Спецификация на формат XML-файла описания классов устройств для клиента k095

1. Первая строка

```
<?xml version="1.0" standalone="yes" ?>
```

2. Корневой тег документа

classlist

3. Описание тега classlist

Состоит из произвольного количества тегов **class**.

3.1 Описание тега class

Описывает классы устройств. Состоит из произвольного количества тегов **param** и **vparam**.

Обязательный строковый атрибут **name** содержит уникальное имя класса устройств.

Например:

```
<class name="rpsw">
...
</class>
```

3.1.1 Описание тега param

Тег **param** описывает параметр класса устройств.

Атрибуты тега **param**: **name**, **type**, **dim, basename**, **bit.**

Тег **param** является контейнером для следующих тегов:
human_name, **info**, **minvalue**, **maxvalue**, **defvalue**, **variants**, **access**.

Строковый обязательный атрибут **name** служит для указания уникального имени параметра.

Константно строковый атрибут **type** служит для указания типа параметра. Возможны следующие варианты задания типа:
UINT — беззнаковое 4-х байтовое целое значение (тип используется по умолчанию)
INT — знаковое 4-х байтовое целое значение
FLOAT — вещественное 4-х байтовое значение с плавающей запятой.
ASCIIZ — строковый параметр из однобайтовых символов, завершающийся кодом 0-ля.
BYTE_ARRAY — бинарные данные (при передаче кодируются строкой шестнадцатеричных кодов от 00 до FF по значению каждого байта данных)

Чистовой необязательный атрибут **dim** служит для указания кратности данного параметра. Фактически, это можно сравнить с объявлением массива в языках программирования.

Например, следующее описание создаст 16 параметров с именами от **i_plus_0** до **i_plus_15**:

```
<param name="i_plus" dim="16">
	<human_name>+I</human_name>
	<info>Ток в канале. Положительное плечо</info>
	<minvalue>0</minvalue>
	<maxvalue>4095</maxvalue>
	<access>R</access>
</param>
```

Строковый необязательный атрибут **basename** служит для указания некоторого базового параметра от значения которого зависит текущий параметр. Это делается в случае, если текущий параметр вычисляется как некоторый бит в значении базового параметра. Номер бита задается атрибутом **bit**.

Ниже приведены примеры описания двух параметров. Сначала описан параметр который является базовым для целого массива зависящих от него параметров.

```
<param name="switch">
	<human_name>Ключи</human_name>
	<info>Ключи по 16 каналам. "0" - выключен, "1" - включен</info>
	<minvalue>0</minvalue>
	<maxvalue>65535</maxvalue>
	<access>RW</access>
</param>
<param name="switch" basename="switch" bit="0" dim="16">
	<human_name>Ключ</human_name>
	<info>Ключ канала</info>
	<variants>0:OFF, 1:ON</variants>
	<defvalue>0</defvalue>
	<access>RW</access>
</param>
```

В данном примере описан базовый параметр **switch**, который состоит из 16 битов, каждый из которых соответствует состояниям соответствующих ключей. Для удобства работы, далее описан массив из 16 битовых параметров **switch_0** .. **switch_15** для которых параметр **switch** является базовым (см. описание атрибута **dim**). Указание атрибута **bit** задает битовую природу массива параметров. В этом случае, значение атрибута **bit** последовательно инкрементируется для всего массива параметров начиная с исходного значения атрибута. Таким образом получаем, что параметр **switch_0** отражает состояние 0-го бита базового параметра **switch**, а параметр **switch_15** — состояние 15-го бита базового параметра **switch**.

3.1.1.1 Тег human_name

Строковый необязательный тег human_name служит для указания человечески благозвучного имени параметра.

Например:

```
<human_name>Счетчик КЗ</human_name>
```

3.1.1.2 Тег info

Строковый необязательный тег **info** служит для размещения подробной информации о параметре. Фактически это строковый комментарий произвольной длины.

3.1.1.3 Тег minvalue

Числовой необязательный тег **minvalue** служит для указания минимального значения для числовых параметров имеющих соответствующее ограничение.

Например:
<**minvalue**>0</**minvalue**>

3.1.1.4 Тег maxvalue

Числовой необязательный тег **maxvalue** служит для указания максимального значения для числовых параметров имеющих соответствующее ограничение.

Например:
<**maxvalue**>0</**maxvalue**>

3.1.1.5 Тег defvalue

Необязательный тег **defvalue** служит для задания значения по умолчанию. Может использоваться для параметров любых типов. В случае, если следует задать значение для параметра типа **BYTE_ARRAY**, значение задается в виде строки из последовательности шестнадцатеричных кодов от 00 до FF для каждого байта данных.

3.1.1.6 Тег variants

Необязательный строковый тег variants позволяет перечислить строковые замены для конечного числа целых значений. Данные замены будут использоваться в табличном представлении параметров и могут быть использованы в соответствующих терминалах.
<**variants**>0:OFF, 1:ON</**variants**>

3.1.1.7 Тег access

Необязательный константно-строковый тег **access** служит для указания уровня доступа к параметру. Возможны следующие значения тега:
R — только чтение
W — только запись
RW — чтение и запись

3.1.2 Описание тега vparam

С помощь тега **vparam** описывает виртуальный параметр класса устройств. Виртуальные параметры не существуют реально, а образуются через арифметико-логические вычисления над значениями других параметров в скрипте на языке javaScript 2.0 (ECMA).

Атрибуты тега **vparam**: **name**.

Обязательный строковый атрибут **name** определяет имя виртуального параметра.

Тег **vparam** является контейнером для тегов: **arg**, **script**.

Например:

```
<vparam name="alarm">
```

```
…
</vparam>
```

Следует различать виртуальные параметры двух типов: стандартные и пользовательские. Стандартные виртуальные параметры существуют для каждого устройства независимо от того, объявляли вы их или нет. Пользовательские виртуальные устройства будут существовать в устройстве только в случае их явного объявления.

На данный момент существует только один стандартный виртуальный параметр — параметр "**alarm**". По умолчанию, если пользователь не описал поведение данного параметра для устройства, он всегда будет возвращать значение 0 (false) — отсутствие аварии. Если же реализовать поддержку данного параметра явно, то можно будет определить ситуацию аварии опираясь в скрипте на значения других параметров, по которым аварийную ситуацию можно определить. Аварией будет считаться любое ненулевое значение данного параметра.

Стандартные параметры используются в различных унифицированных схемах обработки устройств, например, в многоузловых терминалах.

Пример полного тега **vparam**:

```
<vparam name="alarm">
	<arg id="h" param="overheat"/>
	<arg id="c" param="overcurrent"/>
	<arg id="g" param="leakage"/>
	<arg id="s" param="short_circuit"/>
	<script>
		return h || c || g || s;
	</script>
</vparam>
```

3.1.2.1 Тег arg

Тег **arg** служит для указания аргументов, передаваемых в скрипт для вычисления значения виртуального параметра.

Тег arg имеет два обязательных строковых атрибута: id и param. Тег id вводит идентификатор, который будет использован в скрипте для получения значения параметра, задаваемого атрибутом param. Значение атрибута id должно соответствовать правилам обозначения идентификаторов в языке javaScript.

Например:

```
<arg id="g" param="leakage"/>
```

Теперь скрипт сможет ссылаться на значение параметра **leakage** через идентификатор **g**.

Тег arg не является обязательным, однако в этом случае, скрипт сможет возвратить только некоторое константное значение.

Количество тегов **arg** в контейнере **vparam** может быть произвольным. Все теги **arg** должны предшествовать тегу **script**.

3.1.2.2 Тег script

Тег **script** используется для записи тела функции вычисляемой значение виртуального параметра через переданные в нее аргументы, описанные в тегах **arg**. Код функции пишется

на языке javaScript. Результат вычисления функции передается через оператор **return**.

Например:

```
<script>
      return h || c || g || s;
</script>
```

Тег **script** является обязательным тегом в контейнере **vparam**. Не допускается использование нескольких тегов **script** внутри одного контейнера.

Теги **arg**, описывающие аргументы, передаваемые в скрипт, должны предшествовать тегу **script**.

Пример файла описания класса устройств

```
<?xml version="1.0" standalone="yes" ?>
<classlist>
      <class name="TEST1" interface="libk095_test_library.so">

            <param name="uint_prm">
                  <human_name>Param 1</human_name>
                  <info>Первый параметр примера. Без статистики. Поддерживает чтение и
запись.</info>
                  <access>RW</access>
                  <minvalue>0</minvalue>
                  <maxvalue>500</maxvalue>
                  <defvalue>100</defvalue>
            </param>

            <param name="uint_prm_st">
                  <human_name>Param 2</human_name>
                  <info>Второй параметр примера. Параметр содержит данные статистики.
Поддерживает только чтение.</info>
                  <access>R</access>
            </param>

            <param name="prm_int" type="INT">
                  <human_name>Int Param</human_name>
                  <info>Параметр целого типа. Поддерживает чтение и запись.</info>
                  <access>RW</access>
                  <minvalue>-30</minvalue>
                  <maxvalue>50</maxvalue>
                  <defvalue>-10</defvalue>
            </param>

            <param name="float_prm" type="FLOAT">
                  <human_name>Float Param</human_name>
                  <info>Параметр вещественного типа. Поддерживает чтение и
запись.</info>
                  <access>RW</access>
                  <minvalue>-1.5</minvalue>
                  <maxvalue>5.8</maxvalue>
                  <defvalue>0.3</defvalue>
            </param>

            <param name="asciiz_prm" type="ASCIIZ">
                  <human_name>ASCIIZ Param</human_name>
                  <info>Параметр строкового ASCIIZ типа. Поддерживает чтение и
запись.</info>
                  <access>RW</access>
                  <defvalue>Hello world</defvalue>
            </param>

            <param name="byte_array_prm" type="BYTE_ARRAY">
                  <human_name>Byte_Array Param</human_name>
                  <info>Параметр строкового BYTE_ARRAY типа. Поддерживает чтение и
```

```
запись.</info>
                    <access>RW</access>
                    <defvalue>AFF055AA9B</defvalue>
            </param>
        </class>
</classlist>
```