ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 55692— 2013

МОДУЛИ ЭЛЕКТРОННЫЕ

Методы составления и отладки тест-программ для автоматизированного контроля

Издание официальное

Москва
Стандартинформ
2014

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Закрытым акционерным обществом «Авангард-ТехСт» (ЗАО «Авангард-ТехСт») и ОАО «Авангард»

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 420 «Базовые несущие конструкции, печатные платы, сборка и монтаж электронных модулей»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 22 ноября 2013 г. № 2087-ст

4 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (gost.ru)

Стандартинформ, 201

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОДУЛИ ЭЛЕКТРОННЫЕ

Методы составления и отладки тест-программ для автоматизированного контроля

Electronic modules.
Methods for test-programs compiling and debugging for automatized control

Дата введения — 2014—06—01

1 Область применения

Настоящий стандарт распространяется на электронные модули (ЭМ), построенные на цифровых интегральных микросхемах.

Настоящий стандарт устанавливает методы составления и отладки тестовых программ для автоматизированного контроля цифровых ЭМ на установках тестового контроля и диагностики (УТК), разрабатываемых и изготовляемых по ГОСТ Р 52154.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты

ГОСТ Р 52154–2003 Аппаратура радиоэлектронная контрольно-измерительная технологическая. Общие технические условия

ГОСТ 19919–74 Контроль автоматизированный технического состояния изделий авиационной техники. Термины и определения

ГОСТ 20911–89 Техническая диагностика. Термины и определения

Примечание – При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины, определения, обозначения и сокращения

3.1 Термины и определения

В настоящем стандарте применены термины по ГОСТ 19919, а также следующие термины с соответствующими определениями:

3.1.1 **тест-программа:** Программа, по которой выполняется автоматизированный контроль ЭМ.

3.1.2 **логический контроль электронного модуля:** Диагностирование неисправностей логического типа (фиксированный «0» или «1») на выходах логических элементов ЭМ.

Издание официальное

3.1.3 **параметрический контроль электронного модуля:** Проверка уровней напряжений входных и выходных сигналов; проверка задержки прохождения сигналов от входа контролируемого ЭМ до его выхода.

3.1.4 **Тестовое техническое диагностирование:** По ГОСТ 20911.

3.1.5 **канал установки УТК:** Часть оборудования установки УТК, служащая как для формирования и передачи логических сигналов на вход контролируемого ЭМ, так и для приема и анализа сигналов, снимаемых с его выхода.

3.2 Обозначения и сокращения

В настоящем стандарте приняты следующие обозначения и сокращения:

ВК – возврат каретки;

Лог. «1» («0») – логическая единица (нуль);

ОК – объект контроля

ПС – перевод строки;

ПЭВМ – персональная вычислительная машина;

ТУ – технические условия;

УТК – установка тестового контроля;

ФЗМ – функционально законченный модуль;

ЭМ – электронный модуль;

ЯСТЕК – язык системы тестового контроля.

4 Основные нормативные положения

4.1 Установки тестового контроля должны обеспечивать:

- программную коммутацию каналов установки в соответствии с расположением входов-выходов на электрических соединителях контролируемого цифрового ЭМ;
- автоматическую подачу на входы ЭМ наборов входных воздействий тестовой программы;
- автоматический анализ логических сигналов, снимаемых с выходов контролируемого ЭМ, в каждом наборе тестовой программы;
- автоматическую блокировку каналов, не подлежащих анализу на данном наборе;
- автоматическую выдачу информации для оценки результатов контроля.

4.2 Тестовой программой называется последовательность входных сигналов и соответствующих им выходных сигналов, обеспечивающих контроль исправности цифрового модуля, узла или устройства.

Тестовая программа составляется для цифрового модуля конкретного типа.

4.3 Тестовая программа составляется на языке высокого уровня ЯСТЕК (язык системы тестового контроля).

4.4 Тестовую программу, написанную на языке ЯСТЕК, можно использовать для любого типа УТК.

4.6 Ввод тестовой программы должен осуществляться с ПЭВМ, подключенной к установке тестового контроля. Отладочная информация должна выводиться на монитор ПЭВМ.

4.7 Тестовая программа должна обеспечивать следующие виды контроля модуля:

- логический контроль;
- параметрический контроль.

4.8 Тестовая программа должна составляться на основании технических условий на ЭМ. Номенклатуру проверяемых параметров определяет разработчик ТУ.

4.9 Проверки с помощью тестовой программы следует включать в методики проверки ТУ на соответствующий ЭМ.

5 Основные характеристики языка системы тестового контроля

5.1 Описание языка тестового контроля

5.1.1 Язык ЯСТЕК характеризуется следующими данными:

– единый формат представления операторов, что позволяет их использовать, читать и понимать;

– символьное представление переменных величин и констант, наиболее приемлемое для пользователей;

– в процессе трансляции вычисляются значения выражений, используемых в качестве операндов, а результат вычислений воспринимается как значение операнда;

– большой набор условных операторов управляющих ходом тестовой программы;

– библиотечные функции типа «бегущий 0», «бегущая 1», случайный код и т. п., что значительно облегчает составление тестовых программ;

– управление модулями ведется в соответствии с коммутацией, объявленной для переменных.

5.1.2 При разработке алгоритмов контроля функционально законченного модуля (ФЗМ) следует объединять в группы контакты контролируемого модуля, по которым возможно алгоритмическое построение входных воздействий и выходных реакций. У ФЗМ следует выделять операционную, адресную и управляющую части. При разработке той части алгоритма, которая осуществляет генерацию информации для контроля операционной и адресной частей модуля, разработчик должен задавать закон изменения входной информации по наборам и описать, исходя из знаний функций, реализуемых ФЗМ, закон изменений выходной информации как эталонной реакции.

Пример – При контроле операционных запоминающих устройств адресные входы объединяются в одну группу, и закон изменения кода на ней задают, начиная с нулевого значения до максимального в виде прибавления лог. «1».

Генерацию информации для управляющей части следует осуществлять путем непосредственного задания управляющих кодов или временны́х диаграмм.

Пример – Каждое поле микрокоманд задается определенным кодом.

5.1.3 При использовании языка ЯСТЕК значительно сжимается описание тестовой последовательности. Сжатие происходит за счет следующих возможностей:

1) тестовый набор строится только по тем контактам, которые интересуют пользователя на этом наборе;

2) при формировании тестового набора вводятся только изменения относительно предыдущих наборов в прямом коде или в коде задания конкретного закона изменения состояния на контактах контролируемого модуля;

3) любым контактам электрического соединителя могут задаваться нормированные импульсные последовательности;

4) изобразительные средства языка позволяют описывать изменения состояний на контактах электрического соединителя контролируемого модуля в виде временны́х диаграмм;

5) возможна организация условных переходов по результатам сравнения.

2.2 Символы языка тестового контроля (алфавит)

В языке тестового контроля используются следующие символы:

<символ> ::= <буква> | <цифра> | <ограничитель>
<буква> ::= <любая прописная буква латинского алфавита>
*<цифра> ::= < **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**>*
<ограничитель> ::= <знак> | <разделитель> | <скобка>
*<знак> ::= <**+**> | <**-**> | <*****> | <**/**> | <**!**> | <**&**> | <**_**> | <**^**>*
*<разделитель> ::= <**:**> | < > | <**#**> | <ПС> | <ВК> | <**;**> | <**,**> | <**@**>*
*<скобка> ::= <**<>** | <**>>***

Примечания

1 Разделитель < > означает пробел.

2 Использование букв русского алфавита разрешается только в комментариях и текстовых переменных.

2.3 Идентификаторы языка ЯСТЕК

Идентификаторы служат для обозначения меток, переменных, констант.

Структура идентификатора:

<идентификатор> ::= <буква> { <буква> | <цифра>}

Примечания

1 Каждый идентификатор (кроме меток) должен быть предварительно объявлен.

2 Один и тот же идентификатор нельзя использовать для обозначения двух различных переменных, меток, констант.

3 Первые шесть знаков идентификатора являются значащими, каждый идентификатор должен быть единственным в пределах первых шести знаков, первый знак не должен быть цифрой, специальные знаки включать в идентификатор не допускается.

4 В качестве идентификаторов запрещается использовать сочетания знаков:

R0, R1 – R7, SP, PC.

2.4 Операторы языка ЯСТЕК

Основной функциональной единицей языка ЯСТЕК является оператор.

<оператор> ::=

<невыполняемые операторы> |

<выполняемые операторы>

<невыполняемые операторы> ::=

<операторы объявления констант> |

<операторы объявления переменных> |

<операторы объявления имени> |

<операторы настройки транслятора>

<выполняемые операторы> ::=

<арифметические операторы> |

<операторы передачи данных> |

<логические операторы> |

<операторы управления ходом тест-программы> |

<операторы ввода-вывода> |

<операторы вызова библиотечных функций>

Примечание – Каждый оператор занимает одну строку текста тест-программы. Каждая строка может содержать до 128 знаков и заканчиваться знаком «перевод строки» (ПС) или «возврат каретки» (ВК). Допускается использование последовательности ВК ПС.

Пример – Операторы ЯСТЕК:

AD OPER1, 1, OPER2; Сложение OPER1 с единицей, результат в OPER2.

2.5 Общий формат оператора в языке ЯСТЕК

Общий формат оператора в ЯСТЕК имеет следующий вид:

<формат оператора ЯСТЕК> ::=

[<метка> :] [<операция> <операнды>] [;<комментарий>] ВК

<метка> ::= <идентификатор>

<операция> ::= <имя оператора>

<операнды> ::=

<операнд> |

<операнды> <разделительный знак> <операнд>

<разделительный знак> ::= < > | < >

<операнд> ::=

<число> |

<константа> |

<выражение> |

<переменная> |

<аргумент в пределах контекста операции>

<комментарий> ::=

<последовательность основных символов, кроме знаков ПС и ВК>

Примечания

1 Число операндов зависит от типа оператора.

2 Метки и комментарии необязательны.

3 Комментарий может содержать символы русского алфавита.

Примеры
M1: SU OPER1, <2+2>*3, OPER3 ; Вычитание 12 из OPER1, результат в OPER3
VX ADRES ; Перевести контакты, соответствующие переменной ADRES, режим Вход
MV 1023, ADRES1, ADRES2 ; Занести 1023 в переменные ADRES1, ADRES2
KN ; Конец программы

2.6 Константы в языке ЯСТЕК

Константы в языке ЯСТЕК бывают следующих типов:
1) числовые;
2) строковые;
3) форматные;
4) временные диаграммы

<операторы объявления констант> ::=
<операторы объявления числовых констант> |
<операторы объявления строковых констант> |
<операторы объявления форматных констант> |
<операторы объявления временных диаграмм>

Константы, используемые при описании алгоритма контроля, – символьные имена (идентификаторы), значения которых не могут изменяться в процессе выполнения тест-программы.

Пример
TP <Пример> <Устройство>
C10 Constant 1997 ; Объявляют константу, равную 1997
NACH ; Начало выполняемой части программы
PRD Constant ; Выводят константу в десятичном виде
KN ; Завершают программу
KTP

Такая программа выведет на консоль число 1997 и завершит свою работу.

2.7 Числовые константы

Числовые константы используют для придания осмысленных символьных имен (идентификаторов) часто используемым или важным числовым параметрам, которые не изменяются в ходе выполнения программы.

Допускается использование следующих систем счисления:
двоичная;
четверичная;
восьмеричная;
десятичная;
шестнадцатеричная.

Пример – Числовые константы можно использовать для задания верхней границы счетчика цикла или же для задания неких управляющих слов, подаваемых на входы тестируемого устройства.

<оператор объявления числовых констант> ::=
C2 <имя> <двоичное число> |
C4 <имя> <четверичное число> |
C8 <имя> <восьмеричное число> |
C10 <имя> <десятичное число> |
C16 <имя> <шестнадцатеричное число>

Пример – Использование числовой константы:
TP <Пример> <Устройство>
C4 FIRST 23 ; Начальное значение выводимых чисел
C10 MAX_COUNT 5 ; Счетчик общего количества чисел
NUMBER:WD 0 ; Переменная – выводимое число
NACH ; Начало выполняемой части программы
MV FIRST, NUMBER ; Инициализация переменных
LABEL: ; Начало цикла вывода чисел
PRD NUMBER ; Вывод очередного числа
PRVK

IC NUMBER ; Увеличение выводимого числа на 1
DO LABEL, MAX_COUNT ; Конец цикла вывода чисел
KN ; Завершение работы программы
KTP
Эта программа выведет ряд чисел: 11, 12, 13, 14, 15.

2.8 Строковые константы

Строковые константы используются обычно для задания выводимых сообщений. Сами же строки символов задействованы в операторах начала программы, вывода текста на консоль и в объявлении временных диаграмм.

<оператор объявления строковой константы> ::=
<TEXT> *<имя> <строка символов>*
<строка символов> ::= ***<<***символы***>>***

Пример –
TP <Пример> <Устройство> ; Оператор начала программы
TEXT Str <Вот вам и сообщение> ; Описание строковой константы
NACH ; Начало исполняемой части
PRT Str ; Вывод сообщения на консоль
KN ; Завершение работы программы
KTP
Эта программа выведет строку «Вот вам и сообщение».

2.9 Временны́е диаграммы

Константы типа временны́х диаграмм используют для задания стандартных последовательностей обмена с тестируемым устройством.

Пример – Для задания состояний управляющих и контрольных контактов микросхемы памяти при записи и чтении.

<операторы временно́й диаграммы> :: =
<оператор временно́й диаграммы> ВК <операторы строк>
<оператор временно́й диаграммы> ::=
VD <идентификатор> <число тактов > <число строк >
<операторы строк> ::=
<оператор строки> |
<операторы строк> ВК <оператор стоки>
<оператор строки> ::=
S <строка символов>

Примечание – Число тактов и число строк задаются цифрами или простым выражением без использования идентификаторов, например 2+<3-4>.

В строке разрешены только знаки подчеркивания и стрелка, указывающая вверх, с помощью которых стилизовано описание изменения логических состояний. Знак подчеркивания соответствует лог. «0», знак «^» соответствует лог. «1».

Пример
S <_^^__^^> ; 0110011
S <^__^_^_> ; 1001010

2.10 Форматные константы

Форматные константы используют в операторах упаковки и распаковки.

<оператор форматная константа> ::=

F *<имя> <младший разряд> <старший разряд>*

Пример – F FrmConst 5, 10.

2.11 Переменные

Переменные – это имена (идентификаторы), значение которых может изменяться в процессе выполнения программы. В данной версии языка ЯСТЕК существуют переменные следующих типов:

1) числовые переменные;
2) переменные описания группы контактов.

2.12 Числовые переменные

Числовые переменные языка ЯСТЕК – 16-битные целые со знаком

<оператор объявления переменной слово> ::=
<идентификатор> : <имя операции> <операнды>
<имя операции> ::=
<WD> | ***<WD2>*** | ***<WD4>*** | ***<WD8>*** | ***<WD10>*** | ***<WD16>***
<WD>, ***<WD10>*** *::=*
<для введения начального значения переменных по основанию 10>
<WD2> *::= <по основанию 2>*
<WD4> *::= <по основанию 4>*
<WD8> *::= <по основанию 8>*
<WD16> *::= <по основанию 16>*
<операнды> ::= <операнд>
<операнд> ::= <число> | ***<выражение>***
<число> ::= < ***D****xxx>* | ***<^B****xxx>* | ***<^O****xxx>* | ***<^T****xxx>* | ***<^H****xxx>* | *<xxx>*
<^D*xxx> ::= <число задается по основанию 10>*
<^B*xxx> ::= <число задается по основанию 2>*
<^T*xxx> ::= <число задается по основанию 4>*
<^O*xxx> ::= <число задается по основанию 8>*
<^H*xxx> ::= <число задается по основанию 16>*
<xxx> ::=
<число xxx по умолчанию интерпретируется по основанию 10 либо по тому, которое указано в имени операции (WD8 и т. п.)>
<выражение> ::=
<операнд> <знак> <выражение> |
<операнд> |
<знак операции с одним операндом>
<знак операции с одним операндом> ::= **+** | **-**
<операнд> ::=
< число> |
<константа> |
<переменная> |
< *<выражение>* **>**
<знак> ::= <+> | <-> | </> | <*> | <!> | <&>
<+> *::= <сложение>*
<-> *::= <вычитание>*
</> *::= <деление>*
<*> *::= <умножение>*
<!> *::= <логическое или>*
<&> *::= <логическое и>*

Примечание – Выражения обрабатываются слева направо без правил приоритетов, например, 1+<2*3> будет равно 7, а 1+2*3 будет равно 9.

Пример – Объявление переменных:

VAR1: WD161 0FF ; Начальное значение VAR1 = 255_{10}

VAR2: WD 35+<2*4>-3*2+^B10 ; Начальное значение VAR2 = 82_{10}

VAR3: WD8 11+^D11 ; Начальное значение VAR3 = 20_{10}

VAR4: WD 5+<2+<3-4>> ; Начальное значение VAR4 = 6_{10}

2.13 Группы контактов

Переменная описания группы контактов предназначена для формирования тестового набора и обозначает соответствие между разрядами машинного слова, в котором формируется значение переменной, и контактами, к которым подключается модуль.

Язык ЯСТЕК позволяет описывать тестовые наборы длиной до 256 бит. Таким образом, имеется возможность проверять модуль, имеющий до 256 контактов, на которых формируются состояния, соответствующие тестовому набору. Двоичные значения переменных этого типа определяют состояния сигналов, подаваемых на входы контролируемого модуля и ожидаемые состояния на выходах. В группу контактов, описываемых переменной, включаются контакты, объединение которых имеет некоторый смысл.

Пример – Контакты, по которым подаются состояния, соответствующие адресу оперативной памяти.

Все контакты, определяемые переменной описания группы контактов в каждом тестовом наборе, должны быть определены только как входы или выходы контролируемого модуля.

<операторы задания переменных описания группы контактов> ::=
<начало списка> ВК <список контактов> ВК <конец списка>
<начало списка> ::= ***GR***
<конец списка> ::= ***KGR***
<список контактов> ::=
<одна группа> <список контактов> |
<одна группа>
<одна группа> ::= ***GK*** *<идентификатор> <номера контактов>*
<номера контактов> ::= <номер> | *<номер> <номера контактов>*

Пример

GR ; Начало объявления групп контактов

GK DAN 5, 28, 13 ; Объявляется группа DAN из трех контактов

GK ADR 17, 41, 4, 32 ; Объявляется группа ADR из четырех контактов

GK YPR 7 ; Объявляется группа YPR из одного контакта

KGR ; Конец объявления групп контактов

Примечания

1 Использовавшийся ранее параметр оператора GR (число групп контактов) теперь игнорируется, поскольку эти объявления все равно завершаются оператором KGR.

2 Число переменных типа группа контактов не ограничено.

3 Переменная описания группы контактов может описывать от 1 до 32 контактов.

4 Если не будет указана другая система счисления, то номера контактов модуля задаются по основанию 10. Нумерация контактов начинается с 0.

2.14 Операторы структуры тест-программы

Операторы структуры задают название теста, название тестируемого устройства, а также делят программу на область деклараций (объявлений констант и переменных) и область исполняемых операторов. Также существует оператор, завершающий тест-программу.

<оператор структуры> ::=
<оператор начала программы> |
<оператор начала исполняемой части> |
<оператор конца программы>
<оператор начала программы> ::=
TP *<название теста> <название устройства>*
<название теста> ::= <строка>
<название устройства> ::= <строка>
<оператор начала исполняемой части> ::= ***NACH***
<оператор конца программы> ::= ***KTP***

Пример

TP <Ячейка ОЗУ> <XXXX3.428.357.931.1> ; Описывают тест и устройство
C2 Const 1001010 ; Далее идут объявления

NACH ; Начало исполняемой части
PRB Const ; Здесь идут исполняемые операторы
KN
KTP ; Конец программы

Примечания

1 Использовавшиеся ранее операторы .MCALL и VS теперь игнорируются и могут быть опущены.
2 Оператор NACH следует после объявления констант и переменных, используемых в данной тест-программе.

2.15 Операторы передачи данных

Операторы передачи данных служат для копирования данных программы (чисел) из одних областей памяти в другие.

В ЯСТЕК реализованы следующие операторы передачи данных:

1) оператор присваивания – передает содержимое одной переменной или константы в другую переменную;

2) оператор разборки – служит для занесения разрядов источника в младшие разряды получателей;

3) оператор сборки – для сборки младших разрядов источников в одно единое слово;

4) оператор распаковки – выделенная форматной константой часть источника передается получателю;

5) оператор упаковки – младшая часть источника передается в выделенную форматной константой часть получателя.

2.16 Оператор присваивания

Оператор присваивания передает содержимое одной переменной или константы в другую переменную.

<оператор присваивания> ::= ***MV*** *<источник> <список получателей>*
<источник> ::=
<числовое выражение или константа> |
<числовая переменная> |
<переменная группы контактов>
<список получателей> ::=
<получатель> |
<получатель> <список получателей>
<получатель> ::=
<числовая переменная> |
<переменная группы контактов>

Пример
C8 Const 11
NACH
MV 123, Variable ; В переменную 'Variable' заносится 123_{10}
MV 3*<2+2>, Summ ; В переменную 'Summ' заносится 12_{10}
MV Const, Variable ; В переменную 'Variable' заносится 9_{10}
MV Var, Sum, Sum2 ; Пересылка содержимого Var в Sum и Sum2

Примечания

1 Теперь все числа по умолчанию задаются по основанию 10.
2 Требовавшийся ранее знак # при описании чисел теперь не требуется, если он указывается, то число считается по основанию 8.

Применение десятичной точки в конце числа как признака основания 10 запрещено.

2.17 Операторы упаковки и распаковки

Оператор упаковки передает младшую часть операнда-источника шириной в длину формата в соответствующую указанной в форматной константе часть операнда-получателя. Оператор распаковки производит обратную операцию – передает соответствующую часть операнда-источника в младшие разряды операнда-получателя.

<операторы упаковки и распаковки> ::=
<оператор упаковки> |
<оператор распаковки>
<оператор упаковки> ::=
MVF *<источник> <получатель> <форматная константа>*
<оператор распаковки> ::=
FMV *<источник> <получатель> <форматная константа>*
<источник> ::=
<числовое выражение или константа> |
<числовая переменная> |
<переменная группы контактов>
<получатель> ::=
<числовая переменная> |
<переменная группы контактов>

Пример

```
C2 BinConst 01010
F Format 10, 14
F Back 11, 13
Var: WD 0
Dst: WD 0
NACH
MVF BinConst, Dst, Format ; Dst = 10'1000'0000'0000
FMV Dst, Var, Back ; Var = 101
```

2.18 Операторы сборки и разборки

Оператор сборки собирает младшие разряды операндов-источников в единое слово, которое заносится в операнд-получатель. При этом младший бит первого операнда-источника заносится в младший бит операнда-получателя. Оператор разборки проводит обратную операцию – разносит младшую часть операнда-источника по младшим разрядам операндов-получателей. При этом младший бит операнда-источника заносится в младший бит первого операнда-получателя.

<операторы сборки и разборки> ::=
<оператор сборки> |
<оператор разборки>
<оператор сборки> ::=
SBR *<получатель> <список источников>*
<оператор разборки> ::=
RBR *<источник> <список получателей>*
<список источников> ::=
<источник> |
<источник> <список источников>
<список получателей> ::=
<получатель> |
<получатель> <список получателей>
<источник> ::=
<числовое выражение или константа> |
<числовая переменная> |
<переменная группы контактов>
<получатель> ::=
<числовая переменная> |
<переменная группы контактов>

Пример

```
SBR Result, 1, 0, 1, 1, 0, 0, 1 ; Result = ^B1001101
SBR Result, ^B10, ^B11, 1, ^B1011 ; Result = ^B1110
RBR ^B110101100, RD0, RD1, RD2, RD3 ; RD0 = 0
 ; RD1 = 0
```

; RD2 = 1
; RD3 = 1

Примечание – Число операндов не может превышать 25.

2.19 Арифметические операторы

Арифметические операторы выполняют арифметические функции над операндами-источниками и помещают результат в операнд-получатель.

В ЯСТЕК реализованы следующие операторы:

– оператор арифметического сложения – суммирует два источника и помещает результат в получатель;

– оператор арифметического вычитания – из первого источника вычитается второй, разность заносится в получатель;

– операторы инкремента и декремента – изменяют операнд на единицу (увеличивают или уменьшают соответственно).

<арифметические операторы> ::=
<оператор арифметического сложения> |
<оператор арифметического вычитания> |
<оператор инкремента> |
<оператор декремента>
<оператор арифметического сложения> ::=
AD *<источник>, <источник>, <получатель>*
<оператор арифметического вычитания> ::=
SU *<источник>, <источник>, <получатель>*
<оператор инкремента> ::=
IC *<операнд>*
<оператор декремента> ::=
DC *<операнд>*
<операнд> ::= <получатель>
<источник> ::=
<числовое выражение или константа> |
<числовая переменная> |
<переменная группы контактов>
<получатель> ::=
<числовая переменная> |
<переменная группы контактов>

Пример
AD 2, 2, Result ; Result = 4
SU 5, 2, Result ; Result = 3
SU 5, 6, Result ; Result = -1
MV 386, Result
IC Result ; Result = 387
MV 586, Result
DC Result ; Result = 585/

2.20 Логические операторы

Логические операторы выполняют операции над каждым разрядом операндов отдельно. Например, они позволяют провести поразрядное логическое «и»

– двух переменных и занести результат в третью.

– В ЯСТЕК существуют следующие логические операторы:

– оператор логического сложения – проводит поразрядное логическое «или» нескольких операндов-источников и заносит результат в операнд-получатель.

Таблица значений:

a	b	a|b
0	0	0

0 1 1
1 0 1
1 1 1;

– оператор логического умножения – проводит поразрядное логическое «и» нескольких операндов-источников и заносит результат в операнд-получатель.

Таблица значений:

a b a&b
0 0 0
0 1 0
1 0 0
1 1 1;

– оператор сложения по модулю 2 – проводит поразрядное логическое исключающее «или» двух операндов-источников и заносит результат в операнд-получатель.

Таблица значений:

a b a^b
0 0 0
0 1 1
1 0 1
1 1 0;

– оператор инверсии – инвертирует все биты операнда;

– оператор очистки разрядов – биты, которым в операнде-маске присвоены значения лог. «1», в операнде-получателе сбрасываются в лог. «0»;

– оператор сдвига вправо – осуществляет логический сдвиг операнда-источника вправо на указанное во втором источнике число разрядов и заносит результат в получатель. Сдвиг проводится без размножения знака, старшие биты получателя заполняются нулями;

– оператор сдвига влево – осуществляет логический сдвиг операнда-источника влево на указанное во втором источнике число разрядов и заносит результат в получатель. Младшие биты получателя заполняются нулями;

– оператор очистки – все разряды операнда сбрасываются в лог. «0».

<логические операторы> ::=
<оператор логического сложения> |
<оператор логического умножения> |
<оператор сложения по модулю 2> |
<оператор инверсии> |
<оператор очистки разрядов> |
<оператор сдвига вправо> |
<оператор сдвига влево> |
<оператор очистки>
<оператор логического сложения> ::=
BS *<получатель> <список источников>*
<оператор логического умножения> ::=
AN *<получатель> <список источников>*
<список источников> ::=
<источник> <список источников>
<источник>
<оператор сложения по модулю 2> ::=
XR *<источник> <источник> <получатель>*
<оператор инверсии> ::=
CM *<получатель>*
<оператор очистки разрядов> ::=
BC *<маска>, <получатель>*
<маска> ::= <источник>
<оператор сдвига вправо> ::=
SR *<источник>, <число разрядов>, <получатель>*
<оператор сдвига влево> ::=
SL *<источник>, <число разрядов>, <получатель>*

<число разрядов> ::= <источник>
<оператор очистки> ::=
***CL** <получатель>*

Пример
BS Result, ^B1100, ^B1010 ; Result = 1110
AN Result, ^B1100, ^B1010 ; Result = 1000
XR ^B1100, ^B1010, Result ; Result = 0110
MV ^B1011010, Result
CM Result ; Result = 11111111 11111111 11111111 10100101
MV ^B111100, Result
BC ^B010110, Result ; Result = 101000
SR ^B101101, 2, Result ; Result = 001011
SL ^B101101, 2, Result ; Result = 10110100
MV ^B101101, Result
CL Result ; Result = 0/

2.21 Операторы работы с библиотечными функциями

5.21.1 Библиотечными функциями называют арифметические операторы с одним операндом. Они предназначены для корректной работы с переменными типа группа контактов, разрядность которых может быть произвольной (1–32).

В данной версии языка есть следующие библиотечные функции:

– формирование кода «бегущая 1»;
– формирование кода «бегущий 0»;
– формирование кодов прямого счета;
– формирование кодов обратного счета;
– формирование случайных кодов.

1.1.2 Формирование кода «бегущая 1»

Все биты числа, кроме одного, задаются нулями. Бит, значение которого равно единице, перемещается от младшего разряда к старшему. По достижении старшего разряда переменная сбрасывается в начальное состояние. Начальным значением функции считается лог. «0».

Пример – Для 3-разрядной переменной:
000
001
010
100
000
001

5.21.3 Формирование кода «бегущий 0»

Все биты числа кроме одного задаются единицами. Бит, значение которого равно нулю, перемещается от младшего разряда к старшему. По достижении старшего разряда переменная сбрасывается в начальное состояние. Начальным значением функции считается состояние со всеми единицами (конкретное число зависит от разрядности переменной).

Пример – Для 3-разрядной переменной:
111
110
101
011
111
110

1.1.4 Формирование кодов прямого счета

Прибавляет к текущему значению операнда единицу. Отличается от оператора инкремента тем, что числовое кольцо значений заворачивается не при превышении максимального значения целого числа (32-разрядного), а при превышении максимального значения, допустимого для разрядности данной переменной типа группа контактов. Начальное значение функции – 0.

5.21.5 Формирование кодов обратного счета

Вычитает из текущего значения операнда единицу. Действует аналогично формированию кодов прямого счета. Начальное значение функции – 0.

1.1.6 Формирование случайных кодов

Заносит в операнд случайное число. Начальное значение смысла не имеет.

<операторы библиотечных функций> ::=
<оператор вызова функции> |
<оператор установки начального значения>
<оператор вызова функции> ::=
FN *<имя функции> <получатель>*
<оператор установки начального значения> ::=
SFN *<имя функции> <получатель>*
<имя функции> ::=
<формирование бегущей 1> |
<формирование бегущего 0> |
<формирование кодов прямого счета> |
<формирование кодов обратного счета> |
<формирование случайных кодов>
<формирование бегущей 1> ::= ***FB1***
<формирование бегущего 0> ::= ***FB0***
<формирование кодов прямого счета> ::= ***SHP***
<формирование кодов обратного счета> ::= ***SHO***
<формирование случайных кодов> ::= ***SLK***

Пример

```
TP <Библиотечные функции> <UTK-128> ; Описывают тест и устройство
Var: WD 0
NACH
Label:
FN FB1 Var ; формируют бегущую 1 в переменной Var
PRB Var ; Выводят содержимое переменной на консоль
PRVK
DO Label,10 ; функцию FB1 вызывают 10 раз
KN
KTP
```

2.22 Операторы управления ходом тест-программы

5.22.1 Операторы управления ходом тест-программы делятся на четыре группы:

1) операторы передачи управления;
2) операторы условного перехода;
3) оператор цикла;
4) оператор паузы.

1.1.2 Операторы передачи управления

Операторы передачи управления проводят безусловный переход на другую часть программы. Существует три оператора передачи управления:

– оператор безусловного перехода – просто перемещает указатель выборки команд на указанную метку;

– оператор вызова подпрограммы – перемещает указатель выборки команд на указанную метку и помещает в стек возвратов ссылку на команду, которая идет вслед, выполняемую оператором вызова подпрограммы;

– оператор возврата из подпрограммы – выбирает из стека возвратов ссылку на исполняемую команду и осуществляет переход на нее. Число вложенных подпрограмм практически не ограничено

<операторы передачи управления> ::=
<оператор безусловного перехода> |
<оператор вызова подпрограммы> |
<оператор возврата из подпрограммы>

<оператор безусловного перехода> ::=

JP *<имя метки>*

<оператор вызова подпрограммы> ::=

JR *<имя метки>*

<оператор возврата из подпрограммы> ::=

RT

Пример

```
Loop_Label:
JR Sub_One ; Вызывают подпрограмму на Sub_One
JP Loop_Label ; Зацикливаются переходом к Loop_Label
Sub_One:
JR Sub_Two ; Вызывают подпрограмму Sub_Two
RT ; Возвращаются из Sub_One к оператору JP
Sub_Two:
RT ; Возвращаются из Sub_Two к оператору RT
```

1.1.3 Операторы условного перехода

5.22.3.1 Оператор условного перехода передает управление на указанные метками участки программы при выполнении определенных условий, зависящих от типа оператора. Существует также арифметический оператор условного перехода, который действует аналогично такому же в языке Фортран.

5.22.3.2 Арифметический оператор условного перехода

В качестве параметров выступают значение-источник, а также три метки. Переход на первую из них осуществляется, если источник меньше нуля, на вторую – если равен нулю, а на третью – если больше нуля.

5.22.3.3 Переход по результату исключающего «или»

Переход осуществляется в случае, когда исключающее «или» первого и второго операнда-источника не равно нулю.

5.22.3.4 Переход по результату логического «и»

Переход осуществляется в случае, когда поразрядное логическое «и» первого, и второго операнда-источника не равно нулю.

5.22.3.5 Переход по результату логического «или»

Переход осуществляется в случае, когда поразрядное логическое «или» первого и второго операнда-источника не равно нулю.

5.22.3.6 Переход по результату выделения инверсией

Переход осуществляется в случае, когда Not <источник1> And <источник2> = 0.

5.22.3.7 Переход по равенству 0

Переход осуществляется, когда значение источника равно 0. Данный оператор поддерживается для совместимости со старыми программами. Рекомендуется пользоваться переходом по сравнению двух операндов.

5.22.3.8 Переход по равенству 1

Переход осуществляется, когда значение источника равно 1. Данный оператор поддерживается для совместимости со старыми программами. Рекомендуется пользоваться переходом по сравнению двух операндов.

5.22.3.9 Переход по результату сравнения

Переход осуществляется, когда первый операнд-источник равен второму.

<операторы условного перехода> ::=

<арифметический оператор условного перехода> |
<переход по результату исключающего или> |
<переход по результату логического и> |
<переход по результату логического или> |
<переход по равенству 0> |
<переход по равенству 1> |
<переход по результату выделения инверсией> |
<переход по результату сравнения>

<арифметический оператор условного перехода> ::=

IF *<источник>, <метка>, <метка>, <метка>*

<переход по результату исключающего или> ::=

IFXR *<источник>, <источник>, <метка>*

<переход по результату логического и> ::=

IFAN *<источник>, <источник>, <метка>*

<переход по результату логического или> ::=

IFBS *<источник>, <источник>, <метка>*

<переход по равенству 0> ::=

IF0 *<источник>, <метка>*

<переход по равенству 1> ::=

IF1 *<источник> <метка>*

<переход по результату выделения инверсией> ::=

IFBC *<источник>, <источник>, <метка>*

<переход по результату сравнения> ::=

IFCP *<источник>, <источник>, <метка>*

Пример – Использование оператора условного перехода:

```
MV 0, Start ; Начальное значение цикла
MV 2456, Max ; Максимальное значение цикла
MV 1, Step ; Текущий шаг приращения
MV 0, Index ; Текущее значение счетчика
Loop:
PRD Index ; Выводят значение счетчика на экран
PRT <, >
AD Index, Step, Index ; Увеличивают счетчик
AD Step, Step, Step ; Удваивают приращение
SU Index, Max, Res
IF Res, Loop, Exit, Exit ; Сравнивают счетчик с максимумом
Exit:
PRVK ; Конец работы цикла
PRT <Stopped>
```

5.22.3.10 Оператор цикла

Оператор цикла повторяет часть программы, предшествующую ему и начинающуюся с указанной в нем метки N раз, где N – значение, указанное вторым параметром оператора.

<Оператор цикла> ::=

DO *<метка> <источник>*

Пример

```
MV 0, ADR
M1: AD 1, ADR, ADR
.....
MV 1, URK
.....
DO M1, 10
```

Максимальное значение переменной ADR 10

5.22.3.11 Оператор паузы

Оператор паузы приостанавливает выполнение программы после тестового оператора, за которым он установлен.

Пример

```
T 1
PAUSE
T 2
```

После выполнения оператора Т1 происходит остановка выполнения программы. Для выполнения оператора Т2 нужно нажать клавишу ENTER или кнопку ОК в открывшемся диалоговом окне.

2.23 Операторы ввода-вывода

Операторы ввода-вывода обеспечивают:
- обмен информацией с установкой УТК;
- анализ и вывод результатов проверки в процессе исполнения тест-программы;
- работу устройств ввода-вывода алфавитно-цифровой информации

<операторы ввода-вывода> ::=
<операторы формирования и исполнения тестовых наборов> |
<операторы задания функций контактов УТК> |
<операторы вывода алфавитно-цифровой информации> |
<операторы блокировки и восстановления режима ввода-вывода>.

2.24 Операторы формирования и исполнения тестовых наборов

Операторы формирования и исполнения тестовых наборов формируют тестовые наборы, загружают их в установку УТК, управляют установкой УТК, анализируют результаты контроля на сформированных наборах. Все операторы этой группы имеют сквозной порядковый номер (оставлено для совместимости с предыдущими версиями).

В случае отрицательного результата сравнения по тестовым наборам на дисплей выводится номер строки оператора, который зафиксировал отрицательный результат сравнения, и его порядковый номер (оставлено для совместимости). Этим осуществляется привязка информирования о результатах сравнения к той части программы, которая формирует данные наборы. Поэтому номера операторов не должны повторяться. Затем выводятся на дисплее имя группы, по которой был получен отрицательный результат сравнения эталонной информации с информацией, снимаемой с контролируемого модуля, а также список каналов установки УТК, по которым был получен отрицательный результат сравнения.

<операторы формирования и исполнения тестовых наборов> ::=
<оператор формирования тестового воздействия> |
<оператор информирования об ошибках> |
<оператор формирования воздействий по временной диаграмме>.

2.25 Оператор формирования тестового воздействия

Оператор формирует тестовое воздействие из текущих значений переменных описания групп контактов в соответствии с описанием коммутации каналов (вход-выход) и загружает его в установку УТК. Длительность воздействия – 1 такт.

<оператор формирования тестового воздействия> ::=
T *<выражение>*

2.26 Оператор информирования об ошибках

Оператор аналогичен оператору формирования тестового воздействия. Он формирует тестовое воздействие из текущих значений переменных описания групп контактов в соответствии с описанием коммутации каналов (вход-выход) и загружает его в установку УТК. Длительность воздействия – 1 такт.

Однако в отличие от оператора формирования тестового воздействия он проводит информирование об ошибках в контролируемом модуле в случае отрицательного результата сравнения эталонных значений со значениями, полученными с контактов контролируемого модуля.

<оператор информирования об ошибках> ::=
TN *<выражение>*

2.27 Оператор формирования тестовых воздействий с использованием временной диаграммы

Оператор формирует тестовое воздействие, подавая на указанные контакты участки временных диаграмм либо сравнивая на ходу получаемую с них информацию с эталоном. Информация, хранимая в переменных описания групп контактов, также подается на выходы либо сравнивается на соответствующих входах.

Содержимое переменных групп контактов на протяжении действии данного оператора не меняется. Длительность воздействия указывается первым и последним тактами из временной диаграммы. Номера тактов задаются включительно.

В случае обнаружения отрицательных результатов сравнений эталонной информации с полученной оператор проводит информирование об ошибке.

<оператор формирования воздействий по временной диаграмме> ::=
TVD *<номер оператора> <первый такт>, <последний такт>*
<операторы списка>
ISP
<номер оператора> ::= <порядковый номер оператора>
<первый такт> ::= <номер такта временной диаграммы, с которого начинается формирование тестового набора, задается включительно>
<последний такт> ::= <номер такта временной диаграммы, по которому кончается формирование тестового набора, задается включительно>
<операторы списка> ::=
<оператор списка> |
<оператор списка> ВК <оператор списка>
<оператор списка> ::=
PR *<имя диаграммы> <имя группы контактов>*

Пример

```
GR YPR1 10, 27
 GR YPR2 17
 .....
 VD ZAP1 8 2
 S <__^^___^>
 S <_^__^^__>
 VD SBR 8 1
 S <___^^^__>
 .....
 TVD 15 3, 6
 PR ZAP1, YPR1
 PR SBR, YPR2
 ISP
```

Оператор TVD с порядковым номером 15 формирует тестовые воздействия, в которых по группам контактов YPR1 и YPR2 генерируются состояния, имитирующие участки временных диаграмм ZAP1 и SBR, начиная с такта 3 и кончая тактом 6 включительно.

2.28 Оператор формирования нормированных пачек импульсов

Оператор формирует нормированную пачку импульсов (повторяющуюся последовательность из двух отсчетов) и подает ее на указанные группы контактов. Оператор TI формирует последовательность из временной диаграммы «^_», а оператор TIN – из диаграммы «_^».

<оператор формирования нормированных пачек импульсов> ::=
<оператор TI-TIN> <номер оператора> <число импульсов в пачке> <список операндов>
<оператор TI-TIN> ::=
TI | TIN
<список операндов> ::=
<переменная типа группа контактов> |
<переменная типа группа контактов> <список операндов>

Пример

```
GR GROUP 10, 14, 21
GR SYNC 11
...
TI 4, 100, GROUP, SYNC
```

Оператор формирует нормированную пачку из 100 импульсов (200 тест-наборов) на группах контактов GROUP и SYNC. Первый тест набор подает на все контакты этих групп высокое состояние (лог. 1).

2.29 Операторы задания функций контактов

5.29.1 В качестве операндов для этих оператором могут быть только переменные описания группы контактов.

5.29.2 Оператор задания входов контролируемого модуля

Указанные группы контактов служат для передачи данных из установок тестового контроля в контролируемый модуль.

5.29.3 Оператор задания выходов контролируемого модуля

Указанные группы контактов служат для передачи данных из контролируемого модуля в установки тестового контроля.

5.29.4 Оператор задания режимов сравнения

По указанным группам контактов будет осуществляться сравнение получаемой с них информации с эталонной. Группы контактов должны быть коммутированы на передачу данных из контролируемого модуля в УТК.

5.29.5 Оператор задания блокировки

По указанным группам каналов сравнение блокируется, то есть отключается. Для включения сравнения надо еще раз отдать команду задания режима сравнения по данной группе контактов. Оператор задания блокировки аддитивен, то есть существует понятие уровня блокировки: если контроль по группе контактов был два раза заблокирован, то для того чтобы вновь включить контроль, надо два раза выполнить оператор задания режима сравнения.

5.29.6 Оператор задания интегральной блокировки

Блокирует режим сравнения по всем группам контактов, по которым оно осуществляется. Это касается также заблокированных групп, коммутированных на передачу данных из модуля в УТК.

5.29.7 Оператор задания интегрального режима сравнения

Сравнение ведется по всем группам контактов, коммутированных на передачу данных из модуля в УТК (выход). В плане уровне блокировки работает аналогично оператору задания режима сравнения, то есть снимает один уровень блокировки.

<оператор задания функций контактов УТК> ::=
<оператор задания входов контролируемого модуля> |
<оператор задания выходов контролируемого модуля> |
<оператор задания режимов сравнения> |
<оператор задания блокировки> |
<оператор задания интегральной блокировки> |
<оператор задания интегрального режима сравнения>
<оператор задания входов контролируемого модуля> ::=
VX *<список групп контактов>*
<оператор задания выходов контролируемого модуля> ::=
VIX *<список групп контактов>*
<оператор задания режимов сравнения> ::=
K *<список групп контактов>*
<оператор задания блокировки> ::=
BL *<список групп контактов>*
<оператор задания интегральной блокировки> ::=
BLI
<оператор задания интегрального режима сравнения>::=
RBL
<список групп контактов> ::=
<имя переменной группы контактов> |
<имя переменной группы контактов>, <список групп контактов>

2.30 Операторы вывода алфавитно-цифровой информации

Операторы вывода алфавитно-цифровой информации выводят на консоль значения числовых выражений, переменных, а также строковые константы. Существуют также операторы для вывода специальных символов.

<операторы вывода алфавитно-цифровой информации> ::=

<операторы вывода числовых значений> |
<оператор вывода текстовой информации> |
<операторы вывода знаков>
<операторы вывода числовых значений> ::=
<оператор вывода в двоичном виде> |
<оператор вывода в восьмеричном виде> |
<оператор вывода в десятичном виде> |
<оператор вывода в шестнадцатеричном виде>
<операторы вывода знаков> ::=
<оператор вывода знака «,» (запятая)> |
<оператор вывода знака «–» (тире)> |
<оператор вывода знака « « (пробел)> |
<оператор вывода перевода строки>
<оператор вывода в двоичном виде> ::=
PRB *<источник>*
<оператор вывода в восьмеричном виде> ::=
RRV *<источник>*
<оператор вывода в десятичном виде> ::=
PRD *<источник>*
<оператор вывода в шестнадцатеричном виде> ::=
PR16 *<источник>*
<оператор вывода знака «,» (запятая)> ::=
PRZ
<оператор вывода знака «–» (тире)> ::=
PRTR
<оператор вывод знака « « (пробел)> ::=
PRPR
<оператор вывода перевода строки> ::=
PRVK
<оператор вывода текстовой информации> ::=
PRT *<строка>* |
PRT *<имя строковой константы>*

Пример

```
PRT <Начинают вывод>
PRVK
PRT <Выводят 123, 456-378>
PRVK
PRD 123
PRZ
PRPR
PRD 456
PRTR
PRD 378
PRVK
PRT <Так делали раньше, а вот так можно теперь>
PRVK
PRD 123
PRT <, >
PRD 456
PRT <->
PRD 378
; Теперь выводят переменные
MV ^O100, Variable ; Заносят 100_8 в переменную
PRB Variable ; Выводят “1000000”
PRVK
PRV Variable ; Выводят “100”
```

PRVK
PRD Variable ; Выводят "64"
PRVK
PR16 Variable ; Выводят "40"
PRVK

6 Методика автоматизированного построения тестовых программ

6.1 Методика построения модели объекта контроля

6.1.1 Исходным документом при составлении тестовой программы на языке ЯСТЕК является алгоритм контроля объектов контроля (ОК).

6.1.2 Одной из основных задач, решаемых тестовой программой, является проверка функционирования и диагностика неисправностей ОК, представляющих собой готовое к серийному производству или уже выпускаемое цифровое электронное устройство РЭА. Под цифровым электронным устройством понимают набор электронных модулей [ИМС, ПЛИС (программируемая логическая интегральная схема), микропроцессор и т. д.], каждый из которых функционирует по собственному алгоритму. Алгоритм функционирования электронного устройства определяется алгоритмами функционирования модулей, связями между модулями и временны́ми последовательностями входных сигналов.

6.1.3 С физической точки зрения цифровое электронное устройство как *ОК* представляет собой изделие, снабженное внешними (краевыми) разъемами (рисунок 1).

6.1.4 Работа устройства тактируется одним из входных сигналов либо от одного внутреннего генератора, входящего в состав устройства. На входные контакты краевых разъемов могут подаваться входные цифровые тестовые воздействия, а с выходных контактов этих разъемов может быть снята реакция (выходные цифровые сигналы) устройства на входные воздействия. Сопоставление полученных реакций с эталонными позволяет сделать вывод о работоспособности ОК и осуществлять диагностику обнаруженных неисправностей. Совокупность входных воздействий и соответствующие им эталонные реакции ОК представляют собой тест.



Рисунок 1 – Физическое представление устройства

6.1.5 Составление тестов является достаточно трудоемкой задачей. Для решения этой задачи используют систему автоматизированного построения тестов цифровых электронных устройств.

6.1.6 В процессе своей работы система сначала создает программную модель функционирования ОК, а на втором этапе формирует тесты для ОК.

6.1.7 С программной точки зрения устройство в целом как *объект контроля* представляет собой программу, моделирующую функционирование устройства. Программа реализует некоторый алгоритм, обрабатывающий входные сигналы и представляющий собой описание функционирования устройства

в целом. В рамках программы моделирования эти сигналы описываются соответствующими программными переменными, законы изменения которых также задаются специальными программами. Реакцией устройства в этом случае являются законы изменения выходных переменных, полученные в ходе работы моделирующей программы.

Входные и выходные программные переменные описывают входные и выходные сигналы реального устройства. При этом в каждый моделируемый момент времени работы эти переменные принимают определенные значения. Совокупность этих значений и является тестовым «портретом» устройства. Этот «портрет» описывает правильное соответствие входных и выходных сигналов на разъемах устройства и является тестом, который может быть загружен в установку тестового контроля. Следует отметить, что из-за наличия в схеме устройства различных обратных связей и присутствия элементов памяти тестовый «портрет» является, как правило, динамическим. При этом в качестве входного воздействия должна выступать некоторая временна́я последовательность входных сигналов, а выходная реакция должна содержать значения, которые принимают выходные сигналы, после окончания этой входной последовательности. Другими словами, каждой *i*-й реакции ОК соответствует своя последовательность *i* входных воздействий (1 ... *i*).

Статический тестовый «портрет» характерен для относительно простых устройств без памяти. В этом случае каждому входному воздействию однозначно соответствует своя реакция ОК независимо от того, в какой момент времени подано входное воздействие. При этом статический тестовый «портрет» представляет собой таблицу, где графы содержат значения как входных, так и выходных сигналов, а каждая строка соответствует состоянию сигналов в моделируемые моменты времени.

Примерный вид динамического и статического тестовых «портретов» приведен в таблицах 1 и 2 соответственно.

Таблица 1 – Динамический тестовый «портрет»

Время	Входной сигнал 1	Входной сигнал 2	...	Входной сигнал n	Выходной сигнал 1	Выходной сигнал 2	...	Выходной сигнал m
T_1	IS^1_1	IS^2_1	...	IS^n_1	*	*	...	*
T_2	IS^1_2	IS^2_2	...	IS^n_2	*	*	...	*
....	...	...	...	...	...	...	...	....
T_k	IS^1_k	IS^2_k	...	IS^n_k	OS^1_1	OS^2_1	...	OS^m_1
T_{k+1}	IS^1_{k+1}	IS^2_{k+1}	...	IS^n_{k+1}	*	*	...	*
T_{k+2}	IS^1_{k+2}	IS^2_{k+2}	...	IS^n_{k+2}	*	*	...	*
....	...	...	...	...	...	...	...	....
T_{k+h}	IS^1_{k+h}	IS^2_{k+h}	...	IS^n_{k+h}	OS^1_2	OS^2_2	...	OS^m_2
...	...	...	...	...	...	...	...	...

Примечание – В настоящей таблице использованы следующие условные обозначения:
T – дискретное время;
IS – входной сигнал;
OS – выходной сигнал;
* – значение сигнала не рассматривается.

Нижний индекс соответствует номеру отсчета времени. Верхний индекс обозначает номер соответствующего сигнала.

Динамический «портрет» является частным случаем статического. Он может быть получен из него, если просто игнорировать значения выходных сигналов во время подачи входной последователь-

ности и запоминать их значения только по завершении входной серии. Однако следует заметить, что при формировании динамического портрета для получения следующей комбинации выходных сигналов входная последовательность сигналов должна продолжаться. Таким образом, динамический «портрет» представляет собой непрерывную последовательность входных сигналов, для которой выходные сигналы фиксируются только в определенные моменты времени.

Таблица 2 – Статический тестовый «портрет»

Номер комбинации	Входной сигнал 1	Входной сигнал 2	...	Входной сигнал n	Выходной сигнал 1	Выходной сигнал 2	...	Выходной сигнал m
1	IS^1_1	IS^2_1	...	IS^n_1	OS^1_1	OS^2_1	...	OS^m_1
2	IS^1_2	IS^2_2	...	IS^n_2	OS^1_2	OS^1_2	...	OS^1_2
....	...	...	...	...	...	...	...	....
k	IS^1_k	IS^2_k	...	IS^n_k	OS^1_k	OS^2_k	...	OS^m_k
k+1	IS^1_{k+1}	IS^2_{k+1}	...	IS^n_{k+1}	OS^1_{k+1}	OS^2_{k+1}	...	OS^m_{k+1}
k+2	IS^1_{k+2}	IS^2_{k+2}	...	IS^n_{k+2}	OS^1_{k+2}	OS^1_{k+2}	...	OS^1_{k+2}
....	...	...	...	...	...	...	...	....
k+h	IS^1_{k+h}	IS^2_{k+h}	...	IS^n_{k+h}	OS^1_{k+h}	OS^2_{k+h}	...	OS^m_{k+h}
...	...	...	...	...	...	...	...	...

Примечание – В настоящей таблице использованы следующие условные обозначения:
IS – входной сигнал;
OS – выходной сигнал.

Как правило, электронные устройства имеют вход сброса, по которому они приводятся в некоторое определенное начальное состояние. Поэтому для получения набора последующих «портретов» необходимо выдавать на устройство этот сигнал или (в случае моделирования) устанавливать и сбрасывать соответствующую переменную, с помощью которой программная модель приводится в начальное состояние. Получение набора «портретов» в этом случае будет всегда начинаться с одного и того же начального состояния, являющегося к тому же первой строкой в таблице «портрета».

6.1.8 Система автоматизированного построения тестов должна на основании входной информации об устройстве получать тестовые «портреты» устройства, обеспечивающие приемлемое качество тестирования устройства.

6.1.9 Основными компонентами системы автоматизированного построения тестов являются:

– база данных (электронная библиотека) устройств и их компонент;
– анализатор структуры и интегратор описания;
– генератор тестовых последовательностей;
– генератор скриптов моделирования;
– симулятор;
– анализатор результатов моделирования.

Взаимосвязь компонентов системы автоматизированного построения тестов показана на рисунке 2.

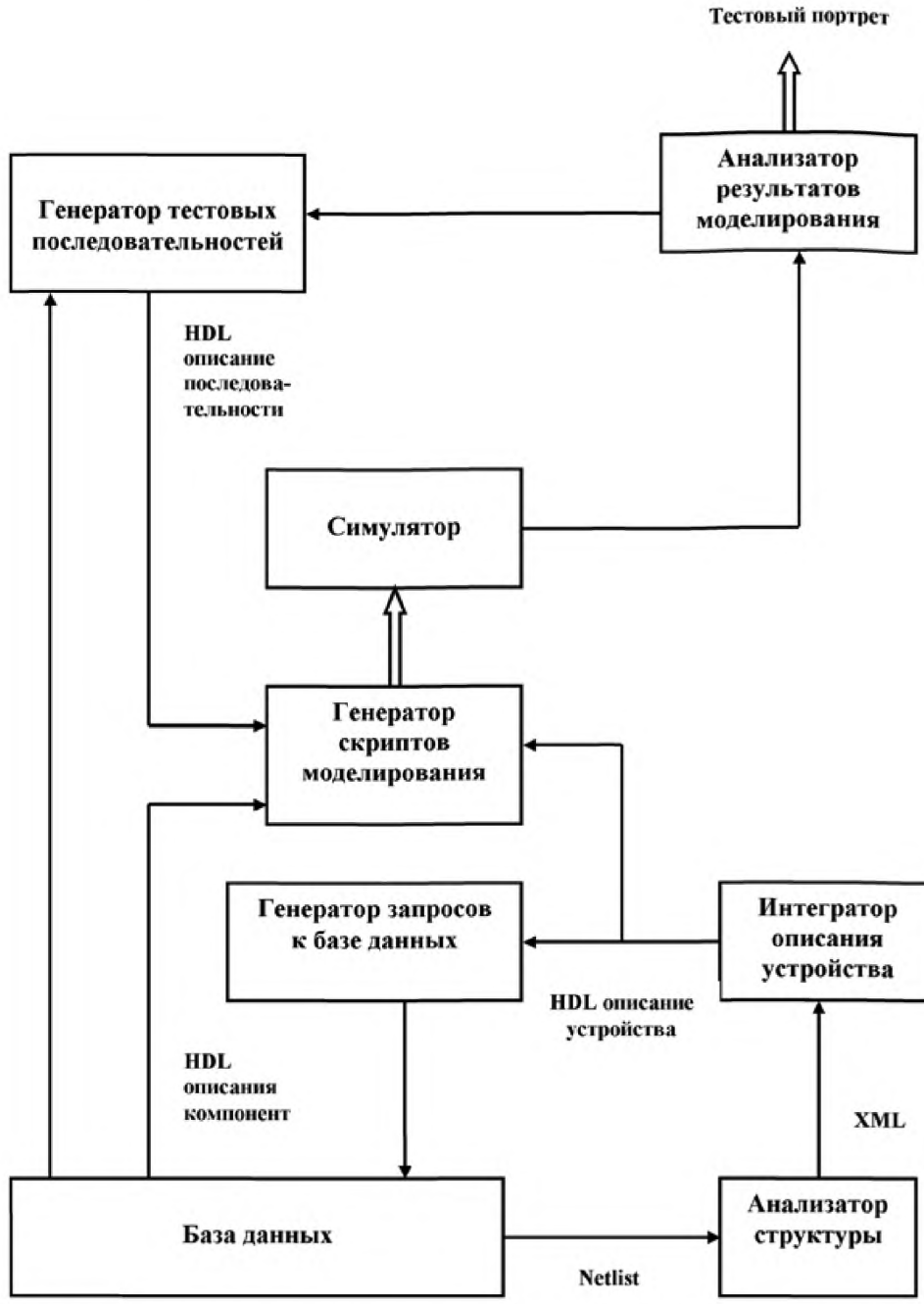

Рисунок 2 – Взаимосвязь компонентов системы автоматизированного построения тестов

6.1.10 Входные данные

Входной информацией, с которой работает система автоматизированного построения тестов, являются описания функционирования компонент электронного устройства (поведенческие модели компонент) и описание структуры электронного устройства.

Поведенческие модели компонент представляют собой тексты программных модулей, написанные на высокоуровневом языке описания аппаратуры (Hardware Description Language – HDL). В настоящее время предполагается использование в качестве описаний компонент языка Verilog HDL.

Описание структуры устройства содержится в файле схематики в формате ASCII.SCH, формируемом с помощью САПР PCAD (ORCAD) на этапе разработки устройства. Этот файл служит для получения информации о составе электронного устройства – наименований внутренних элементов и сигналов, их связей между собой, а также с внешними разъемами и прочими необходимыми атрибутами (формат NetList).

6.1.11 База данных

Центральным компонентом системы является база данных. С ее помощью осуществляется хранение исходных данных, в нее записываются все необходимые промежуточные и окончательные результаты обработки и моделирования. При разработке базы данных предполагается, что каждый компонент, а также устройства имеют уникальные имена.

База данных служит для синхронизации работы всех программных средств, так как данные для моделирования извлекаются и помещаются в нее с помощью генератора запросов, который в соответствии с текущим этапом обработки формирует тот или иной запрос на необходимое в данный момент действие.

В результате работы в базе данных накапливается информация об электронных устройствах, их компонентах, тестовых «портретах». При создании тестов для новых устройств сначала происходит поиск в базе данных на наличие в ней всех компонентов, входящих в состав этого устройства. При отсутствии какого-либо компонента запускается процедура добавления информации о недостающем компоненте. Следует отметить, что любое устройство, занесенное в базу данных (вместе со сгенерированными тестовыми портретами), с информационной точки зрения ничем не отличается от элементарных компонент. Поэтому оно в дальнейшем может использоваться как составная часть более сложных устройств:

/i/i[@v="netlist"]/i[@v="net" and i[@v="attr"]/p[@v="Input"]]/p/@v

/i/i[@v="netlist"]/i[@v="net" and i[@v="attr"]/p[@v="Output"]]/p/@v

Список всех компонентов, подключенных к выбранной линии, номера их контактов:

/i/i[@v=»netlist»]/i[@v=»net»and [@v=»<название_линии>»]]/i[@v=»node»]

Интегратор описания устройства получает информацию с помощью указанных запросов и строит на их базе общее описание устройства на языке Verylog, учитывающее связи между всеми компонентами устройства, наименования этих компонентов и информацию об их сигнальных контактах.

6.1.12 Генератор тестовых последовательностей

Генератор тестовых последовательностей по некоторым унифицированным параметрам строит входные тестовые последовательности в виде таблицы: дискретное время – значения входных сигналов, которая может быть сохранена в виде файла, с сопутствующими атрибутами. Тестовые последовательности строятся так, чтобы, по возможности, проверить все одиночные, константные неисправности 0 и 1 на контактах внешних разъемов устройства и на контактах входящих в него его модулей. Затем эта таблица входных тестовых последовательностей преобразуется в HDL описание тестовых стимулов, представляющее собой программный модуль на HDL языке. Такое описание необходимо для работы симулятора.

6.1.13 Генератор скриптов моделирования

Генератор скриптов моделирования формирует необходимые команды управления работой симулятора. К этим командам относятся:

- подготовительные команды работы с симулятором;
- команды загрузки HDL модулей – главный модуль описания устройства, модули описания компонент устройства и модуль описания тестовых стимулов;
- команды компиляции модулей;
- команды перехода в симулирующий режим;
- команды настройки сбора статистики моделирования;
- команды запуска и остановки моделирования;
- команды ввода-вывода данных;
- команды профилирования.

Совокупность команд образует скрипт, управляющий последовательностью запуска и работы одного прогона модели в пакетном режиме.

6.1.14 Симулятор

Симулятор служит для моделирования функционирования электронного устройства. Симулятор позволяет на основе описания устройства на каком-либо HDL языке и входных воздействий получить полную картину изменения во времени состояния всех сигналов, как внутренних, так и внешних. Таким

образом, результатом работы симулятора являются динамические и статические «портреты» моделируемого устройства.

В качестве симулятора выбран программный пакет ModelSim компании Mentor Graphics. В настоящее время этот пакет является самой распространенной системой HDL-моделирования. В пакете реализована полная поддержка всех основных стандартов языков VHDL и Verilog и их расширений. В данном симуляторе предусмотрена поддержка библиотек всех ведущих фирм-изготовителей как ПЛИС семейств FPGA (Field Programmable Gate Array) и CPLD (Complex Programmable Logic Device), так и ASIC (Application-Specific Integrated Circuit).

Данный симулятор имеет открытую архитектуру, что позволяет интегрировать его с разрабатываемой системой. Пользователь может выполнять все этапы моделирования в рамках разрабатываемой системы для построения тестов. Средства управления пользовательским интерфейсом Tcl (Tool command language) и Tk (Tool kit) предоставляют возможность организации прямого доступа к моделирующему ядру симулятора, загрузки информации о выполнении процесса моделирования и его результатов в базу данных, а также возможность управления работой симулятора через интерфейс применяемых средств. Симулятор может работать как в интерактивном, так и в пакетном режиме.

Следует отметить, что в этом симуляторе имеется индикатор активности кода, позволяющего быстро создавать полные тестовые последовательности. Этот инструмент предоставляет возможность проследить строки исходного HDL кода устройства, которые не «активизировались» в процессе моделирования, что дает возможность организовать управление входными тестовыми последовательностями и оценить глубину тестирования устройства. Индикатор активности кода может быть использован как на уровне отдельного блока, так и для всей системы в целом.

Пакет ModelSim существует как в виде коммерческого продукта, так и в виде свободно распространяемой версии. Свободная версия имеет ряд ограничений, но предварительное ее использование показало, что эти ограничения не являются существенными для разрабатываемой системы. Поэтому в качестве симулятора и была выбрана свободно распространяемая версия пакета.

6.1.15 Анализатор результатов моделирования

Этот компонент системы осуществляет анализ полученных в ходе одного прогона моделирования тестовых «портретов» устройства и принимает решение об их состоятельности. Он вырабатывает также команды для генератора тестовых последовательностей, с помощью которых обеспечивается изменение параметров генерации входных сигналов. Затем весь процесс подготовки и запуска моделирования начинается заново с новыми входными последовательностями. Таким образом, обеспечивается обратная связь между выходом и входом системы, с помощью которой реализуются алгоритмы поиска состоятельных тестовых «портретов».

Все компоненты системы автоматизированного построения тестов работают в тесном взаимодействии друг с другом. Результатом работы системы применительно к конкретному ОК являются его тестовые «портреты» устройства. На их основе в дальнейшем формируются тест-программы на языке ЯСТЭК для использования в установках тестового контроля и диагностики.

3.2 Пример составления тестовой программы на языке системы тестового контроля ЯСТЭК

```
TP <Тест установки УТК> <UTK>
 GR
 GK RE0 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31
 GK RE1 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30
 GK RE2 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63
 GK RE3 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62
 GK RE4 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95
 GK RE5 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94
 GK RE6 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127
 GK RE7 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126
 GK RE8 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159
 GK RE9 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158
 GK REA 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191
 GK REB 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190
 GK REC 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223
```

```
 GK RED 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222
 GK REE 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255
 GK REF 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254
 KGR
KOT0:WD 0
KOT1:WD 0
 VD V1 16 16
 S <^_____________>
 S <_^____________>
 S <__^___________>
 S <___^__________>
 S <____^_________>
 S <_____^________>
 S <______^_______>
 S <_______^______>
 S <________^_____>
 S <_________^____>
 S <__________^___>
 S <___________^__>
 S <____________^_>
 S <_____________^>

 NACH
 ;SFN FB1 KOT0
M: ; Четные контакты – входы, нечетные — выходы
 VX RE0, RE2, RE4, RE6, RE8, REA, REC, REE
 VIX RE1, RE3, RE5, RE7, RE9, REB, RED, REF
 ; Контроль ведется по всем каналам
 K RE0, RE1, RE2, RE3, RE4, RE5, RE6, RE7
 K RE8, RE9, REA, REB, REC, RED, REE, REF
M1: ; Подают 300 циклов бегущей единицы
 FN FB1 KOT0
 MV KOT0, RE0, RE1, RE2, RE3, RE4, RE5, RE6, RE7
 MV KOT0, RE8, RE9, REA, REB, REC, RED, REE, REF
 T 1
 DO M1, 300
 ; Переключают каналы входа на выход и наоборот
 VIX RE0, RE2, RE4, RE6, RE8, REA, REC, REE
 VX RE1, RE3, RE5, RE7, RE9, REB, RED, REF
 ; Первый проход этого цикла – четные = входы, нечетные = выходы,
 ; а второй проход – наоборот
 DO M1, 2
 ; для контроля прогоняют бегущую единицу еще раз
 DO M, 2
 ; осуществляют прогон несколько раз
 T 121
 T 122
 T 123
 T 124
 T 125
 ; Подают бегущую единицу на все каналы
 ; посредством временнóй диаграммы
 TVD 5 1, 16
 PR V1, RE0
```

PR V1, RE1
PR V1, RE2
PR V1, RE3
PR V1, RE4
PR V1, RE5
PR V1, RE6
PR V1, RE7
PR V1, RE8
PR V1, RE9
PR V1, REA
PR V1, REB
PR V1, REC
PR V1, RED
PR V1, REE
PR V1, REF
ISP
M2: KN

3.3 Методика отладки тестовой программы

6.3.1 Отладку тест-программы на языке Ястек следует проводить в три этапа:

– устранение синтаксических ошибок тест-программы;

– автономная отладка тест-программы без подключения модуля к установке УТК;

– комплексная отладка тест-программы совместно с подключенным к установке УТК модулем.

6.3.2 Тест-программа, разрабатываемая на языке ЯСТЕК, имеет строго определённую синтаксическую структуру операторов и программы, благодаря чему синтаксические ошибки обнаруживаются с помощью программ обеспечения трансляции. Информация об ошибках выдается пользователю ПЭВМ. Сообщается номер оператора и тип ошибки.

Для семантической (содержательной) отладки тест-программы имеется набор отладочных процедур-директив. Директивы отладчика приведены в таблице 3.

Таблица 3

Номер директивы	Директивы отладчика (режим работы отладчика)	Краткое описание
1	Блокировка печати	Блокируется выдача отрицательных результатов подачи тестовых воздействий
2	Остановка по номеру тестового оператора	Тест-программа останавливается по заданному пользователем номеру тестового оператора
3	Остановка по заданному числу повторений тестового оператора с номером К	Тест-программа останавливается после заданного числа повторений тестового оператора с номером К
4	Остановка по счетчику тестовых воздействий	Тест-программа останавливается после того, как число тестовых воздействий станет равным заданному оператором
5	Остановка по первому непрошедшему тесту	Тест-программа останавливается после первого отрицательного результата тестового воздействия
6	Зациклить программу	Указанным оператором участок тест-программы повторяется определенное время
7	Установить точки остановки	Оператор может задать адрес, в котором работа тест-программы будет остановлена
8	Трассировка тест-программы	Оператор может указать адрес в тест-программе, прохождение через который будет сопровождаться его индикацией на экране дисплея

Окончание таблицы 3

Номер директивы	Директивы отладчика (режим работы отладчика)	Краткое описание
9	Просмотр и изменение содержимого памяти	Оператор указывает адреса в тест-программе, содержимое которых надо просмотреть или изменить
10	Остановка по реакции с проверяемого объекта	Произойдет останов тест-программы в случае совпадения реакции проверяемого объекта с заданной оператором по определенной группе контактов
11	Остановка по ключевым словам	Исполнение тест-программы остановится в случае сравнения содержимого со значением, заданным оператором
12	Динамический вывод значений переменных	В процессе исполнения тест-программы на экран дисплея выводятся значения переменных в виде временных диаграмм и чисел в шестнадцатеричной, восьмеричной, двоичной, десятичной системах
13	Работа в режиме логического анализатора	Происходит высвечивание временной диаграммы по каналам (устанавливается оператором)

Работа с использованием директив ведется в режиме диалога, который осуществляется через дисплей ПЭВМ. Отладочные средства системы выдают запрос, на который оператор отвечает одной из директив. Заданная директива сопровождается сообщением, идентифицирующим ее функцию.

В комплект математического обеспечения на языке ЯСТЕК входит инструкция по эксплуатации, в которой изложены правила работы в режиме отладки с директивами и подробно описаны сами директивы.

6.3.3 В режиме отладки тест-программы без подключения модуля к УТК целесообразно использовать директиву 12. Используя различные типы остановов и выдачи значений переменных, необходимо проверить правильность хода выполнения тест-программы, особенно если тест-программа содержит циклы и условные переходы. Следует проверит правильность формирования диаграмм управляющих сигналов, используя режим динамического вывода значений переменных в виде временны́х диаграмм по группам, на которых формируются управляющие сигналы

6.3.4 В режиме отладки тест-программы с подключенным к УТК модулем, для контроля которого написана тест-программы, нужно использовать заведомо исправный модуль. Начинать отладку следует, используя директиву 5.

После останова по первому непрошедшему тестовому воздействию необходимо проверить правильность задания законов изменения выходных реакций модуля на этом тестовом воздействии. В случае невозможности обнаружения ошибки на проверяемом тестовом воздействии целесообразно перейти к работе в режиме логического анализатора. Помещая щупы, подключенные к свободным каналам установки УТК, в различные точки модуля, режим позволяет наблюдать изменение сигналов в этих точках на экране дисплея в процессе исполнения тест-программы. После останова возможно осуществить дальнейший пуск тест-программы и просмотр следующей информации. Используя директивы останова тест-программы по номеру тестовых воздействий, счетчику тестового воздействия и др., следует локализовать и устранить ошибку в тест-программе. После корректировки тест-программы следует вернуться к работе с использованием директивы 5. Процедуру необходимо повторять до тех, пока тест-программа не будет полностью отлажена.

УДК 681.3.023-182.77 ОКС 35.080 Э67 ОКП 50 1250 ОКС 35.240

Ключевые слова: электронные модули, методы, составление и отладка тест-программ, автоматизированный тестовый контроль, язык системы тестового контроля, ЯСТЕК, установка тестового контроля

Подписано в печать 01.11.2014. Формат 60х841/8.

Усл. печ. л. 3,72. Тираж 43 экз. Зак. 3975.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ФГУП «СТАНДАРТИНФОРМ»
123995 Москва, Гранатный пер., 4.

gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru