

Программный комплекс АСНИКА-К

**Система АСНИКА-К-РЭС**

( расчёт показателей надежности «структурно-сложных» электронных  
средств )

**Описание языка**

**RU.17701729.22005-01 35**

**(на CD–дисках)**

Листов 21

2023

Литера

Инв. N подл.	Подл. и дата	Взам. инв. N	Инв. N дубл.	Подл. и дата

## АННОТАЦИЯ

Система анализа надёжности реконфигурируемых изделий АСОНИКА-К-РЭС предназначена для расчётов показателей надёжности реконфигурируемых электронных средств (электронных средств, при отказе составных частей которых восстановление работоспособности осуществляется путём реконфигурации исходной структуры) по данным о характеристиках надёжности составных частей и об алгоритмах реконфигураций. Система АСОНИКА-РЭС может эксплуатироваться как автономно, так и в составе программного комплекса АСОНИКА-К, что позволяет существенно снизить время расчётов за счёт использования интенсивностей отказов составных частей, полученных с помощью системы АСОНИКА-К-СЧ.

Система АСОНИКА-К-РЭС реализует метод имитационного моделирования, что позволяет проводить расчёты надёжности электронных средств, схема расчета надёжности которых может содержать алгоритмы реконфигурации, «неприводимые» графы и комплекты ЗИП. Это достигается за счёт встроенного специализированного языка, на котором формируется описание алгоритмов реконфигураций при отказах составных частей.

Система АСОНИКА-К-РЭС позволяет поддерживать практически неограниченное количество компонентов схем расчёта надёжности электронных средств, число которых определяется только техническими характеристиками ЭВМ, на которой установлена система.

В руководстве приведено описание языка для расчетов надёжности «структурно-сложных» электронных.

Руководство содержит 21 л.

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

## СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ.....	4
2. ЭЛЕМЕНТЫ ЯЗЫКА.....	4
2.1. Описание синтаксиса базовых элементов языка .....	4
2.2. Описание семантики базовых элементов языка .....	7
2.3. Описание синтаксиса составных элементов языка .....	8
2.4. Описание семантики составных элементов языка .....	11
3. СПОСОБЫ СТРУКТУРИРОВАНИЯ ПРОГРАММЫ.....	15
4. СРЕДСТВА ОБМЕНА ДАННЫМИ .....	17
5. ВСТРОЕННЫЕ ЭЛЕМЕНТЫ.....	17
6. СРЕДСТВА ОТЛАДКИ ПРОГРАММЫ.....	18
СПИСОК ЛИТЕРАТУРЫ.....	19
Лист регистрации изменений.....	21

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

## 1. ОБЩИЕ СВЕДЕНИЯ

1.1. Назначение языка: Имитационное моделирование процесса отказов и реконфигураций электронных средств (ЭС).

1.2. Основные характеристики языка: Содержит встроенные модели для компонентов ЭС и структурирован с учетом специфики моделируемых событий

1.3. Основные возможности языка: Позволяет моделировать произвольные законы распределения, построить имитационную модель ЭС с различными законами реконфигурации, верифицировать модель и провести серию экспериментов и получить статистические данные о надежности моделируемой системы.

1.4. Область применения языка: Исследования надежности резервированных и реконфигурированных ЭС.

## 2. ЭЛЕМЕНТЫ ЯЗЫКА

### 2.1. Описание синтаксиса базовых элементов языка

<word> - слово, последовательность символов, должна начинаться с латинской буквы или знаков «\_», «\$» и может содержать латинские буквы, цифры и знаки «\_», «\$». В ключевых словах языка регистр не учитывается, в остальных случаях имеет значение.

<decimal> - число, может быть, как целым, так и десятичным (десятичный разделитель - «,») формы записи числа на примере:

целое -10;

десятичное - 10,0 ; , 10;

экспоненциальное - 1e1 ; 2,1e2 ; ,3e2;

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

boolean - логическая переменная, может принимать значения строго 0 или 1.

variable <Name>; - объявление переменной, Name - имя.

<Name> = <decimal> - присваивание переменной Name значения decimal.

Var1 = Var2+<decimal>; - присваивание значения переменной и использование переменной в выражении.

distribution - модель распределения.

Distribution <Name> (<param>); - объявление распределения с именем Name и параметрами param.

function - последовательность операторов, возвращает значение типа variable.

Логические операторы:

- «!» - логическое отрицание;
- «>» - больше;
- «<» - меньше;
- «>=» - больше или равно;
- «<=» - меньше или равно;
- «|» - логическое «или»;
- «&» - логическое «и»;

синтаксис применения:

- ! <value1>;
- <value1> > <value2>;
- <value1> < <value2>;
- <value1> >= <value2>;
- <value1> <= <value2>;
- <value1> | <value2>;
- <value1> & <value2>;

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

value1 и value2 могут быть типам данных - variable и Boolean. При этом Variable преобразуется в Boolean (любое отличное от нуля значение считается истинной). результат всегда будет Boolean.

Математические:

- «+» - сложение;
- «-» - вычитание;
- «\*» - умножение;
- «/» - деление;
- «^» - возведение в степень;

Синтаксис применения

- <value1> + <value2>;
- <value1> - <value2>;
- <value1> \* <value2>;
- <value1> / <value2>;
- <value1> ^ <value2>;

value1 и value2 могут быть типам переменными - variable.

Function <Name>

```
{
<operators>;
return <value>;
```

}; - функция с именем Name. В теле функции содержатся исполняемые операторы operators. Обычно используется для определения состояния узла. Так же может быть вызвана из любого места исполняемого кода.

switch\_Event – событие, в нем могут изменятся состояния узлов. не могут быть вызваны и не возвращают значения.

switch\_Event [<Name>]

(<logic expression>)

{

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

<operators>;

};

- синтаксис объявления события с именем Name (указывать не обязательно); logic expression - логическое выражение, условие выполнения операторов тела события; operators - операторы тела события.

## 2.2. Описание семантики базовых элементов языка

Variable – хранит значение, чтение/запись значения осуществляется по имени. В зависимости от места объявления может быть локальной или глобальной. Глобальное имя доступно в любом месте исполняемого кода. Локальные переменные доступны только внутри объекта-владельца (функции, события) и их значение не сохраняется от вызова к вызову.

Distribution – в явном виде не используется в управляемом коде, необходимо только для описания узлов. Поддерживаются несколько вариантов наборов значений параметров.

Distribution Dis ([exp]<decimal>); - моделирует экспоненциальное распределение с параметром  $\lambda = \text{decimal}$ . decimal должно быть больше нуля. Ключевое слово exp – необязательно.

Distribution Dis (const <decimal>); - моделирует постоянное распределение с параметром decimal (время разыгранное по этому распределению всегда будет равно decimal). decimal должно быть больше нуля.

Distribution Dis (table <decimalT1>,<decimalP1>,...,<decimalTn>,<decimalPn>); - моделирует распределение заданное таблицей из n пар значений (время; вероятность). Значения вероятности должны быть больше 0, меньше равно 1 и не

Инв. N подп.	Подп. и дата	Взам. инв. N	Инв. N дубл.	Подп. и дата

повторятся. График, образуемый последовательность должен строго возрастать. При построении значения времени - округляются до целого в меньшую сторону.

### 2.3. Описание синтаксиса составных элементов языка

knot – модель элемента ЭС, может быть простой или сложной. Простой моделирует конечный элемент, состояние которого описывается законами распределения. Сложный моделирует элемент, включающий в себя другие элементы и состояние которого определяется их состояниями.

Синтаксис определения.

```
[general] knot <Name>
```

```
{
```

```
state: <state list>;
```

```
mode: <mode list>;
```

```
startState: <stState>;
```

```
startMode:<stMode>;
```

```
cntrlMode: <cntrlMode_value>;
```

```
[tableDistribution:
```

```
<state1> |<state2> ... |<sateN>
```

```
<mode1> | <distr11> |<distr12> ...
```

```
<mode2> | <distr21> | ... ;]
```

```
tableStateChange:
```

```
<state1> |<state2> ... |<sateN>
```

```
<mode1> | <change11> | <change21> ...
```

```
<mode2> | <change12> | ... ;
```

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

}

- general - идентификатор главного узла при его отказе эксперимент завершается
- Name - имя узла;
- state list - список возможных состояний узла. Перечисление имен через запятую. Первым указывается названия состояние отказа
- mode list - список возможных состояний узла. Перечисление имен через запятую;
- stState - имя состояния в котором узел находится в момент начала эксперимента
- stMode - имя режима в котором узел находится в момент начала эксперимента
- cntrlMode\_value - тип узла, может принимать значения
- unDistribution - состояние узла определяется распределениями по таблицам tableDistribution и tableStateChange
- unFunction - состояние узла определяется функциями по таблице tableStateChange (tableDistribution не заполняется)
- state1, state2 - перечисление состояний, необходимо перечислить все
- mode1, mode2 - перечисление режимов, необходимо перечислить все
- distr11 - распределение, которому подчиняется узел в состоянии state1 и режиме mode1
- <change11> - имя состояния в котором узел перейдет при выходе из состояния state1 и режима mode1, либо функция, определяющая это состояние

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

**Пример**

```

knot BP_1
{
state: Fail, Work;
mode: Normal,Hard;
startState: Work;
startMode: Normal;

cntrlMode: unDistribution;

tableDistribution:
    | Normal    |    Hard    |
Work | Dis_BP_Normal|    Dis_BP_Hard;

tableStateChange:
    Normal|Hard
Work    |Fail |Fail ;

}

```

, где BP\_1 - имя узла,

state: Fail, Work; - перечисление состояний узла, состояние указанное первым считается необратимым отказом, если оно присвоено узлу, то изменить его уже нельзя.

mode: Normal,Hard;- перечисление режимов работы узла

startState: Work;- состояние при начале эксперимента

startMode: Normal;- режим при начале эксперимента

cntrlMode: unDistribution;- тип контроля, значение unDistribution означает что данный узел изменит состояние по прошествии времени разыгранного генератором случайных чисел

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

tableDistribution:

| Normal | Hard |

Work | Dis\_BP\_Normal| Dis\_BP\_Hard; –Для данного узла это время наработки на отказ, которое имеет разное распределение для двух режимов функционирования, что и отражено в данной таблице.

tableStateChange:

Normal |Hard

Work |Fail |Fail; – таблица, указывающая, в какое состояние перейдет данный узел по истечении разыгранного времени. Узел выбранный в качестве примера отказывает, следовательно, во всех ячейках указаны состояния Fail.

## 2.4. Описание семантики составных элементов языка

<distrib\_dec> – объявление модели распределения случайной величины. Далее <dis\_name> может быть использовано в составе <knot\_dec>, в исполняемом коде в явном виде не используется. Имя распределения <dis\_name> не должно совпадать с любыми именами других элементов модели (<distrib\_name>, <var\_name>, <swe\_name>, <proc\_name>, <fun\_name>). <dis\_param> может принимать один из следующих вариантов значений:

- [exp]<decimal> - моделирует экспоненциальное распределение с параметром лямбда = <decimal>. <decimal> должно быть больше нуля. Ключевое слово exp – необязательно.

- const <integer> - моделирует постоянное распределение с параметром <integer> (время, разыгранное по этому распределению всегда будет равно <integer>). <integer> должно быть больше нуля.

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

- `table {<dis_point>}` `<dis_point>` - моделирует распределение, заданное таблицей из  $n$  пар значений (время; вероятность). Значения вероятности должны быть больше 0, меньше равно 1 и не повторяются. График образовываемый последовательностью должен строго возрастать.

`knot` – модель элемента ЭС, может быть простой или сложной. Простой моделирует конечный элемент, состояние которого описывается законами распределения. Сложный моделирует элемент, включающий в себя другие элементы и состояние которого определяется их состояниями.

- `general` – идентификатор главного узла при его отказе эксперимент завершается

- `<knot_name>` - имя узла;

- `<state_list>` - список возможных состояний узла. Перечисление имен через запятую. Первым указывается названия состояние отказа

- `<state>` - имя состояния, необходимо обеспечивать уникальность имени в списке состояний и режимов.

- `mode list` - список возможных режимов узла. Перечисление имен через запятую;

- `<mode>` - имя режима, необходимо обеспечивать уникальность имени в списке состояний и режимов.

- `<start_st_dec>` – объявление имени состояния, в котором узел находится в момент начала эксперимента.

- `<st_state>` - имя начального состояния, должно совпадать с одним из имен списка `<state_list>`

- `<start_md_dec>` – объявление имени режима, в котором узел находится в момент начала эксперимента.

- `<st_mode>` - имя начального режима, должно совпадать с одним из имен списка `<mode_list>`

- `<cntrlMode_value>` - тип узла, может принимать значения

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

- unDistribution - состояние узла определяется распределениями по таблицам tableDistribution и tableStateChange
- unFunction - состояние узла определяется функциями по таблице tableStateChange (tableDistribution в этом случае исключается)
- <dis\_tb\_dec>- таблица, определяющая законы распределения для каждой пары состояние - режим.
- в таблице первой строкой перечисляются имена всех режимов кроме первого
- каждая следующая строка должна содержать имя режима, к которому относится и далее должны идти имена распределений, соответствующих паре режим – состояние в порядке перечисления состояний в первой строке.
- <change\_st\_tb\_dec>- таблица определяющая состояние, в которое переходит узел из текущего. Может определяться в явном виде путем указания имени состояния, а может определяться через значение, возвращаемое функцией. Задается отдельно для каждой пары состояние - режим. Правила заполнения таблицы аналогичны <dis\_tb\_dec>.

Подробно раскрыть семантику <knot\_dec> и поведение описываемого объекта более удобно на примере:

```
knot BP_1
```

```
{
```

```
state: Fail, Work;
```

```
mode: Normal,Hard;
```

```
startState: Work;
```

```
startMode: Normal;
```

```
cntrlMode: unDistribution;
```

<i>Инва. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инва. N дубл.</i>	<i>Подп. и дата</i>

tableDistribution:

```

    | Normal    |    Hard    |
Work | Dis_BP_Normal|    Dis_BP_Hard;

```

tableStateChange:

```

    Normal |Hard
Work      |Fail |Fail ;};

```

, где

BP\_1 - имя узла,

- state: Fail, Work; – перечисление состояний узла, состояние указанное первым считается необратимым отказом, если оно присвоено узлу, то изменить его уже нельзя.

- mode: Normal,Hard;- перечисление режимов работы узла

- startState: Work; - состояние при начале эксперимента

- startMode: Normal; - режим при начале эксперимента

- cntrlMode: unDistribution; – тип контроля, значение unDistribution означает что данный узел изменит состояние по прошествии времени, разыгранного генератором случайных чисел

- tableDistribution:

```

□ | Normal    |    Hard    |

```

- Work | Dis\_BP\_Normal| Dis\_BP\_Hard; –Для данного узла это время наработки на отказ, которое имеет разное распределение для двух режимов функционирования, что и отражено в данной таблице.

- tableStateChange:

```

□ Normal    |Hard

```

Work |Fail |Fail; – таблица, указывающая, в какое состояние перейдет данный узел по истечении разыгранного времени. Узел выбранный

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

в качестве примера отказывает, следовательно, во всех ячейках указаны состояния Fail.

Таким образом представленный узел в начальный момент времени примет состояние Work и режим Normal. По прошествии времени, разыгранного по закону распределения Dis\_BP\_Normal узел откажет. В случае если до отказа его режим работы будет изменен на Hard действием операторов в составе <switch\_ev\_dec>, тогда время до отказа будет разыграно заново уже по закону распределения Dis\_BP\_Hard.

<switch\_ev\_dec> - является одним из наиболее важных элементов программы. Служит для моделирования реконфигурация и резервирования, схож с условным оператором if.

```
switch_event [<swe_name>] (<arg>) {<operators>}
```

Для switch\_event имя не является обязательным элементом, потому что вызов событий происходит только на системном уровне, и не может быть включен в исполняемый код модели. Аргумент <arg> указанный в круглых скобках определяет момент наступления события, если он не равен нулю, то начинают выполняться операторы <operators>, указанные в фигурных скобках. В общем, оператор switch\_event повторяет оператор if и служит для проверки необходимости провести изменения в модели через выражение <arg> и проведения этих изменений последовательностью операторов. Это механизм необходим для моделирования реконфигураций и резервирования.

### 3. СПОСОБЫ СТРУКТУРИРОВАНИЯ ПРОГРАММЫ

Изначально модель имеет структурированный вид. В общем виде она делится на четыре части:

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

1. Модели элементов ЭС (knot)
2. Модели действий реконфигурации (switch\_ewent)
3. Критерии отказов (function)
4. Вспомогательные элементы (distribution, variable)

В тексте программы рекомендуется размещать элементы в таком порядке:

1. Объявления распределений (distribution)
2. Объявления глобальных переменных (variable)
3. Объявление узлов (knot)
4. Функции (function)
5. События (switch\_event)

Такой порядок обеспечивает правильную последовательность объявлений имен до их использования. Такой порядок является не является обязательным и может быть изменен с сохранение правила предшествования объявления имени его использованию.

Для правильного структурирования операторов в программе необходимо учитывать порядок исполнения имитационного эксперимента.

Каждый эксперимент начинается в момент системного времени, установленного в 0. Всем узлам присваивается начальные состояния и режимы. Далее происходит розыгрыш времен по соответствующим распределениям и системное время устанавливается в момент ближайшего изменения состояния узла. После изменения его состояния начинается процесс определения состояния всех составных узлов путем вызова указанных в теле функций. После этого проверяются условия событий switch\_event и выполняются их операторы. В случае изменения состояния еще какого-либо узла вызовы функций и проверка событий повторяются. Этот процесс повторяется до тех пор, пока не будет достигнуто устойчивое

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

состояние системы или не привесится счетчик зацикливания (по умолчанию 1000 повторов).

#### 4. СРЕДСТВА ОБМЕНА ДАННЫМИ

Стандартный отчет о ходе эксперимента – при проведении серии экспериментов в файл выводятся результаты эксперимента в виде реализации наработки на отказ. Эти данные могут в дальнейшем быть обработаны в сторонних программах.

Расширенный отчет – при установке соответствующего флага в окне компилятора в файл logEx.txt выводится подробный отчет о ходе эксперимента. В отчете содержится детализация каждого шага выполнения программы для всех экспериментов в серии. В детализацию входит следующая информация:

- Имена исполненных событий
- Состояния назначенные узлам выполнившимися событиями
- Имя и результат выполнения функции определяющей состояние узла
- Таблица режимов и состояний узлов

Данный отчет рекомендуется использовать для поиска критических ошибок, не обнаруженных в ходе отладки. Его формирование замедляет моделирование и отчет занимает значительный объем.

#### 5. ВСТРОЕННЫЕ ЭЛЕМЕНТЫ

Встроенные функции предназначены для работы с объектами модели. Они предоставляют программисту инструменты для задания условий событий и внесения изменений в состояние модели.

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

Knot1:State1 – возвращает значение типа boolean: 1 если узел Knot1 находится в состоянии State1, иначе 0. Аналогично для режима.

Knot1:State - возвращает индекс текущего состояния. Аналогично для режима.

Knot1:State1->Knot1:State2 - возвращает значение типа boolean: 1, если в данный момент времени узел Knot1 переходит в состояние State2 из State1. Аналогично для режима.

SetState (Knot1:State1) - изменяет состояние для узла Knot1 на State1.

SetMode(Knot1:Mode1) - изменяет режим для узла Knot1 на Mode1.

## 6. СРЕДСТВА ОТЛАДКИ ПРОГРАММЫ

Управляемый эксперимент – средство контроля за изменениями в модели ЭС. Позволяет провести имитационный эксперимент определяя отказывающийся элемент ЭС каждом шаге моделирования и контролируя действия и реакцию модели используя подробный отчет о ходе экспериментов.

Применение данного способа дает возможность удостовериться в правильности модели. Для использования управляемого эксперимента рекомендуется составить список возможных последовательностей событий и правильных реакций на них.

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

## СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 27.002-2015. Надёжность в технике. Термины и определения.
2. ГОСТ РВ 20.39.303-98. Комплексная система общих технических требований. Аппаратура, приборы, устройства и оборудование военного назначения. Требования к надёжности. Состав и порядок задания.
3. ГОСТ 27.301-95. Расчёт надёжности. Основные положения.
4. ГОСТ РВ 20.39.302-98. Комплексная система общих технических требований. Аппаратура, приборы, устройства и оборудование военного назначения. Требования к программам обеспечения надёжности и стойкости к воздействию ионизирующих и электромагнитных излучений.
5. РДВ 319.01.05-94, ред. 2-2000 «Комплексная система контроля качества. Аппаратура, приборы, устройства и оборудование военного назначения. Принципы применения математического моделирования при проектировании».
6. Жаднов, В. В. Автоматизация проектных исследований надёжности радиоэлектронной аппаратуры. / В. В. Жаднов, Ю. Н. Кофанов, Н. В. Малютин и др. - М.: Изд-во «Радио и связь», 2003. - 156 с.
7. Жаднов, В. В. Управление качеством при проектировании теплонагруженных радиоэлектронных средств. / В. В. Жаднов, А. В. Сарафанов. М.: Изд-во «Солон-Пресс», 2004. - 464 с.
8. Шалумов, А. С. Автоматизированная система АСОНИКА для проектирования высоконадежных радиоэлектронных средств на принципах CALS-технологий: Том 1. / А. С. Шалумов, Ю. Н. Кофанов, Н. В. Малютин, Д. А. Способ, В. В. Жаднов и др. // Под ред. Ю. Н. Кофанова, Н. В. Малютина, А. С. Шалумова. - М.: Изд-во «Энергоатомиздат», 2007. - 538 с.

<i>Инв. N подл.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

RU.17701729.22005-01 35

9. РДВ 319.01.16-98. Радиоэлектронные системы военного назначения. Типовые методики оценки показателей безотказности и ремонтпригодности расчетно-экспериментальными методами.
10. RU.17701729.22002-01 31 01. Программный комплекс АСОНИКА-К. Система АСОНИКА-К-РЭС. Описание применения.
11. RU.17701729.22002-01 33 01. Программный комплекс АСОНИКА-К. Система АСОНИКА-К-РЭС. Руководство программиста.

<i>Инв. N подп.</i>	<i>Подп. и дата</i>	<i>Взам. инв. N</i>	<i>Инв. N дубл.</i>	<i>Подп. и дата</i>

