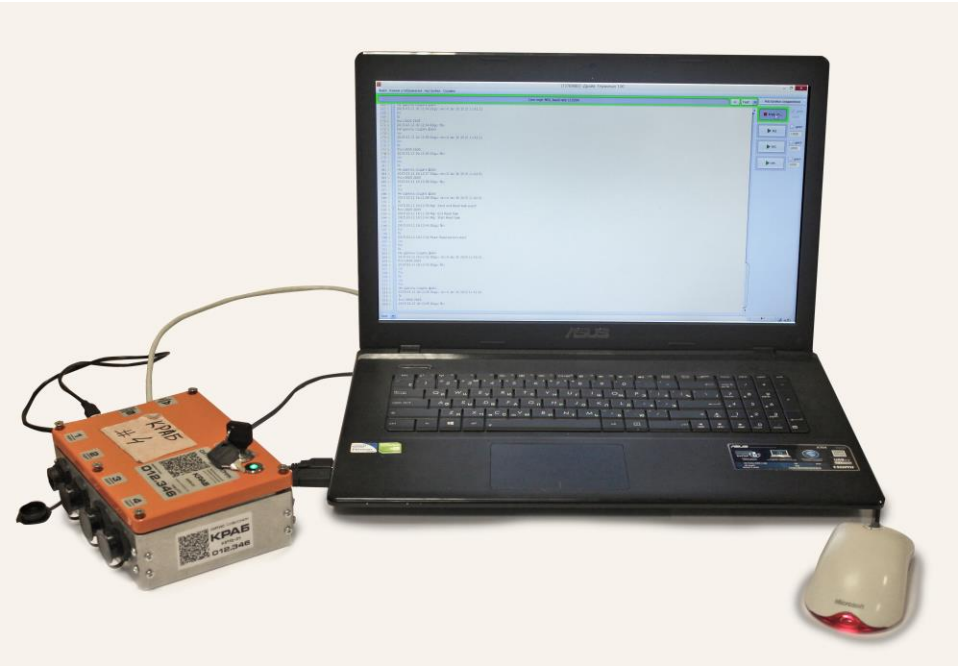


***Рис. 3** Работа программы с устройством «Краб» на ноутбуке*

Интерфейс для компьютеров и планшетов имеет следующую структуру см. рисунок 4:

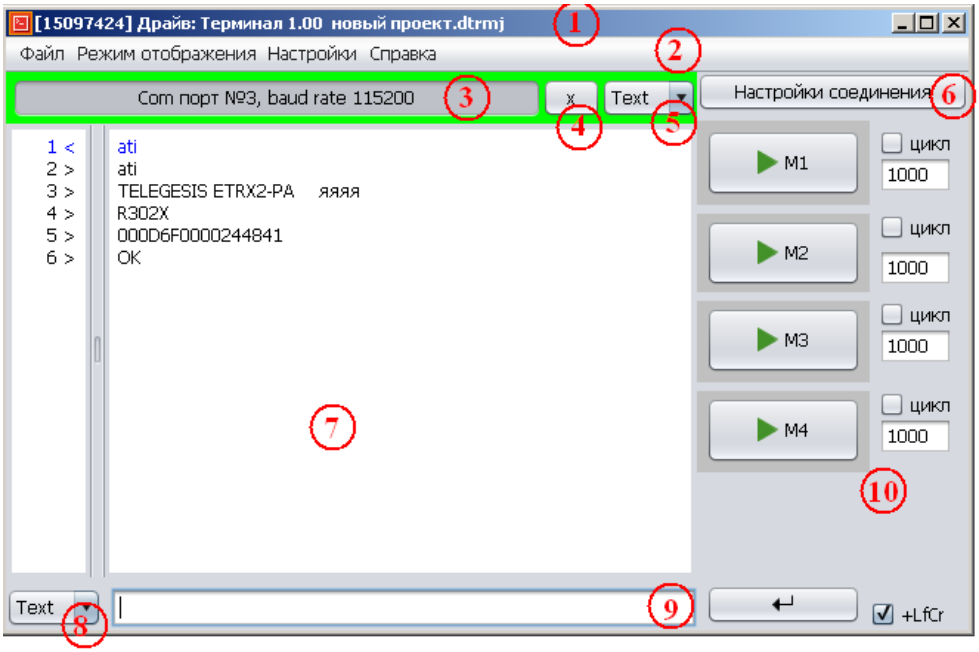



***Рис. 4** Интерфейс окна программы для компьютеров и ноутбуков с ОС – Windows, Linux*

1) Заголовок окна. См. п. 8.1
2) Главное меню. См. п. 8.2
3) Отображение параметров подключения. См. п. 8.3
4) Режим очистки поля введенных команд. См. п. 8.4
5) Режим отображения введенных команд и ответа системы. См. п. 8.5
6) Установка соединения. См. п. 8.6
7) После списка введенных команд и ответов системы. См. п. 8.7
8) Режим ввода команд. См. п. 8.8
9) Командная строка. См. п. 8.9
10) Макросы и скрипты. См. п. 8.10

## 5. ИНТЕРФЕЙС ДЛЯ ПЛАНШЕТОВ с ОС Windows

Интерфейс для планшетов с ОС Windows подходит для устройств с небольшими экранами (например, 8-10 дюймов) и ориентирован на ввод с использованием экранных клавиатур на сенсорных экранах. Для планшетов с небольшими экранами предусмотрен режим отображения «Огромные кнопки» см. п. 4.2.2. Обратите внимание, что режим «Огромные кнопки» больше подходит для вертикальной ориентации планшета. Ввод команд, макросов и скриптов сенсорный. На рисунке 5 представлена фотография планшета с программой.

***Рис. 5*** *Вид программы на планшете Asus Note 8*

На рисунке 6 представлен комплекс оборудования - планшет с установленной программой и устройство «Краб».


***Рис. 6*** *Работа программы с устройством «Краб» на планшете Asus Note 8*

Интерфейс планшетов с небольшими экранами имеет следующую структуру см. рис. 7:

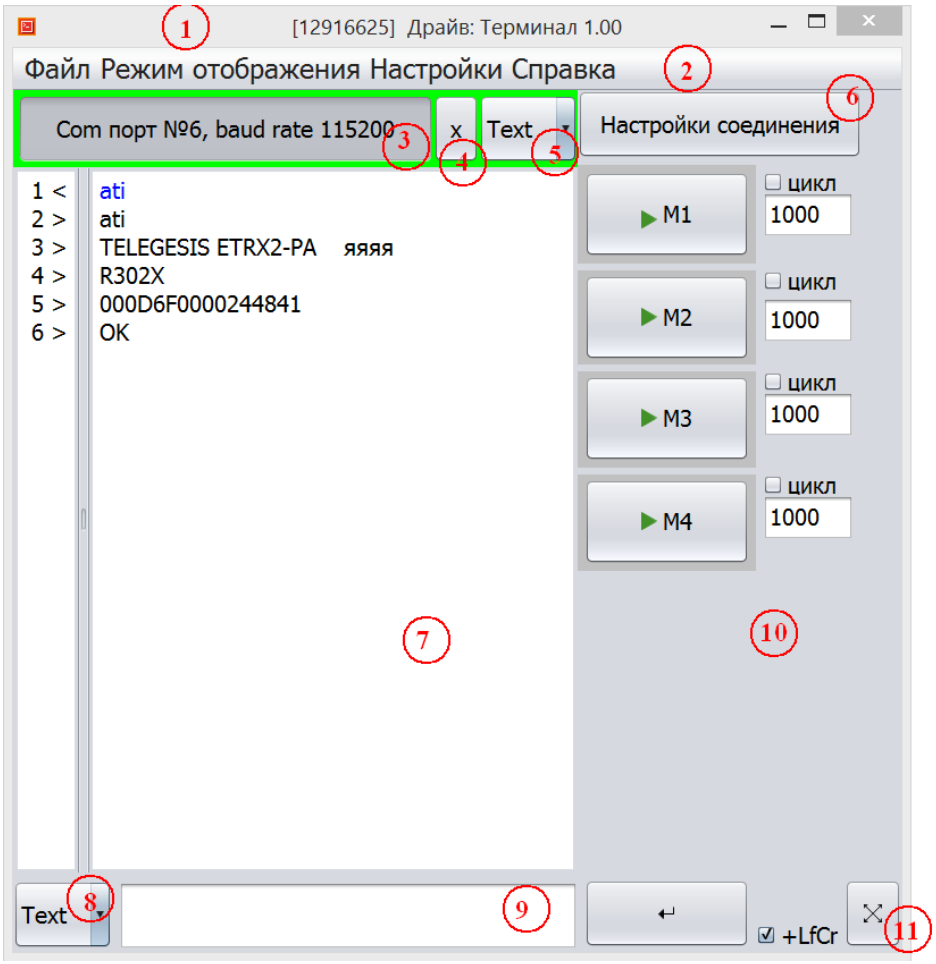



***Рис. 7** Интерфейс окна программы для планшетов с ОС – Windows, Linux*

1) Заголовок окна. См. п. 8.1
2) Главное меню. См. п. 8.2
3) Отображение параметров подключения. См. п. 8.3
4) Режим очистки поля введенных команд. См. п. 8.4
5) Режим отображения введенных команд и ответа системы. См. п. 8.5
6) Установка соединения. См. п. 8.6
7) После списка введенных команд и ответов системы. См. п. 8.7
8) Режим ввода команд. См. п. 8.8
9) Командная строка. См. п. 8.9
10) Макросы и скрипты. См. п. 8.10
11) Автоматическое размещение окна программы в верхней части экрана см. п. 8.11

# 6. ИНТЕРФЕЙС ДЛЯ ПЛАНШЕТОВ И СМАРТФОНОВ С ОС ANDROID

Для планшетов и смартфонов с ОС Android специальный интерфейс – с тремя окнами. В первом окне осуществляется работа с устройством, во втором – работа с макросами, в третьем - работа со скриптами. Для того чтобы перейти от одного окна к другому нужно провести пальцем вправо (лево). Ввод команд, макросов и скриптов сенсорный. На рисунке 8 представлена фотография смартфона с программой «Драйв: Терминал 1.00» (ОС – Android).

***Рис. 8*** *Вид программы на смартфоне HTC One 32 Gb*

На рисунке 9 представлен комплекс оборудования - смартфон с программой и устройство «Краб».

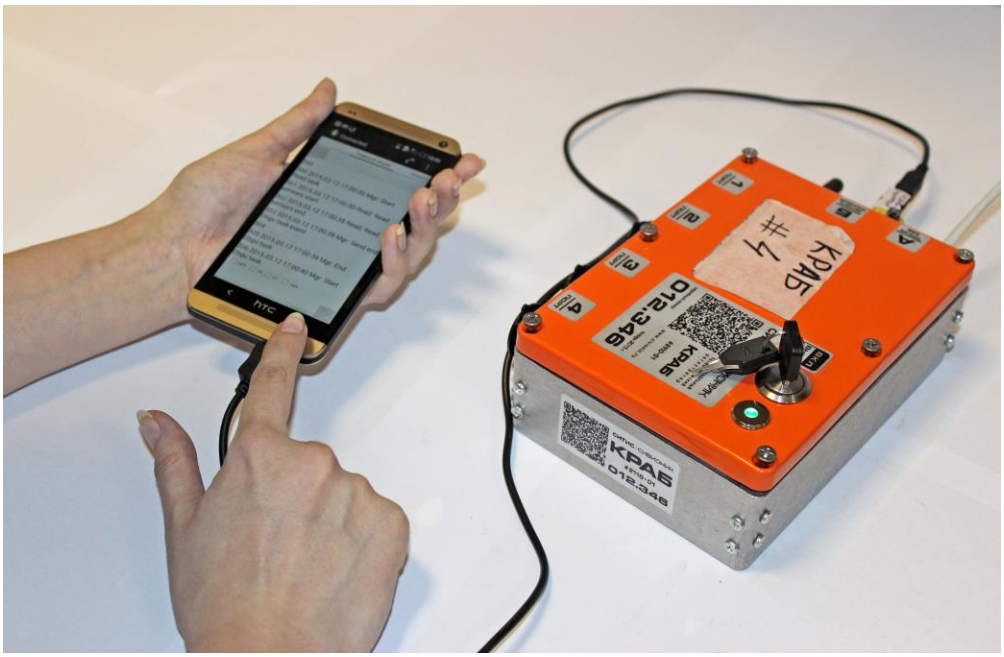


***Рис. 9*** *Работа программы с устройством «Краб» на смартфоне HTC One 32 Gb*

Интерфейс смартфонов имеет следующую структуру см. рис. 10:

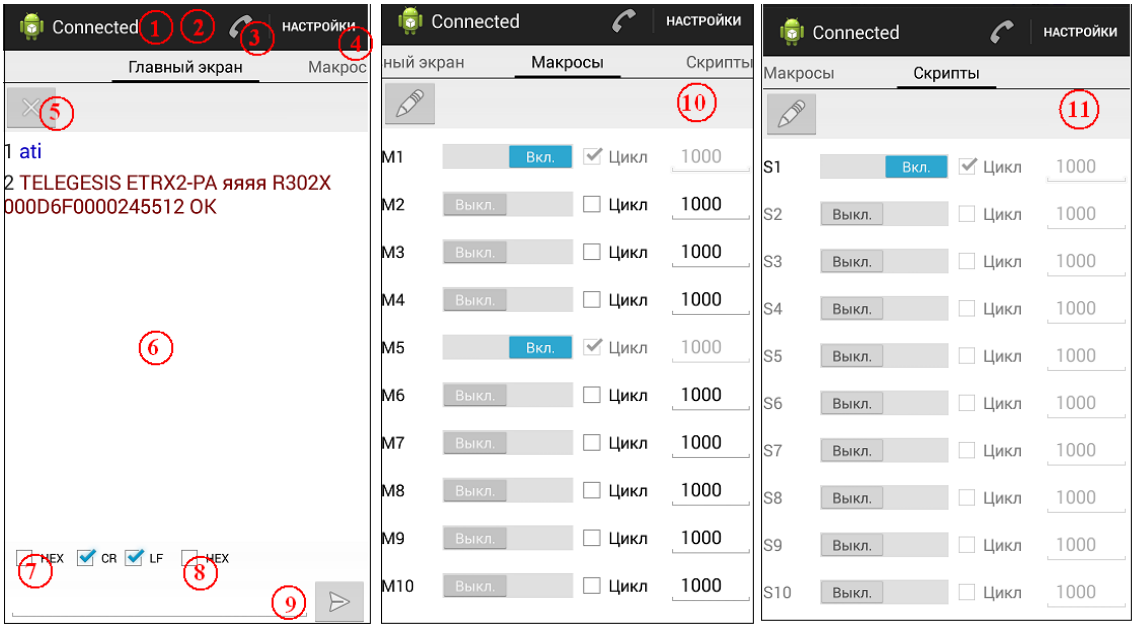



***Рис.10*** *Интерфейс окон программы для смартфонов с ОС – Android*

1) Заголовок окна. См. п. 9.1
2) Отображение параметров подключения. См. п. 9.3
3) Установка соединения. См. п. 9.6
4) Настройки. См. п. 9.2
5) Режим очистки поля введенных команд. См. п. 9.4
6) Поле списка введенных команд и ответов системы. См. п. 9.7
7) Режим ввода команд. См. п. 9.8
8) Режим отображения введенных команд и ответа системы. См. п. 9.5
9) Командная строка. См. п. 9.9
10) Макросы. См. п. 9.10
11) Скрипты. См. п. 9.11

## 7. КОНСОЛЬНЫЙ РЕЖИМ ДЛЯ МИНИКОМПЬЮТЕРОВ

Консольный режим для миникомпьютеров с ОС Windows подходит для устройств с слабыми графическими адаптерами, поддерживающими небольшое разрешение мониторов. Ввод команд, запуск макросов и скриптов осуществляется при помощи текстовых команд, без графического интерфейса пользователя.

На рисунке 11 представлена фотография миникомпьютера


***Рис. 11*** *Миникомпьютер Ebox 3310mx-ap*

Запуск программы с ключом r=c указывает на необходимость работы в консольном режиме (не следует использовать другие ключи при запуске, если ключ r не был введен)

Для запуска программы в консольном режиме необходимо ввести в командной строке следующую команду:

*DTerminal.jar r=c*

В консольном режиме могут также использоваться ключи, задающие запуск макросов и скриптов.

В ОС Windows для запуска консольного режима может быть создан bat файл, содержащий параметры иключи запуска программы, например:

*java –jar DTerminal.jar r=c p=project.dtrm,1*

Пример bat файла для запуска программы в консольном режиме находится в папке с установленной программой (console.bat).

**Ключи запуска программы в консольном режиме:**

Ключи p, m и s - взаимоисключающие.

r=c – запуск программы в терминальном режиме

p=путь_к_файлу_проекта , номер_объекта – открыть проект «Драйв: Терминал» (*.dtrmj) для автоматического запуска объекта (скрипта или макроса) из проекта.

**Пример:** p=project.dtrmj,1 – запуститься на выполнение первый объект из проекта « project.dtrmj». Свойства цикличности, задержки и интерпретации (скрипт или макрос) будут также прочитаны из файла проекта.

m=путь_к_файлу_макроса – с настройками по умолчанию запустить в цикле макрос, находящийся в файле «путь_к_файлу_макроса».

s=путь_к_файлу_скрипта – с настройками по умолчанию запустить в цикле скрипт, находящийся в файле «путь_к_файлу_скрипта».

В случае, когда выполнение скрипта или макроса происходит не из указанного проекта программы, настройки по умолчанию для открытия соединения берутся из файла “defaultComSettings.json”, в котором они могут быть заданы пользователем по своему усмотрению.

Прекратить выполнение цикла позволяет команда stop, введенная в консоли работающей программы.

Команды для управления программой в консольном режиме см разделе 12

# 8. ЭЛЕМЕНТЫ ИНТЕРФЕЙСА ПРОГРАММЫ

## 8.1. Заголовок окна.

В заголовке окна программы «Драйв: Терминал 1.00» отображаются информационные данные о программе и файле: наименование и версия программы (например, Драйв: Терминал 1.00.1508) и наименование файла (например, новый проект. .dtrmj) см. рис. 12.


***Рис. 12** Заголовок окна программы для устройств с ОС Windows*

## 8.2. Главное меню.

С помощью меню осуществляется работа с программой. Меню состоит из следующих пунктов:

- Файл
- Режим отображения
- Настройки
- Справка

**ПУНКТ МЕНЮ «ФАЙЛ»**

Пункт меню «Файл» содержит следующие подпункты:

- Открыть
- Сохранить
- Сохранить как
- Выход

Пункт «Открыть» - открывает существующий проект с расширением . dtrmj.

Пункт «Сохранить» - сохраняет текущий проект программы с расширением . dtrmj

Пункт «Сохранить» - сохраняет текущий проект программы «Драйв: Терминал 1.00» с расширением . dtrmj с новым именем файла.

Пункт «Выход» - закрывает программу «Драйв: Терминал 1.00»

**ПУНКТ МЕНЮ «РЕЖИМЫ ОТОБРАЖЕНИЯ»**

Пункт меню «Режимы отображения» содержит следующие подпункты:

- Огромные кнопки
- Большие кнопки
- Обычные кнопки

Пункт «Огромные кнопки» - делает кнопки и шрифт окон программы «Драйв: Терминал 1.00» огромными. Этот режим удобно применять для работы на планшете с небольшой диагональю экрана (например, 8 дюймов).

Пункт «Большие кнопки» - делает кнопки и шрифт окон программы «Драйв: Терминал 1.00» большими. Этот режим удобно применять для работы на планшете.

Пункт «Обычные кнопки» - делает кнопки и шрифт окон программы «Драйв: Терминал 1.00» стандартного размера. Этот режим удобно применять для работы на ПК или ноутбуке.

**ПУНКТ МЕНЮ «НАСТРОЙКИ»**

Пункт меню «Настройки» содержит следующие подпункты:

- Настройки интерфейса
- Настройки соединения

Пункт «Настройки интерфейса» открывает окно с настройками терминала см. рисунок 13.

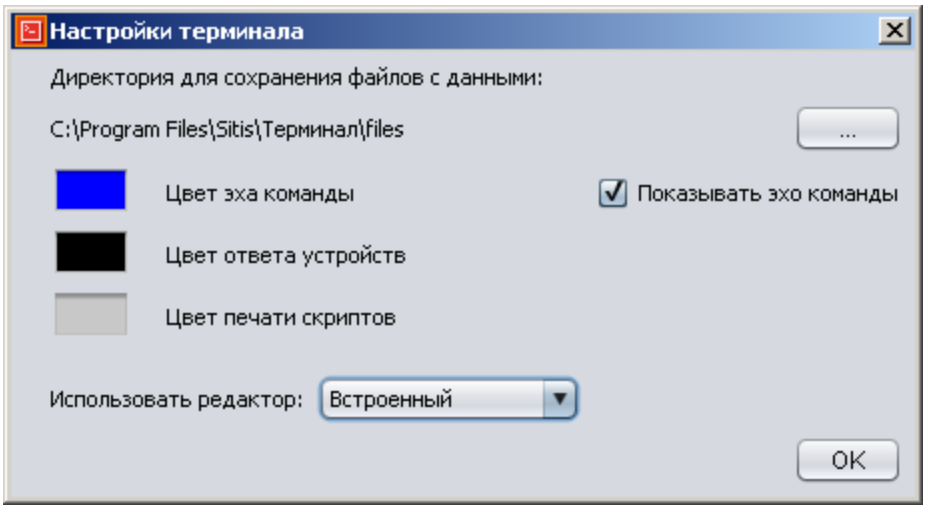



***Рис. 13** Настройки программы*

В окне «Настройки терминала» при помощи кнопки  «Выбрать директорию» можно выбрать директорию для сохранения файлов с данными, полученных при помощи команды fls.

Галочка возле строки «Показывать эхо команды» показывает отображать или не отображать эхо команды в поле списка введенных команд и ответа системы.

Цвет эха команды можно изменить, нажав по цветному квадрату возле строки «Цвет эха команды». Цвет вывода эха команды отличается от вывода ответа устройств.

Цвет ответа устройств можно изменить, нажав по цветному квадрату возле строки «Цвет ответа устройств».

Цвет печати скриптов можно изменить, нажав по цветному квадрату возле строки «Цвет печати скриптов».

Выпадающий список «Использовать редактор», определяет какой редактор нужно использовать для редактирования макросов и скриптов. Редактор может быть «встроенным» и «сторонний». Сторонний редактор определяет пользователь. Например, в качестве стороннего редактора может быть выбрана программа «Блокнот».

Пункт «Настройки соединения» открывает окно с настройками COM-порта см. рисунок 14.

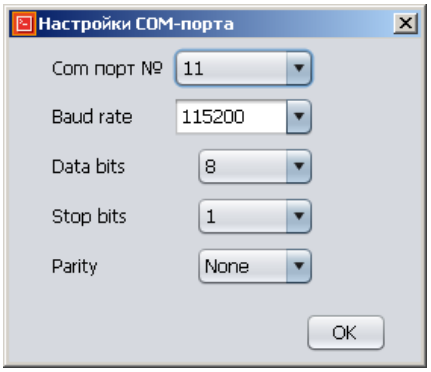



***Рис. 14** Настройки соединения COM-порта в программе*

Чтобы установить соединение, необходимо через форму настроек COM-порта задать параметры желаемого соединения и нажать кнопку «ОК» для подключения. Параметры подключения отображаются на кнопке подключения.

Подключиться можно только к доступному порту.

ПУНКТ МЕНЮ «СПРАВКА

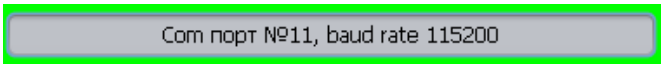
Пункт меню «Справка» содержит следующие подпункты:

- О программе
- Руководство пользователя программы Драйв-Тест: Терминал.pdf

Пункт «Справка» открывает окно с информацией о программе.

Пункт «Руководство пользователя программы Драйв-Тест: Терминал.pdf» открывает документ с руководством пользователя программы.

## 8.3. Отображение параметров подключения

Параметры подключения отображаются в кнопке 

«Подключение».

Нажатое состояние и зеленая рамка вокруг кнопки подключения соответствуют установленному подключению. Если кнопка не нажата и зеленая рамка отсутствует, то подключение не выполнено. Параметры подключения отображаются на кнопке подключения (Например, Com-порт №11, baud rate 115200).

Существует возможность произвести подключение, не изменяя настроек. После клика по кнопке подключения будет произведена попытка установки соединения с отображаемыми параметрами.

## 8.4. Режим очистки поля введенных команд

Очистить поле введенных команд можно с помощью кнопки 

«Очистить поле вывода». Для этого нужно нажать на кнопку и поле введенных команд очиститься. После очистки поля введенных команд нумерация начинается заново

## 8.5. Режим отображения введенных команд и ответа системы

Режим отображения веденных команд и ответа системы осуществляется при помощи выпадающего списка

.

Поле списка введенных команд и ответа системы можно просматривать в двух режимах «text» и «hex».

## 8.6. Установка соединения

Установка соединения осуществляется с помощью кнопки «Настройки соединения». Кнопка «Настройки соединения» открывает окно с настройками COM-порта и имеет следующий вид см. рисунок 15:

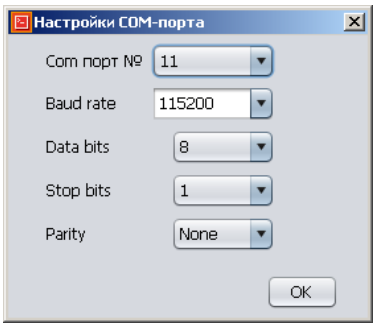



***Рис. 15*** *Настройки соединения COM-порта в программе*

Чтобы установить соединение, необходимо через форму настроек COM-порта задать параметры желаемого соединения и нажать кнопку «ОК» для подключения. Параметры подключения отображаются на кнопке подключения.

Подключиться можно только к доступному порту.

## 8.7. Поле списка введенных команд и ответа системы

В поле введенных команд и ответа системы отображаются введенные пользователем команды и ответы программы на эти команды. Введенные команды имеют цвет эха команды, заданный в окне «Настройки терминала». Вывод полученных данных осуществляется в окно программы терминал за исключением обработки специальных команд.

Цвет вывода эха команды отличается от вывода ответа устройств.

Если ответа на переданную команду нет в течение некоторого таймаута, выводится сообщение о разрыве соединения. Отключение от порта не происходит. При обнаружении данного сообщения пользователь должен самостоятельно проверить соединение с устройством.

Если com-порт в системе был физически отключен, произойдет разрыв соединения.

## 8.8. Режим ввода команд

Режим отображения веденных команд и ответа системы осуществляется при помощи выпадающего списка


.

Команды можно ввести в двух режимах «text» и «hex». «Text» - текстовый формат ввода команд. «Hex» - шестнадцатеричный формат для ввода команд.

## 8.9. Строка ввода команд

Отправка команд устройству осуществляется из командной строки терминала по нажатию кнопки «send» или Enter см. рисунок 16

***Рис. 16*** *Строка ввода команд в программе*

Отправка сообщений возможна лишь при установленном соединении, иначе кнопка «send» неактивна.

## 8.10. Макросы и скрипты

Работать с макросами нужно при помощи кнопки «Макросы и скрипты» см. рисунок 17.


***Рис. 17** Кнопка для работы с макросами и скриптами*

Для того чтобы связать текст макроса или скрипта с элементом интерфейса (кнопкой), нужно воспользоваться контекстным меню (нажатие правой кнопки на элементе). Пункт меню «редактировать» откроет окно, в котором можно записать текст макроса или скрипта. Пункт меню «Макросы» позволяет выбрать содержимое одно из сохраненных макросов. Пункт меню «Скрипты» позволяет выбрать содержимое одно из сохраненных скриптов.

Посмотреть макрос или скрипт можно, вызвав форму редактирования.

Помимо разового выполнения макроса или скрипта, существует возможность непрерывного его повторения с заданной задержкой (в миллисекундах). Для этого необходимо отметить галочкой «Цикл».

Установленные макросы и скрипты сохраняются в проекте. Более подробно работу с макросами можно посмотреть в п. 3.2. Более подробно работу с макросами можно посмотреть в п. 3.3.

## 8.11. Автоматическое размещение окна.

Автоматическое размещение окна программы в верхней части экрана используется на планшетах с ОС Windows, чтобы экранная клавиатура не закрывала окно ввода-вывода. Режим осуществляется при помощи нажатия на кнопку  «Разместить окно программы в верхней части экрана». Кнопка «Разместить окно программы в верхней части экрана» находится в правом нижнем углу окна программы «Драйв: Терминал» и становится доступной только при использовании режимов «Огромные кнопки» и «Большие кнопки».

# 9. ИНТЕРФЕЙС ПРОГРАММЫ для ОС Android

## 9.1. Заголовок окна.

В заголовке окна программы «Драйв: Терминал» для устройств с ОС Android отображается наименование программы см. рис. 18. Обратите внимание, что заголовок окна перестает отображаться в главных окнах после подключения. В окне с настройками соединения, редактирования макросов и других не главных окнах заголовок окна остается.

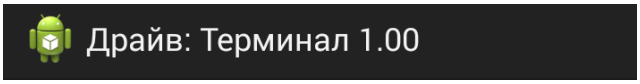



***Рис. 18** Заголовок окна программы для устройств с ОС Android*

## 9.2. Настройки.

Главное меню программы «Драйв: Терминал 1.00» для устройств с ОС Android вызывается кнопкой «Настройки».

Пункт «Настройки» состоит из следующих подпунктов:

- Настройки соединения
- Настройки интерфейса
- Информация об устройстве

Пункт «Настройки соединения» открывает окно с настройками COM-порта см. рисунок 19.

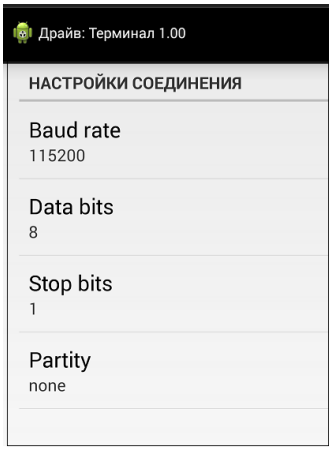



***Рис. 19** Настройки соединения COM-порта в программе*

Чтобы установить соединение, необходимо через форму настроек соединения задать параметры желаемого соединения. Для этого нужно нажать на нужную строку, после этого появится форма со значением в этой форме нужно выбрать значение. Выбранные значения будут отображаться под название строки с величиной. Если значения в окне «Настройки соединения не выбраны» по умолчанию программа принимает значения:

- Baud rate - 115200
- Data bits – 8
- Stop bits – 1
- Partity – none

Пункт «Настройки интерфейса» открывает окно с настройками программы см. рисунок 20.

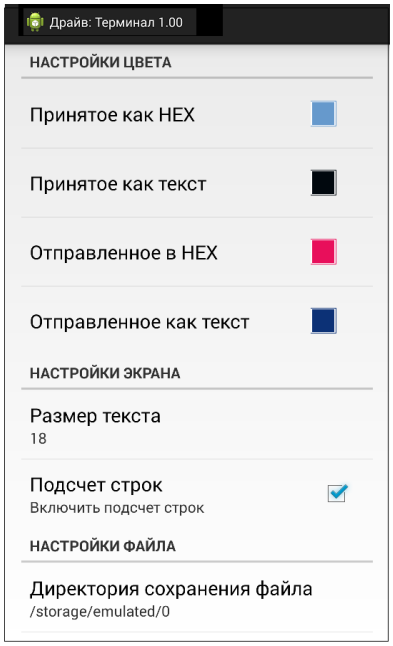



***Рис. 20** Основные настройки программы*

Все настройки делятся на 3 группы: «Настройки цвета», «Настройки экрана», «Настройки файла».

Настройки цвета состоят из следующих значений:

- «Принятое как HEX» — обозначает каким цветом будет выделен принятый текст в шестнадцатеричном режиме (HEX).
- «Принятое как текст» — обозначает каким цветом будет выделен принятый текст в текстовом режиме.

- «Отправленное в HEX» — обозначает каким цветом будет выделен отправленный текст (команда) в шестнадцатеричном режиме (HEX).
- «Отправленное как текст» — обозначает каким цветом будет выделен отправленный текст (команда) в текстовом режиме.

Настройки цвета состоят из следующих значений:

- «Размер текста» — определяет размер шрифта в поле вывода команд и ответа системы. Для смены размера теста нужно нажать на строку «Размер текста» и в появившемся окне указать нужное значение.
- «Подсчет строк» — при поставленной галочке добавляет подсчет строк в поле ввода команд и ответа системы, при отключенной галочке строке в поле ввода команд и ответа системы не нумеруются.

Настройки файла состоят из значения:

- «Директория сохранения файла» — в этой строке можно выбрать директорию для сохранения файлов с данными, полученных при помощи команды fls.

Пункт «Информация об устройстве» — открывает окно с информацией об устройстве и количестве макросов, которые можно запускать одновременно. См. рисунок 21

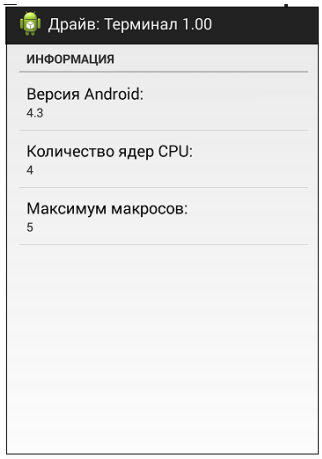



***Рис. 21** Информация об устройстве и макросах*

## 9.3. Отображение параметров подключения

Параметры подключения можно посмотреть на вкладке «Settings»/ «Настройки соединения». Состояние подключения «Connected»/ «Disconnected» отображается в главных окнах в заголовке окна см. рисунок 22


***Рис. 22** Состояние подключения к устройству*

## 9.4. Режим очистки поля введенных команд

Очистить поле введенных команд можно с помощью кнопки  «Очистить поле вывода». Для этого нужно нажать на кнопку и поле введенных команд очиститься. После очистки поля введенных команд нумерация будет продолжаться.

## 9.5. Режим отображения введенных команд и ответа системы

Поле списка введенных команд и ответа системы можно просматривать в двух режимах «text» и «hex». Для того чтобы установить режим «hex» нужно установить галочку возле правого значения «hex» см. рисунок 23.

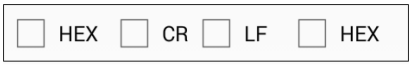



***Рис. 23** Включенный режим «текст»*

## 9.6. Установка соединения

Для того чтобы установить соединение необходимо в правом верхнем углу нажать на “трубку”  : Параметры соединения задаются на вкладке «Settings»/ «Настройки соединения».

## 9.7. Поле списка введенных команд и ответа системы

В поле введенных команд и ответа системы отображаются введенные пользователем команды и ответы программы на эти команды. Введенные команды имеют цвет эха команды, заданный в окне «Настройки терминала». Вывод полученных данных осуществляется в окно программы терминал за исключением обработки специальных команд.

## 9.8. Режим ввода команд

Поле списка введенных команд и ответа системы можно просматривать в двух режимах «text» и «hex». Для того чтобы установить режим «hex» нужно установить галочку возле правого значения «hex» см. рисунок 24.


***Рис. 24*** *Ввод команд в режиме «текст»*

## 9.9. Строка ввода команд

Отправка команд устройству происходит с помощью командной строки см. рисунок 25.

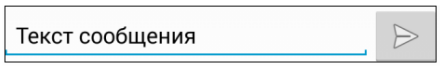



***Рис. 25*** *Командная строка*

Отправка команд устройству осуществляется из командной строки терминала по нажатию кнопки  «Отправить». Отправка сообщений возможна лишь при установленном соединении.

## 9.10. Макросы

Настройка макросов осуществляется с помощью кнопки  . Окно для редактирования макросов представлено на рисунке 26

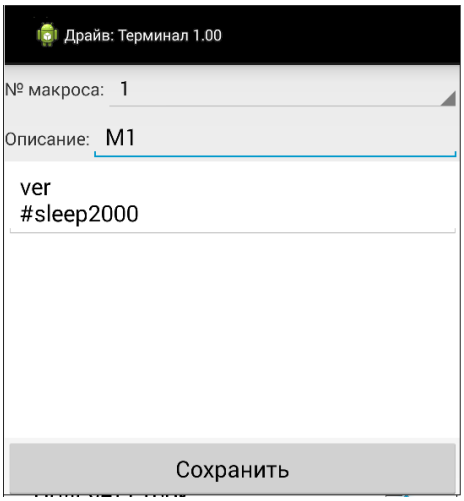



***Рис. 26*** *Окно для редактирования макросов*

В строке «№ макросов» нужно выбрать номер макроса, который необходимо редактировать.

В строке «Описание» задается описания к макросу.

В поле, идущим после строки «Описание», задаются команды макроса.

Список всех макросов (всего их может быть 10) будет отображаться в окне 2 см. рисунок 27.

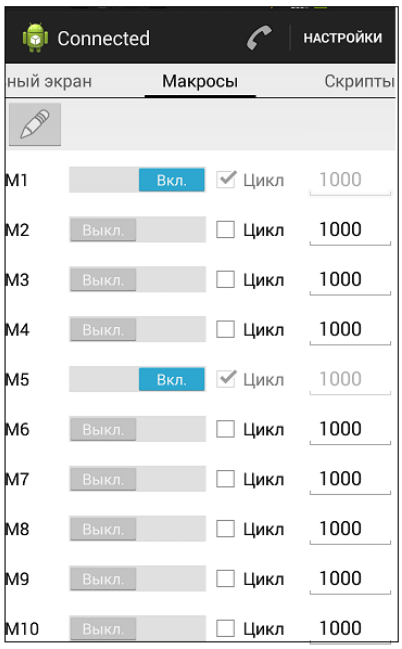



***Рис. 27** Список макросов в программе.*

Для того, чтобы запустить макрос нужно установить значение «1» или «вкл» (в зависимости от ОС).

Помимо разового выполнения макроса, существует возможность непрерывного его повторения с заданной задержкой (в миллисекундах). Для этого необходимо отметить галочкой «Цикл».

## 9.11. Скрипты

Настройка скриптов осуществляется с помощью кнопки  . Окно для редактирования скриптов представлено на рисунке 28

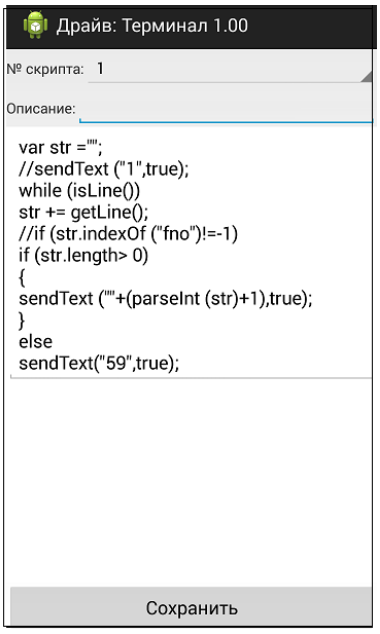



***Рис. 28** Окно для редактирования макросов*

В строке «№ скрипта» нужно выбрать номер скрипта, который необходимо редактировать.

В строке «Описание» задается описания к скрипту.

В поле, идущим после строки «Описание», записывается скрипт.

Список всех скриптов (всего их может быть 10, но запущенным может быть только 1) будет отображаться в окне 3 см. рисунок 29.

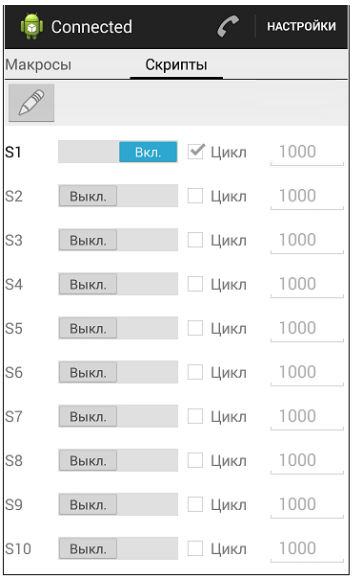



***Рис. 29** Список скриптов в программе*

Для того, чтобы запустить скрипт нужно установить значение «1» или «вкл» (в зависимости от ОС).

Помимо разового выполнения скрипта, существует возможность непрерывного его повторения с заданной задержкой (в миллисекундах). Для этого необходимо отметить галочкой «Цикл».

В текущей версии программы работа со скриптами недоступна.

## 10. ТАБЛИЦА ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

| | **Компьютеры, ноутбуки (Windows, Linux)** | **Планшеты (Windows, Linux)** | **Планшеты, смартфоны (Android)** |
|---|---|---|---|
| Ввод с помощью мышки и клавиатуры | + | + | + |
| Сенсорный ввод | + | + | + |
| Заголовок окна | + | + | + |
| Главное меню | + | + | + |
| Отображение параметров подключения | + | + | - |
| Режим очистки поля введенных команд | + | + | + |
| Режим отображения введенных команд и ответа системы | + | + | + |
| Установка соединения | + | + | + |
| Поле списка введенных команд и ответов системы | + | + | + |
| Режим ввода команд | + | + | + |
| Командная строка | + | + | + |
| Автоматическое размещение окна программы в верхней части экрана | | + | |
| Макросы | + | + | + |
| Скрипты | + | + | + |
| JavaScript | + | + | + |
| Phyton | + | + | |
| SciLab | + | + | |

# 11. БИБЛИОТЕКИ ФУНКЦИЙ

## 11.12. Описание библиотек

Для возможности вызова из произвольных скриптовых языков, функции библиотек в программе реализованы как методы специальных статических классов, написанных на языке Java.

Для удобства программирования на скриптовых языках, на этих языках написаны библиотеки функций, соответствующих базовым методам встроенных классов Java.

В состав поставки программы входит библиотека функций для JavaScript. Библиотеки для других скриптовых языков передаются по запросу.

Описание методов базовых классов Java приведено в отдельном документе.

## 11.13. Библиотека JavaScript

### 11.13.1 Работа с данными в текстовом режиме

**sendCmd** - отправка одной команды в текстовом режиме с добавлением LfCr

***SendCmd****(команда)*

**sendText** - отправка текстовых данных.

***sendText****(текст сообщения, LfCr)*

*LfCr (bool) - признак последнего блока данных (добавлять ли в конце LfCr)*

**getLine** - получение строки. Возвращает текстовую строку.

***getLine()***

**isLine** - проверка наличия данных в буфере

** (если установлен текстовый режим чтения, то выполняется проверка наличия именно строки, оканчивающейся ('\n') Если установлен шестнадцатеричный режим, то выполняется проверка данных в буфере независимо от признака окончания строки).*

***function isLine()***

### 11.13.2 Функции для работы с данными в шестнадцатиричном и бинарном режимах

**sendHex** - отправка данных в 16-чной форме (два символа - один байт)

***function sendHex****(данные в Hex формате, LfCr)*

**sendBin** - отправка бинарных данных

***sendBin****(набор бинарных данных, длина набора, LfCr)*

**isByte** - проверка наличия байта в буфере

***isByte()***

**getByte** - получение байта

***getByte()***

***getBin(binEnd)*** *- получения окончания бинарного набора данных*

### 11.13.3 Функции для работы с файлами устройств

**tryGetTextFile** – попытка чтения текстового файла с устройства (до признака окончания файла EOT). Возвращает строку с содержимым файла.

***tryGetTextFile()***

**createMeasureFile** - создание файла с измерениями на диске (с устройства «Краб»). Возвращает признак успешного создания файла

***createMeasureFile****(папка для сохранения файла)*

**createMSRFileAndGetName** - создание файла с измерениями на диске (с устройства «Краб»). Возвращает путь до созданного файла

*c****reateMSRFileAndGetName****(папка для сохранения файла)*

### 11.13.1 Функции для работы с наборами датчиков и величин

**dataSet** – создание пустого набора данных, или очистка набора

***dataSet*** *(номер набора)*

**dataAdd** – добавление данных в набор

***dataAdd****(номер набора, серийный номер датчика, тип величины, номер величины)*

***dataArrayAdd*** *(номер набора, массив серийных номеров датчиков, массив типов величины, массив номер величины)*

***dataAdd****(номер набора, имя набора в текущем файле проекта) – не реализована*

***dataAdd****(номер набора, имя файла, тип файла=0) – не реализована*

*тип файла: 0 – по расширению, 1-JSON, 2-CSV*

**sensorNum** – количество датчиков в наборе

***sensorNum****(номер набора)*

**valueNum** – количество величин в наборе

***valueNum****(номер набора)*

***valueNum****(номер набора, серийный номер датчика)*

**sn** – серийный номер датчика в наборе

*sn(номер набора, порядковый номер датчика)*

**tv** – тип величины

### 11.13.2 Функции для работы с базами данных

**createDB** – создание базы данных

***createdDB****(номер базы, тип базы, вид базы, имя файла(Sqlite) или базы (MySql), хост, порт = 3306, пользователь, пароль)*

*тип базы – число, определяющее тип базы: хост, сервис, сетевая среда, дамп (числовое значение типа базы берется из списка констант)*

*вид базы – число, определяющее тип соединения: Sqlite, MySql (числовое значение типа базы берется из списка констант)*

***createDB****(номер базы, тип базы, вид базы, имя файла или базы) - перегруженный метод (для удобства создания Sqlite баз, где не требуются аргументы кроме имени файла)*

**openDB** – открытие базы данных

***openDB*** *(номер базы, имя базы в файле проекта)*

**closeDB** – закрытие базы данных.

***closeDB****(номер базы)*

**deleteDB** – удаление базы данных. В скрипте удаляются только базы данных, созданные этим скриптом

***deleteDB****(номер базы)*

**copyAll** – копирование всех данных

***copyAll*** *(номер базы источника, номер базы приемника ) - копируются все данные*

***copyAll*** *(номер базы источника, номер базы приемника, номер списка величин)*

**copyNew** – копирование данных после последней операции copyAll или copyNew. Время последной операции запоминается для каждой тройки “номер базы источника, номер базы приемника, номер списка величин”.

***copyNew*** *(номер базы источника, номер базы приемника) ; - копирование данных с даты последней операции copyAll или copyNew для всех наборов величин, добавленных методом dataAdd.*

***copyNew*** *(номер базы источника, номер базы приемника, номер списка величин) ;*

**copy** – копирование данных за заданный период времени.

**copyNew** (номер базы источника, номер базы приемника, номер списка величин, начало периода, конец периода) ;

### 11.13.3 Функции для работы с вводом-выводом

**dprint** – вывод сообщения в стандартный поток вывода (по умолчанию – в окно вывода программы).

В JavaScript print является функцией самого языка, поэтому следует различать print и ds.dprint.

***dprint****(текст);*

**dprintln** – вывод сообщения в стандартный поток вывода с переносом на новую строку(по умолчанию – в окно вывода программы).

***dprintln****(текст);*

**fopen** – открытие текстового файла

*int* ***fopen****(имя файла, тип=1); тип= 0-чтение, 1 - запись с начала файла, 2 – дописывать текст в конец файла.*

**fclose** – закрытие текстового файла

***fopen****(файл);*

**fprint** – вывод в текстовый файл

***fprint****(файл, текст);*

**fprintln** – вывод в текстовый файл с переносом на новую строку

***fprintln****(файл, текст);*

**freadln** – чтение следующей строки файла. Возвращает null, если достигнут конец файла.

*строка =* ***freadln****(файл )*

### 11.13.4 Функции для работы с журналом работы

**log** – вывод сообщения в лог и на экран

***log****(текст, тип вывода =1);*

*Тип вывода: 1- окно программы, 2- в файл, 3-на экран и в файл*

*Имя файла задается в файле конфигурации common.cfsj, свойство «LogFilePattern».*

*Имя по умолчанию – "script_log_%d.txt", где %d – порядковый номер файла лога, берется из файла конфигурации common.cfsj, свойство «LogFileNextNumber».*

**logFile** – возвращает абсолютный путь до файла лога, если в текущем выполнении скрипта вызывалась функция log с записью в файл, иначе возвращается null

*String* ***logFile****()*

### 11.13.5 Функции для работы с импортом данных

***importSkatLite****(файл Скат Лайт, номер хост-базы приемника);*

***importSkatPro****(хост, логин, пароль, проект Скат Про, номер хост-базы приемника);*

***importMsrFile****(имя файла, номер хост-базы приемника);*

### 11.13.6 Функции для работы с файлами CSV

**fcreateCSV** – создание файла CSV

*int* ***fcreateCSV****(имя файла, тип, заголовок)*

*тип – число:*

*1 - в файл записываются строки, соответствующие базе данных величин (время, SN датчика, SN регистратора, тип величины, номер величины, величина),*

*2 - в файл записываются строки с датой и величинами. Первая строка заголовка – список номеров датчиков. Вторая строка заголовка – список номеров регистраторов. Третья строка заголовка – список типов величин. Четвертая строка заголовка – список номеров величин. Эти строки формируются при первом вызове метода копирования (copyCSV, copyNewCSV, copyAllCSV) и в дальнейшем не изменяются.*

*заголовок - размещается до данных как комментарий*

**fcloseCSV** – закрытие файла.

**fcloseCSV**(номер файла)

**copyAllCSV** – копирование всех данных

***copyAllCSV*** *(номер базы источника, номер файла CSV приемника)*

***copyAllCSV*** *(номер базы источника, номер файла CSV приемника, номер списка величин)*

**copyNewCSV** – копирование данных после последней операции copyAll или copyNew. Время последней операции запоминается для каждой тройки “номер базы источника, номер базы приемника, номер списка величин”.

***copyNewCSV*** *(номер базы источника, номер файла CSV приемника) ; - копирование данных с даты последней операции copyAll или copyNew для всех наборов величин, добавленных методом dataAdd.*

***copyNewCSV*** *(номер базы источника, номер файла CSV приемника, номер списка величин) ;*

***copyCSV*** *– копирование данных за заданный период времени.*

***copyCSV*** *(номер базы источника, номер файла CSV приемника, номер списка величин, начало периода, конец периода) ;*

**setDatePrecisionCSV** - задать параметр совмещения для файла CSV. Если совмещение не задано, то оно по умолчанию = 1. Заданное значение совмещения используется в последующих вызовах методов copyCSV, copyNewCSV, copyAllCSV.

***setDatePrecisionCSV*** *(номер файла CSV, совмещение)*

*номер файла – полученный в результате выполнения ф-ции fcreateCSV*

*совмещение - область (временной интервал в секундах), в которой два значения измеряемых величин считаются единомоментными. Используется только в csv-файлах 2-го типа, действительное число.*

**setExportParamsCSV** – задать настройки экспорта в csv:

1. таблицу соответствия междуsn датчика и его именем для csv-файла. Если соответствие задано, то в csv-файле после серийного номера датчика будет идти его имя.
2. Формат даты-времени.
3. Базовые (нулевые) показания.
4. Признак – записывать в csv только показания, дата которых больше даты базового показания.
5. Признак – нумерация строк с показаниями

***setExportParamsCSV*** *(номер файла CSV, имя json –файла с настройками экспорта в csv)*

Пример файла.

***setToZeroValuesCSV*** *– приводить показания при записи в csv к нулю. По умолчанию к нулю не приводятся*

***setToZeroValuesCSV*** *(номер файла CSV, приводить к нулю = true/false)*

**Методы работы с величинами:**

(во всех методах по умолчанию тип величины равен «температура», номер величины равен 1.)

**date** – возвращает Java-дату. Может использоваться для получения аргумента-даты в методах **setvt, vt.**

***date****(год, месяц, день, час, минуты, секунды)*

**v** – значение величины в текущий момент времени. Берется значение, которое входит в интервал: [тек.время – 1 сек.; тек.время + 1 сек.]. Если значений несколько, то возвращается ближайшее к текущему времени показание.

*Величина =* ***v****(номер базы данных, серийный номер датчика или свойства, тип величины, номер величины)*

**setv** – установка значения величины в текущий момент времени.

***setv****(номер базы данных, значение, серийный номер датчика или свойства, тип величины, номер величины)*

**vp** – значение величины в заданном диапазоне от текущего момента времени: [тек.время – интервал; тек.время + интервал]. Интервал задается в секундах. Если значений несколько, то возвращается ближайшее к текущему времени показание.

*Величина =* ***vp****(номер базы данных, серийный номер датчика или свойства, тип величины, номер величины, интервал)*

**vt** – значение величины в заданный момент времени.

*Величина =* ***vt****(номер базы данных, время, серийный номер датчика или свойства, тип величины, номер величины)*

**setvt** – установка значения величины в заданный момент времени.

***setvt****(номер базы данных, время, значение, серийный номер датчика или свойства, тип величины, номер величины)*

**vmax** – максимальное значение величины за заданный период времени от текущего

*Величина =* ***vmax*** *(номер базы данных, период, серийный номер датчика или свойства, тип величины, номер величины)*

**vmin** – минимальное значение величины за заданный период времени от текущего

**vmed** – среднее значение величины за заданный период времени от текущего

**vmsd** – среднеквадратичное отклонение значений величины за заданный период времени от текущего

**values** – показания датчика за период

*массив показаний = values(номер БД, дата начала периода, дата окончания периода, серийный номер датчика или свойства, тип величины, номер величины)*

*Массив показаний = MeasureData[]*

*Объект MeasureData имеет следующие методы:*

*getMeasureDate() – дата показания*

*getMeasureValue() – значение*

*getMeasureType() – тип величины*

*getMeasureNo() – номер величины*

*getDeviceSN() – серийный номер датчика*

*getLoggerSN() – серийный номер регистратора*

### 11.13.7 Функции оповещения

**beep** – звуковой сигнал

***beep****(номер стандартного сигнала, длительность в секундах=1);*

*номер сигнала: 0 - \Sounds\warning.wav, 1 - \Sounds\Critical.wav*

**popup** – всплывающее окно с сообщением. Если окно с указанным номером не существует, то создается новое, иначе меняются настройки существующего окна.

***popup*** *(номер окна, заголовок окна, текст сообщения, цвет окна=0, время показа сообщения=0, размер шрифта = 14, время мигания = 0, цвет мигания=0);*

*цвет окна/мигания: 0 – стандартный цвет окна, 1-инфо зеленый, 2-предупреждение желтый, 3-ошибка красный.*

*текст сообщения: простой текст или html разметка, например, “<html><font size="14" color="red" face="Arial">Текст</font></html>”. (см. http://htmlbook.ru/html)*

*время мигания в секундах.*

**popupZone** – создать новую зону сообщений. Зона представляет собой сетку на экране монитора, в ячейке которой можно разместить сообщение. Размер ячеек = размеру создаваемых в зоне окон = указанному размеру окна.

***popupZone****(номер зоны, координата x левого верхнего угла, координата y левого верхнего угла, количество строк, количество столбцов, ширина окна=360, высота окна=180);*

**popupInZone** – всплывающее окно с сообщением. Если окно с указанным номером не существует, то создается новое, иначе меняются настройки существующего окна.

***popupInZone*** *(номер окна, номер зоны, номер строки, номер столбца, заголовок окна, текст сообщения, цвет окна=0, время показа сообщения=0, размер шрифта = 14, время мигания = 0, цвет мигания=0);*

*цвет окна/мигания: 0 – стандартный цвет окна, 1-инфо зеленый, 2-предупреждение желтый, 3-ошибка красный.*

*текст сообщения: простой текст или html разметка, например, “<html><font size="14" color="red" face="Arial">Текст</font></html>”. (см. http://htmlbook.ru/html)*

*время мигания в секундах.*

**popupCanvas** – всплывающее окно с подложкой для рисования. Если окно с указанным номером уже существует, то меняются его свойства.

***popupCanvas****(номер окна, заголовок окна, фон)*

***popupCanvas****(номер окна, заголовок окна, фон, номер зоны, номер строки, номер столбца)*

*фон может задаваться 2 способами:*

*числом: 0 – стандартный цвет окна, 1-инфо зеленый, 2-предупреждение желтый, 3-ошибка красный.*

*массивом из 3 чисел – цвет в формате RGB*

**popupDrawRect** – рисует прямоугольник в окне.

***popupDrawRect** (номер окна, x координата верхнего левого угла прямоугольника, y координата верхнего левого угла прямоугольника, ширина, высота, цвет границы, толщина линии границы=2, цвет заливки = цвет границы)*

*цвет задается 2 способами:*

*1 .числом: 0 – стандартный цвет окна, 1-инфо зеленый, 2-предупреждение желтый, 3-ошибка красный.*

*2. массивом из 3 чисел – цвет в формате RGB, либо массивом из 4х чисел – цвет в формате RGB + alpha (число от 0 до 255, задающее прозрачность)*

**popupDrawEllipse**– рисует эллипс в окне.

***popupDrawEllipse** (номер окна, x координата верхнего левого угла описанного прямоугольника, y координата верхнего левого угла описанного прямоугольника, ширина, высота, цвет границы, толщина линии границы=2, цвет заливки = цвет границы)*

*цвет задается как в методе popupDrawRect.*

**popupDrawLine**– рисует отрезок в окне.

***popupDrawLine** (номер окна, x, y координаты начала отрезка, x, y координаты конца отрезка, цвет границы, толщина линии границы=2)*

*цвет задается как в методе popupDrawRect.*

**popupDrawPolygon**– рисует многоугольник в окне.

***popupDrawPolygon** (номер окна, массив x координат вершин, массив y координат вершин, цвет границы, толщина линии границы=2, цвет заливки = цвет границы)*

*цвет задается как в методе popupDrawRect.*

**popupDrawText** – размещает текст в окне.

***popupDrawText**(номер окна, текст, x, y координаты левого нижнего угла описанного прямоугольника, цвет=черный, размер шрифта=14)*

*цвет задается 2 способами так же, как и в popupCanvas.*

**popupDrawHint** – размещение подсказки в окне.

***popupDrawHint**(номер окна, текст подсказки, x, y координаты точки, к которой привязана подсказка, x, y координаты точки левого верхнего угла подсказки, ширина, высота, цвет границы, толщина линии границы = 2, цвет заливки = цвет границы, отступ текста от границы = 5)*

*текст подсказки может быть простым текстом, либо html, например, <html><font size="14" color="red" face="Arial">Текст</font></html>.*

*цвет задается 2 способами так же, как и в popupDrawShape.*

**popupCanvasClear** – очищает окно, удаляя все нарисованные на подложке фигуры.

***popupCanvasClear()***

**sound** – звуковое сообщение

***sound** (имя файла, громкость=5, повтор=1);*

***sound** (массив имен файлов, громкость=5, повтор=1);*

**soundText** – звуковое сообщение

***soundText*** *(текст, громкость=5, повтор=1);*

**saveTextAsSound** – строка текста сохраняется как файл для последующего воспроизведения.

***saveTextAsSound*** *(текст, имя файла, папка);*

**sms** – отправка СМС

***sms****(номер телефона, текст сообщения);*

*Текст сообщения ограничен 70 символами. Если символов больше 70, то при включенной генерации исключений (throwExceptions=0) генерируется ScriptException и смс не отправляется, при выключенной (throwExceptions=1) – отправляется только 70 первых символов, исключение не генерируется, но запись об ошибке (текст смс превышает допустимую длину) записывается в лог.*

*Отправка осуществляется через GSM модем, подключенный к ПК по usb. Если модемов подключено несколько, то смс отправляется через первый обнаруженный модем.*

*Настройки подключения к GSM модему определяются в файле настроек приложения. Значения по умолчанию:*

*GSMBaudrate (скорость передачи для порта) = 9600*

*GSMDatabits (стандартное число битов данных в байте) = 8*

*GSMStopbits (число стоповых битов) = 1*

*GSMParity (протокол контроля четности) – четность не проверяется.*

**email** – отправка электронной почты. Текст сообщения может быть простым текстом или HTML, может быть задан явно в виде строки, либо указан номер файла, открытого для чтения, с текстом сообщения (номер файла возвращается как результат метода fopen). Количество прикрепляемых файлов ограничено лишь максимальным размером электронного письма.

*Html текст должен начинаться с тега <html>, иначе он будет восприниматься как простой текст.*

*Для вставки изображений в тело письма в тексте html нужно указать в качестве источника картинки file+номер файла, например file2. Файлы нумеруются с 1.*

*Пример текста html (см. справка по html: http://htmlbook.ru/html).*

*<html>*

*Текст письма с картинкой*

*<br>*

*<img src=”cid:file1”>*

*</html>*

*Файлы, не указанные в тексте html, прикрепляются к письму как вложение.*

***email*** *(адрес почты, текст темы, текст сообщения или номер файла с текстом сообщения, имя прикрепляемого файла1=””, файла2=””, файла3=””, файла4=””, файла5=””, …);*

**query** – всплывающее окно с запросом, возвращает результат ответа (0-нет, 1-да, -1 нет ответа)

*int* ***query*** *(номер окна, текст сообщения, цвет окна=0, время показа сообщения=60, время мигания=0, цвет мигания=0);*

*текст сообщения: простой текст или html разметка, например, “<html><font size="14" color="red" face="Arial">Текст</font></html>”. (см. http://htmlbook.ru/html)*

*цвет окна/мигания: 0 – стандартный цвет окна, 1-инфо зеленый, 2-предупреждение желтый, 3-ошибка красный.*

### 11.13.8 Функции для чтения и записи xml и json

**Определения**

**Документ** – набор структурированных данных. Документ состоит из одного элемента, называемого корневым.

Объект документ является экземпляром класса Document.

Методы:

**Element getRoot()** – возвращает корневой элемент документа.

**setRoot(element)** – устанавливает элемент в качестве корневого

**Элемент** – структура данных. Каждый элемент может включать пары ключ-значение и должен содержать либо 1 значение простого типа (число/текст) – простой элемент, либо несколько дочерних элементов – составной элемент.

Объект элемент является экземпляром класса SimpleElement или ComplexElement, наследующихся от базового класса Element.

Общие методы классов:

**getName()** – возвращает имя элемента

**setName (имя)** – устанавливает имя элемента

**setKeyValue**(ключ, значение)

**getKeyValue(ключ**) – возвращает значение по ключу

**hasChildElements()** – имеет ли элемент дочерние (true/false)

Методы простых элементов:

**setValue (значение)** - устанавливает значение элемента

getValue() – возвращает значение элемента

Методы составных элементов:

**addChild(элемент)** – добавить дочерний элемент

**getChildByName(имя)** – возвращает дочерний элемент по имени

**createDoc** – создание документа.

*документ **createDoc** ()*

**createElement**– создание элемента.

***createElement(имя, тип)***

*тип: 1 – простой элемент; 2 – с дочерними элементами*

**writeFile –** запись файла заданного типа

***writeFile(документ, имя файла, тип)***

*тип : 1=XML, 2=JSON*

**readFile –** чтение файла и разбор его структуры

*документ **readFile(имя файла, тип)***

**Пример записи в файл.**

```
var doc = createDoc();
var root = createElement("root", 2);
	root.setAttrValue("attr1", "Текст");
		root.setAttrValue("attr2", 5.46);
		var child1 = createElement("child1", 1);
		child1.setAttrValue("attr1", 56);
		child1.setValue("Content string.");
		root.addChild(child1);
		doc.setRoot(root);
	//запись в xml
writeFile(doc, “C:\\doc.xml", 1);
//запись в json
writeFile(doc, " C:\\doc.json", 2);
```

Результат

| XML | JSON |
|---|---|
| <?xml version="1.0" encoding="UTF-8" standalone="no"?><br><root attr1="Текст" attr2="5.46"><br><child1 attr1="Attribute 1 of child1"><br>Content string.<br></child1><br></root> | {<br>"attr1" : "Текст",<br>"attr2" : 5.46,<br>"child1" : {<br>"attr1" : "Attribute 1 of child1",<br>"Content" : "Content string."<br>}<br>} |

**Пример чтения из файла, изменения прочитанного документа и запись в новый файл.**

//читаем xml

```
	var docXml = readFile("C:\\doc.xml", 1);
	//корневой элемент документа
	var root = docXml.getRoot();
	//изменяем значение существующего ключа/добавляем новую пару ключ-значение
	root.setKeyValue("attr1", "111");
	root.setKeyValue("attr3", "atribute 3");
	//поиск дочернего элемента по имени
	var child = root.getChildByName("child1");
	if(child!==null)
	{
		child.setValue("NEW VALUE!");
	}
	//запись в файл
	writeFile(docXml, "C:\\doc_copy.xml", 1);
```

Для поддержки функций задания данных в соответствии с Техническими требованиями МЧС для серверов СМИС/СМИК, пишется обертка с обобщающими функциями.

### 11.13.9 Функции для команд управления

**waitSec** – приостановить работу скрипта на указанное количество секунд.

***waitSec****(кол-во секунд)*

// Методы будут уточнены позже

### 11.13.10 Функции для работы с FTP

**openFTP –** открытие FTP соединения. Фактически настройки ftp только запоминаются, а соединение открывается/закрывается непосредственно в методах downloadFTP и uploadFTP.

*Int* ***openFTP*** *(хост, логин, пароль, порт=21)*

**closeFTP –** окончание FTP соединения. Удаляет настройки ftp из памяти, для этого соединения нельзя будет вызывать downloadFTP и uploadFTP.

***closeFTP*** *(соединение)*

**uploadFTP** – загрузка на FTP сервер

***uploadFTP*** *(соединение, имя загружаемого файла );*

***uploadFTP*** *(соединение, имя загружаемого файла, папка загрузки); - загрузка в определенную папку ftp сервера (должна существовать!)*

**downloadFTP** – загрузка с FTP сервер

***downloadFTP*** *( соединение, имя загружаемого файла, папка сохранения="");*

### 11.13.11 Функции идентификации

**stamp** – вывод строки с идентификацией

*string* ***stamp****(тип="12380")*

*тип – строка из символов X:*

*X:*

*0 - время*

*1 - код– четырехзначное число – хэшфункция серийных номеров и контрольной суммы скрипта*

*2 –серийный номер экземпляра программы*

*3- описание экземпляра программы (серийный номер, тип, название, версия)*

*4 – детальное описание экземпляра программы (комментарий,)*

*5-серийный номер компьютера*

*6-описание компьютера (серийный номер, тип, название)*

*7 – детальное описание компьютера (комментарий, данные –система, Процессор, частота, память, Диск и т.п.)*

*8 – контрольная сумма скрипта (включая ссылочные файлы Header)*

*9 –описание скрипта(серийный номер(если есть), название файла)*

*A –детальное описание скрипта (версия, комментарий, дата и т.п)*

*B – IP адрес (если возможно)*

*C - местоположение IP адреса*

*D – координаты*

*E – адрес (по координатам)*

*F – точность координат, источник координат (GPS, A-GPS и т.п.)*

### 11.13.12 Функции построения с графиков

**createChart** - создать график.

*int номер графика = **createChart** (тип, имена осей ординат)*

*тип - 1- x-ось времени, 2 - x-ось с числовыми значениями*

*Всем осям У присваиваются номера согласно этому массиву имен. Номер оси = индекс в массиве.*

**createLine** – создать линию

*int номер линии= **createLine** (номер графика, номер оси У, Имя линии, цвет в формате RGB = [0,0,0], толщина линии=2)*

*Имя линии отображается в легенде.*

**addDoublePoint** - добавить точку на график. Точка вида: (число, число)

***addDoublePoint** (номер графика, номер оси У, номер линии, x,y)*

**addTimePoint** - добавить точку на график. Точка вида: (дата, число)

***addTimePoint** (номер графика, номер оси У, номер линии, x,y)*

**exportChartPNG** - Экспорт графика в формате PNG

***exportChartPNG** (номер графика, имя файла)*

**exportChartSVG** - Экспорт графика в формате SVG

***exportChartSVG** (номер графика, имя файла)*

### 11.13.13 Функции построения с 3D графиков

**createChart3D** – открывает окно с 3D графиком

*int номер графика = **createChart3D**(тип, заголовок окна)*

*тип: 1-3D поверхность, 2-2D контуры*

**deleteChart3D** – удалить график из памяти и закрыть окно с графиком

***deleteChart3D**(номер графика )*

**clearChart3D** – очистить график

***clearChart3D**(номер графика)*

**addSurface** – добавить точки поверхности

***addSurface**(номер графика, массив точек)*

*массив точек - массив координат точек: [[x,y,z], [x,y,z], ...]*

### 11.13.14 Функции для работы с ошибками

**error** – возвращает код ошибки из списка кодов ошибок по индексу или -1, если указан неверный индекс или список пуст, 0 – индекс первого элемента списка.

***int error(**индекс элемента)*

*Коды ошибок перечислены в файле JSON в общей папке программ Драйв: C:\Program Files\Sitis\SharedData\log.json*

*Список ошибок очищается каждый раз перед выполнением метода объекта ds. Таким образом, в списке хранятся ошибки выполнения предыдущего метода ds.*

**errorLast** – возвращает код последней ошибки или -1, если ошибок нет.

*int errorLast()*

**errorNum** – возвращает количество ошибок указанного типа.

***int errorNum****(тип ошибки)*

| Тип | Диапазон кодов ошибок | Описание |
|---|---|---|
| 1 | 1000-1999 | Ошибки при работе с файловой системой: открытие, сохранение, ввод-вывод и т.д. |
| 2 | 2000-2999 | Ошибки при работе с базами данных. |
| 3 | 3000-3999 | Системные ошибки. |
| 4 | 4000-4999 | Ошибки работы пакета программ Драйв: недопустимое значение аргумента функции, ошибка при инициализации объекта и т.д. |
| 5 | 5000-5999 | Ошибки в объектах dv, ds при выполнении скрипта. |

**errorClear** – удалить из списка ошибки указанного типа.

***errorClear****(тип ошибки)*

**Свойства**

**logType** – тип вывода в лог: 1-в окно программы; 2-в файл; 3-в окно программы и в файл. Значение по умолчанию 1.

**errorType** – тип ошибок, выводимых в лог: 0 - все, 1- синтаксические и ошибки времени исполнения скрипта, 2 – остальные ошибки. Значение по умолчанию 0.

**throwExceptions** - генерация исключений: 0 - всех, 1- отключена. Значение по умолчанию 1.

**Пример использования методов работы с ошибками.**

```
ds.dataAdd(0, 100, 4001, 1);
if(ds.errorLast()>0) {
    println("Ошибка с кодом: " + ds.errorLast());
    println("Всего ошибок: " + ds.errorNum(0));
}
```

**Исключения**

Если установлено значение свойства **throwExceptions** равным 0, то выполнение методов ds может генерировать исключения типа drv.ScriptException. Чтобы обработать исключение необходимо использовать конструкцию try-catch. У объекта типа drv.ScriptException можно получить код ошибки – метод getCode().

**Пример для Java 7:**

```
try{
    ds.dataAdd(0, 1290, 3002, 2);
}
catch(exc){
    var scriptException = exc.javaException;
    if(scriptException !== 'undefined'){
          // исключение Java
          var code = scriptException.getCode();
println("Произошла ошибка! Код ошибки: " + code);
}
```

```
else{
    //исключение JavaScript
}
}
Пример для Java 8:
try{
    ds.dataAdd(0, 1290, 3002, 2);
}
catch(exc){
    if(exc instanceof Java.type("java.lang.Exception")){
           // исключение Java
           var code = exc.getCode();
println("Произошла ошибка! Код ошибки: " + code);
}
else{
    //исключение JavaScript
}
}
```

Класс drv.ScriptException имеет наследников, представляющих более узкие классы исключений. Их можно перехватывать в конструкции try-catch аналогичным способом.

**drv.NullDataSetException** – исключение, генерируемое при null значении набора датчиков и величин в методах работы с наборами.

*Методы: getDataSetNum() – возвращает номер набора данных.*

**drv.DataSetNotFoundException** – генерируется, когда набор датчиков и величин с определенным номером не найден, в методах работы с наборами.

*Методы: getDataSetNum() – возвращает номер набора данных.*

**drv.DatabaseException** – исключение, генерируемое в методах работы с базами данных: открытии, создании, удалении. Коды ошибок с 5010 по 5019.

*Методы: getDatabaseNum() – возвращает номер базы данных.*

**drv.CopyDatabaseException** – исключение, генерируемое в методах копирования данных из одной базы в другую. Коды ошибок с 5020 по 5030.

*Методы: getSourceDbNum() – возвращает номер базы-источника;*

*getReceiverDbNum() – возвращает номер базы-приемника.*

**drv.SoundException** – ошибки при воспроизведении звука в методах beep и sound. Коды ошибок с 5038 по 5044.

**drv.IOException** – ошибки ввода-вывода в методах fopen, fprint, fclose. Коды ошибок с 5049 по 5056.

**drv.CSVException** – ошибки в методах работы с файлами CSV. Коды ошибок с 5062 по 5073.

*Методы: getCsvFile() – возвращает имя файла CSV*

*getFileNum() – возвращает номер файла CSV*

**drv.CopyCSVException** – ошибки в методах работы с файлами CSV. Наследует класс **drv.CSVException**. Коды ошибок с 5068 по 5073.

*Методы: getCsvFile() – возвращает имя файла CSV приемника (метод базового класса drv.CSVException)*

*getFileNum() – возвращает номер файла CSV*

*getSourceDbNum() – возвращает номер базы-источника.*

**drv.FTPException** – ошибки методов FTP. Коды ошибок с 5081 по 5084.

*Методы: getFtpNum() – возвращает номер ftp-соединения.*

**Пример для Java 7:**

```
ds.throwExceptions=0;
try{
    ds.dataAdd(16, 1290, 3002, 2);
}
catch(exc){
    var scriptException = exc.javaException;
    if(scriptException !== 'undefined'){
           var code = scriptException.getCode();
           println("Произошла ошибка! Код ошибки: "+code);
           if(scriptException.getClass().getName() == "drv.DataSetNotFoundException"){
                  println("Набор не найден: "+ scriptException.getDataSetNum());
           }
    }
}
```

## 12. КОМАНДЫ ДЛЯ УПРАВЛЕНИЯ ПРОГРАММОЙ В КОНСОЛЬНОМ РЕЖИМЕ

**help** – вывести справку о командах консольного режима работы программы

**open** – открыть соединение с com-портом

*open No baudRate dataBits stopBits Parity*

*No – номер открываемого порта, baudRate – скорость соединения, dataBits – количество бит данных (от 4 до 8), stopBits – количество стоп бит (1,2 или 3(эквивалентно 1.5) Parity – контроль четности, от 0 до 4, соответствует None,Odd,Even,Mark,Space)*

*Пример open 3 115200 8 1 0 – открыть 3й порт со скоростью 115200 бод, количество бит данных 8, 1стоп-бит, без контроля четности.*

**aboutCom** – вывести на консоль информацию о соединении с портом

**cmd** - отправить текстовую команду, добавив к ней LfCr.

*cmd text*

**hex** – отправить шестнадцатиричные данные

*hex data*

**text** - отправить текстовую команду, не добавляя LfCr.

*cmd text*

**script** – запустить на выполнение скрипт из файла

*script file_path [delay]*

**file_path** – путь к файлу со скриптом, delay – задержка (в миллисекундах). Если значение задержки не задано (команда вызвана с одним параметром, который будет интерпретирован, как путь к файлу), скрипт запустится в режиме однократного выполнения. Если значение задержки задано, скрипт будет запущен в цикле с соответствующей задержкой между итерациями.

**macro** – запустить макрос на исполнение

*macro file_path [delay]*

**file_path** – путь к файлу с макросом, delay – задержка (в миллисекундах). Если значение задержки не задано (команда вызвана с одним параметром, который будет интерпретирован, как путь к файлу), макрос запуститься в режиме однократного выполнения. Если значение задержки задано, макрос будет запущен в цикле с соответствующей задержкой между итерациями.

*Внимание! При выполнении команд macro или script при открытом существующем проекте, значения, введенные пользователем, будут помещены в первый объект проекта.*

**stop** – остановить выполнение цикла макроса или скрипта, соответствующего номеру объекта. Без параметров останавливает выполнение первого цикла на первом объекте.

**run** - при открытом файле проекта отправить на исполнение объект (макрос или скрипт) из проекта по номеру.

*run 1*

**close** – закрыть соединение с портом

**exit** – выйти из программы.

## 13. ОПИСАНИЕ ФАЙЛА ПРОЕКТА ПРОГРАММЫ

В файле проекта «Драйв: Терминал» содержатся настройки подключения к com-порту, активные объекты программы (макросы или скрипты), а также некоторые интерфейсные настройки, касающиеся вывода получаемой информации в окне терминала.

Файл проекта представляет собой текстовый файл в формате json, с расширением «.dtrmj», содержащий следующие поля:

- Имя программы "Name"
- Номер com-порта "ComPortNo"
- Скорость соединения с com-портом "ComPortBaudRate"
- Количество бит данных com-порта "ComPortDataBits" (значение 4-8)
- Количество стоп бит для com-порта "ComPortStopBits" (значение 1-3, 3 соответсвует 1.5 стоп-бит)
- Контроль четности com-порта "ComPortParity" (значение 0-4)
- Объекты (макрос или скрипт) Object1, Object2, Object3, Object4
- Директория для сохранения файлов по умолчанию "SavedFilesDirectory"
- Добавление перевода каретки и переноса строки при отправке команд "AddLfCr"
- Отправка данных в текстовом режиме "SendByteTextMode"
- Режим отображения эха команд "CommandEchoShow"
- Цвет отображения эха команд "CommandEchoColor"
- Цвет отображения полученных данных "ReadDataColor"
- Цвет печати ответа скриптов "ScriptOutputColor"
- Режим отображения (обычные, большие и огромные кнопки) "ShowBigFormMode"
- Путь к внешнему редактору объектов "MacrosAndScriptEditorPath"

## 13.1. Понятие объекта в проекте программы

Объект проекта «Драйв: Терминал» представляем самостоятельным объектом, который может содержать в себе текст макроса или скрипта, связанного с соответствующим элементом интерфейса программы. Объект содержит поля: имя объекта, текст, значение задержки между итерациями, цикличность, признак, нужно ли интерпретировать текст как скрипт или как макрос.

Пример объекта, содержащего в себе макрос, который периодически с задержкой в 2 секунды отправляет устройству команду «ati»:

```
"Object1" : {
  "TypeName" : "CMacrosScriptObjectTemplate",
  "CycleDelay" : 2000,
  "IsCycle" : true,
  "Text" : "ati",
  "IsSctipt" : false,
  "Name" : "null",
  "ObjectName" : "ATI"
},
```

## 13.2. Проект по умолчанию

По умолчанию, создается подключение к -1 (не существующему) порту, на скорости 115200 бод, с 8 битами данных, 1 стоп битом, без контроля четности. Настройки подключения по умолчанию содержатся в файле «defaultComSettings.json».

Все объекты по умолчанию – это пустые макросы.

Папка для сохранения файлов – папка «files» в директории установки программы.

Внешний редактор не используется.

Эхо команд включено и отображается синим цветом, перевод каретки добавляется, выбран текстовый режим отправки команд.

При создании подключения и редактировании объектов, изменения сохраняются в текущем проекте. После сохранения проекта его можно использовать при последующих запусках программы.