МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Утверждаю «___» _____ 20__ г.

Зав. кафедрой САПР ВС _____
(подпись)

ЗАДАНИЕ

на бакалаврскую выпускную квалификационную работу

Студент ***Костенко Никита Михайлович, группа 549***

1. Тема выпускной квалификационной работы бакалавра «***Разработка виртуальной отладочной платы для библиотеки SystemC»***

2. Срок сдачи студентом законченной выпускной квалификационной работы бакалавра

25 июня 2019 года

3. Руководитель бакалаврской выпускной квалификационной работы ***Шибанов Владимир Александрович к.т.н., доцент, ФГБОУ ВО РГРТУ имени В.Ф. Уткина каф. САПР ВС, г.Рязань***
(фамилия, имя, отчество полностью, место работы, должность)

4. Исходные данные к выпускной квалификационной работе бакалавра _____

4.1. Язык программирования C++.

4.2. Библиотека Qt.

4.3. Язык описания аппаратуры SystemC

4.3.1. Библиотека SystemC версия 2.2

4.4. Среда разработки Qt Creator

5. Содержание расчетно-пояснительной записки

Введение.

5.1. Технико-экономическое обоснование.

5.2. Теоретическая часть.

5.3. Алгоритмическая часть.

5.3.1. Разработка алгоритмов.

5.3.2. Разработка программ.

5.4. Разработка технической документации.

5.4.1. Описание применения

5.4.2. Руководство системного программиста.

5.4.3. Руководство программиста.

5.4.4. Руководство оператора.

5.5. Контрольные примеры.

Заключение.

6. Перечень графического материала (с точным указанием обязательных чертежей)

6.1.Постановка задачи - 1 лист А3;

6.2. Теоретическая часть — 1 лист А3;

6.3. Схемы алгоритмов — 2 листа А3;

6.4. Экспериментальные результаты — 1 лист А3;

6.5. Демонстрационная версия программы;

7. Консультанты по выпускной квалификационной работе бакалавра (с указанием относящихся к ним разделов работы)

Дата выдачи задания « ___ » ***мая*** ***2018 г.***

Руководитель _____
(подпись)

Задание принял к исполнению « ___ » ***мая*** ***2018 г.***

Подпись студента _____

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Рязанский государственный радиотехнический университет имени В.Ф.Уткина»

Факультет: ФВТ
Направление: 11.03.03

К защите
Зав. кафедрой САПР ВС _____
« ___ » ____июня____ 2019 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе на тему

«РАЗРАБОТКА ВИРТУАЛЬНОЙ ОТЛАДОЧНОЙ ПЛАТЫ ДЛЯ БИБЛИОТЕКИ SYSTEMC»

Студент _____(__*Костенко Н.М*_____)

Руководитель ВКР _____(__*Шибанов В.А*_____)

« __ » __июня__ 2019 г.

Содержание

ВВЕДЕНИЕ

Отладочная плата (development board) — это тип печатной платы , используемый для изучения работы конкретного электронного компонента и макетирования на ее основе устройств. Обычно такие платы содержат в себе отлаживаемый компонент — это может быть ПЛИС микросхема или микроконтроллер, а также дополнительный набор компонентов, необходимый для достаточно простой работы отлаживаемым компонентом. Примерами отладочных плат являются: платы проекта Arduino, отладочные платы Intel FPGA (на основе ПЛИС Altera),отладочные платы OLMEX.

Пример отладочной платы приведен на рис 1.

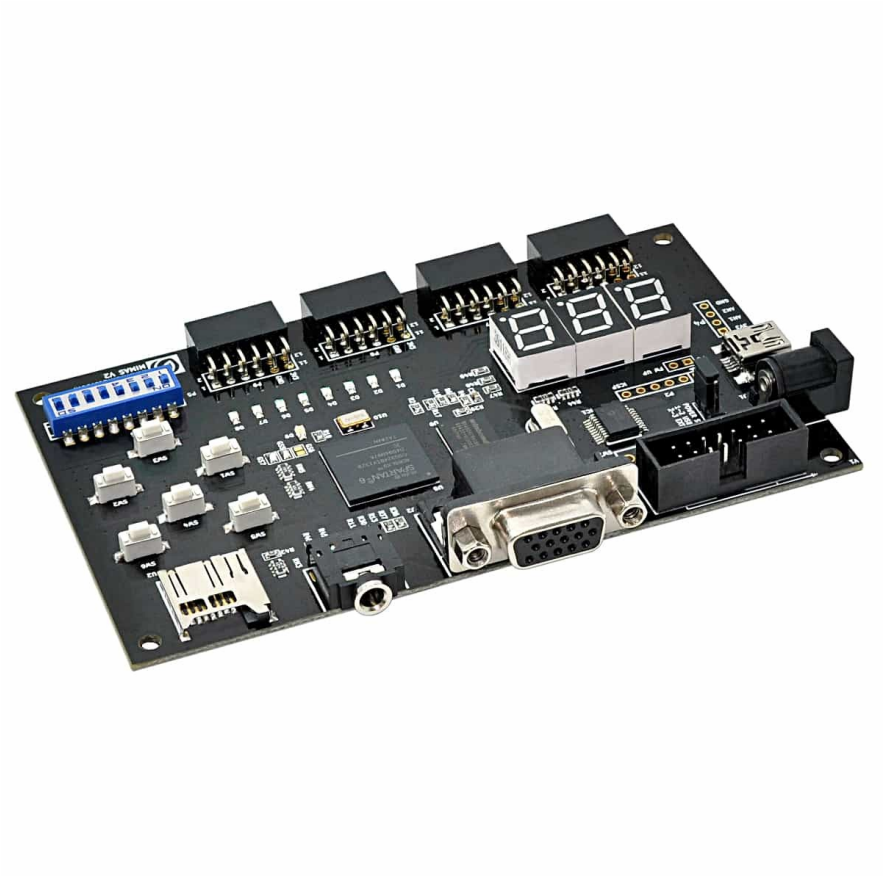

Рис. 1. Отладочная плата Mimas V2 Spartan 6 FPGA

Отладочные платы для ПЛИС могут содержать в себе самые разные компоненты:

Последовательные интерфейсы для передачи данных:

- USB;
- Serial ATA;

- PCI Express;
- Ethernet;
- Графические интерфейсы;
- DisplayPort;
- VGA;
- Контроллеры памяти;
- Различные разъемы, переключатели, индикаторы.

Для описания электронной схемы ПЛИС используются специальные языки описания аппаратуры (hardware description language или HDL).

Verilog — Язык описания электронной аппаратуры часто используемый в при проектировании СБИС, описания аналоговых(Для этого используется расширение Verilog-AMS) и цифровых электрических схем. Данный язык был разработан Филом Морби в 1985 году. Данный язык описывается стандартом IEEE 1364.

SystemVerilog — Язык описания аппаратуры основанный на Verilog использующийся для сложных систем.

VHDL — Язык описания интегральных схем. Широко используется при разработке вычислительных устройств. Разработан в 1983 году по заказу Министерства Обороны США Данный язык описывается стандартом IEEE 1076.

SystemC — Особенность данного языка является, то что он представлен в виде библиотеки для языка C++ с открытым исходным кодом. В данном языке при этом нет запрета на использования возможностей C++. Данный язык в основном используется для моделирования систем, однако существуют средства синтеза схем написанных для данной библиотеки. SystemC описывается стандартом IEEE 1666.

JHDL — Также как и SystemC является библиотекой для языка программирования java.

При моделировании устройств описанных на данных языках используются специальные программные средства. Однако данные программные средства

являются не настолько наглядными для изучения, как использование макетной платы.

Целью данной выпускной квалификационной работы является разработка программы для моделирования, представленной в виде виртуальной отладочной платы, которая упрощала бы изучение программирования схем для ПЛИС, для UNIX-подобных операционных систем.

1. ТЕХНИКО ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

Значительное количество существующих средств для моделирования. Таких как ModelSim являются проприетарными, что вводит определенные ограничения при работе с ними. И они бывают порой не на столько наглядны для того, чтобы работу схемы на конкретном устройстве.

Для данных целей имеются также открытые проекты. Однако их недостатком довольно часто является значительная сложность при взаимодействии и не всегда хорошая поддержка со стороны производителей ПЛИС и макетных плат.

SystemC имеет в себе средства моделирования, однако взаимодействие с пользователем в данном языке необходимо необходимо разрабатывать самому, вместе с описанием самого устройства

Решением данной задачи является разработка программы с графическим интерфейсом которая возьмет на себя взаимодействие с моделью устройства для библиотекой SystemC, и будет предоставлять пользователю понятный интерфейс взаимодействия с данной моделью в виде виртуальной отладочной платы.

Такое решением является разрабатываемая в данной выпускной квалификационной работе программа: Виртуальная отладочная плата qtvirtboard.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.

2.1. Библиотека SystemC.

Рассмотрим по подробнее язык SystemC.

Базовым понятием языка SystemC является модуль. Модуль — это описание устройства в SystemC, которое представляет собой класс языка C++. Модуль может включать в себя порты, каналы, процессы, события, и другие модули. Порты нужны для передачи данных из модуля или в модуль. Каналы служат для связи модулей между собой. Процессы — это функции, которые описывают поведения модуля.

При моделировании в SystemC используется модули, которые называются тестерами(testbench). Такие модули соединяются сигналами с моделируемым модулем в основном теле программы моделирования в функции sc_main() и используется для подачи данных на моделируемый модуль и получения выходных данных от моделируемого модуля.

Библиотека SystemC использует дискретно-событийного моделирование. Моделирование в библиотеке проходит по следующим этапам:

1. Инициализация — выполняются все процессы.

2. Оценочный этап — выбор процесса, который который готов к исполнению и запустить его выполнение. В случае с SC_METHOD выполняется вся функция целиком, в случае с SC_THREAD с предыдущего вызова wait(). Это может вызвать события немедленно, что может привести появлению дополнительных процессов готовых к запуску на этом же этапе.

3. Если есть еще готовы процессы. Перейти к шагу 2.

4. Этап обновления.

5. Если есть отложенные события, определите какие процессы готовы к запуску из-за отложенных событий и перейдите к шагу 2.

6. Если нет больше событий остановить моделирование.

7. Продвижение текущего модельного времени моделирования до самого раннего события.

8. Определение, какие процессы готовы к запуску в связи с событиями. Перейти к шагу 2.

2.2. Библиотека Qt

Qt — это объектно-ориентированная библиотека C++ для создания платформо-независимых программ с графическим интерфейсом пользователя. Данная библиотека имеет обширную документацию и имеет открытый исходный код, что значительно упрощает разработку под нее. Qt доступна для многих устройств: от персонального компьютеров до мобильных телефонов, и для большого числа операционных систем: linux, Windows, macOS, FreeBSD и т.д. Существуют также версии библиотек для других языков — например PyQt для Python.

Данная библиотека хорошо задокументирована и имеет открытый исходный код.

По мимо возможностей по созданию графических интерфейсов: у библиотеки Qt есть следующие возможности:

- работа с сетью,
- работа с файлами,
- работа с процессами,
- поддержку программирования баз данных базами данных.
- и другие возможности.

По этим и многим причинам библиотека Qt получила довольно большую популярность у разработчиков.

2.3. Стандарт POSIX.

POSIX(Portable Operating System Interface) - Это группа стандартов стандартов разрабатываемые организацией IEEE, для совместимости различных операционных систем. Изначально разрабатывался для unix-подобных систем, но может применяться для других операционных систем. Данный стандарт описывает в том числе программный интерфейс приложений для языка Си.

В стандартах описывается программный интерфейс приложений.

POSIX поддерживается следующими операционными системами:

Linux;

FreeBSD;

macOS;

Данный стандарт не поддерживается Windows, однако существуют проекты по добавлению окружения POSIX для данного семейства операционных систем. Самый известный из них Cygwin

В данной работе будут применяться следующие возможности POSIX описанные для языка Си:

Работа с файлами

Сигналы

Таймеры

2.3.1. Работа с файлами в POSIX

Для работы с файлами в POSIX используют понятие дескрипторов.

Дескриптор — это целое неотрицательное число соответствующее некоторому открытому файлу. Помимо обычных дескрипторов в POSIX, есть дескрипторы соответствующее стандартному потоку ввода STDIN_FILENO, вывода STDOUT_FILENO и ошибок STDERR_FILENO для данного процесса.

Работа с файлам происходит с использованием следующих функций: open, creat, read, write, select.

Функция open() описана в заголовочном файле fcntl.h:

Через данную функцию можно получить дескриптор файла в файловой системе.

```
int open(const char * pathname, int flags, mode_t mode);
```

Аргумент `pathname` имеет строковый тип и содержит имя файла, который необходимо открыть.

Аргумент `mode` имеет перечислимый тип и содержит режим в котором откроется файл: O_RDONLY — только для чтения, O_WRONLY — только для записи, O_RDWR — для чтения или записи.

В аргументе `flags` целого типа указываются дополнительные опции через операцию побитового ИЛИ. Некоторые из них

O_TRUNC — очистить файл, если он существует и открывается;

O_CREAT — создать файл, если он не используется;

O_APPEND — дописать в конец файла;

Возвращает дескриптор файла в случае успеха и значение -1 в случае если файл открыть не удалось.

Функция creat() описана в заголовочном файле fcntl.h:

```
int creat(const char * pathname, mode_t mode);
```

Данная функция эквивалентна вызову функции:

```
int open( pathname, O_WRONLY | O_TRUNC | O_APPEND, mode);
```

Функция read() описана в заголовочном файле unistd.h:

```
int read( int filedes, void * buf, size_t nbytes);
```

С помощью данной функции выполняется чтение данных из файла.

Аргумент `filedes` целочисленного типа в котором указывается дескриптор файла из которого необходимо считать данные.

Аргумент `buf` принимает указатель на область памяти в которую будут читаться данных.

В аргументе `nbytes` указывается максимальное количество байт, которые будут будут получены из файла данной функцией.

Возвращает количество полученных байт в случае успеха и значение -1 в случае если не удалось прочесть данные.

Функция write() описана в заголовочном файле unistd.h:

```
ssize_t write( int filedes, void * buf, size_t nbytes);
```

Аргумент `filedes` целочисленного типа в котором указывается дескриптор файла из которого необходимо считать данные.

Аргумент `buf` принимает указатель из которой будут писаться данные в файл.

В аргументе `nbytes` указывается количество байт, которые будут записаны в файл.

Возвращает количество записанных байт в случае успеха и значение -1 в случае если не удалось прочесть данные.

Функция select() описана в заголовочном файле sys/select.h:

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);
```

Данная функция предназначена для мультиплексированного ввода данных. Она ожидает в течении определенного времени, когда какой нибудь файловый дескриптор будет готов к операции. После того как какой либо дескриптор готов к операции, ожидание прекращается.

Тип fd_set — представляет собой набор файловых дескрипторов. Для работы этим типом используют функции:

void FD_ZERO(fd_set * fdset) — сбрасывает набор fdset;

void FD_SET(int fd, fd_set * fdset) — добавляет дескриптор fd в набор fdset;

int FD_ISSET(int fd, fd_set * fdset) — возвращает 1 если дескриптор fd есть в наборе fdset, 0 иначе;

void FD_CLR(int fd, fd_set * fdset) — удаляет дескриптор fd из fdset;

У функции select() три аргумента данного типа:

`readfds` — Здесь указываются дескрипторы, для которых будет ожидаться готовность к чтению.

`writefds` — Здесь указываются дескрипторы, для которых будет ожидаться готовность к записи.

`expectfdss` — Здесь указываются дескрипторы, для которых будет ожидаться исключительная ситуация.

Структура timeval служит для хранения временных значений. Поля структуры:

tv_sec целого типа. В нем указываются секунды;

tv_usec целого типа. В нем указываются микросекунды;

Аргумент timeout задает максимальное время ожидания. Если аргумент равен NULL, то время ожидания не ограничено. Если tv_sec и tv_usec равны 0, то возвратить значение без ожидания.

Аргумент `nfds` должен быть на единицу больше максимального по значению дескриптора.

Функция возвращает количество готовых к операциям дескрипторов в случае успеха и -1 в случае ошибки.

2.3.2. Сигналы POSIX.

Сигналы представляют собой программные прерывания которые могут быть посланы другим процессом или самой программой. Они могут быть перехвачены, проигнорированы и обработаны по умолчанию. Если сигнал был перехвачен, то выполняется его обработка, если проигнорирован, то не делается ничего.

В данной работе для управления обработкой сигналов используется функция sigaction():

```
int sigaction(int signum, const struct  sigaction *act,
              struct sigaction *oldact);
```

Данная функция позволяет назначить обработчик для определенного сигнала `signum`.

Для этого используется аргумент `act`, который является структурой `sigaction`:

```
struct sigaction {
              void     (*sa_handler)(int);
              void     (*sa_sigaction)(int, siginfo_t *, void *);
              sigset_t   sa_mask;
              int        sa_flags;
};
```

Поле `sa_handler` хранит указатель на функцию-обработчик.

Поле `sa_sigaction` хранит указатель на альтернативный обработчик. Используется только при установленном флаге `SA_SIGINFO`. Данный обработчик может получать дополнительные данные через структуру siginfo_t.

Поле `sa_mask` хранит набор сигналов, которые будут блокироваться при обработке данного сигнала

Поле `sa_flags` необходимо для установки дополнительных параметров.

2.3.3. Таймеры POSIX.

В данной работе используются интервальные таймеры POSIX. Для использования таймеров предназначены функции timer_create(), timer_settime(), timer_delete().

Функция timer_create() нужна для создания нового интервального таймера:

```
int timer_create(clockid_t clockid, struct sigevent *sevp,
                 timer_t *timerid);
```

Аргумент `clockid` нужен для выбора системных часов, с которыми будет работать таймер

Аргумент `sevp` нужен, чтобы назначить действие по истечению таймера. Через данный аргумент в функцию передается структура `sigevent`.

Поля структуры:

1) В поле `sigev_notify` целого типа выбирается тип реакции на истечение таймера. Возможные значения:

- SIGEV_NONE — не делать ничего.
- SIGEV_SIGNAL — послать сигнал.
- SIGEV_THREAD — запускает в новом потоке выполнения функцию на которое указывает поле sigev_notify_function.

2) В поле `sigev_signo` целого типа указывается номер посылаемого сигнала.

3) Поле `sigev_value`, которое является объединением `sigval` вида:

```
union sigval {
   int     sival_int;
   void   *sival_ptr;
};
```

В данном поле указываются значения которые будут содержаться в структуре siginfo при обработке сигнала от данного таймера.

Аргумент `timerid` используется для возвращения идентификатора таймера.

Данная функция возвращает 0 в случае успуха, -1 в случае ошибки.

Функция timer_settime():

```
int timer_settime(timer_t timerid, int flags,
                  const struct itimerspec *new_value,
                  struct itimerspec *old_value);
```

Аргумент `timerid` передает функции идентификатор таймера.

Аргумент является указателем на структуру `itimerspec`, в которой передаются новые значения периода работы

Если аргумент `flags` равен TIMER_ABSOLUTE, то поле value в структуре `itimerspec`, инрепретируется как абсолютное время выбранных для таймера системных часов.

Функция timer_delete() удаляет интервальный таймер:

```
int timer_delete(timer_t timerid);
```

Аргумент `timerid` передает функции идентификатор таймера.

2.4. Генератор лексических анализаторов flex

Программа flex является предназначена для сканирования текстовых файлов и поиска в них определенных последовательностей символов. На входе программа получает текст с набором правил, а на выходе выдает сгенерированный код анализатора на языке Си.

Входной файл представляет собой текстовый файл вида:

Определения

%%

Правила

%%

Пользовательский код

В разделе определений объявляются идентификаторы, которые будут использоваться внутри анализатора.

В разделе правила описывается поведение работы программы с помощью правил, которые состоят из двух частей: регулярное выражение и код на языке Си который будет выполняться при обнаружении строки соответствующей данному выражению во стандартном потоке ввода.

В разделе пользовательский код пишется код, который будет исполняться по окончанию чтения файла.

3. АЛГОРИТМИЧЕСКАЯ ЧАСТЬ.

Особенностью разрабатываемой виртуальной отладочной платы является то, что моделировании она будет состоять из двух параллельно запущенных процессов:

Основной процесс qtvitrboard используется для взаимодействия с пользователем и управления процессом моделирования.

Процесс моделирования используемый для запуска моделирования и синхронизации реального и модельного времени. Исполняемый файл данного процесса создается программой qtvirtboard. Взаимодействие с данной программой будет осуществляться через модуль-тестер, описанный в отдельной библиотеки.

3.1. Разработка алгоритмов.

3.1.1. Укрупнена блок-схема процесса моделирования.

При моделировании будет использоваться модуль-тестер со следующим алгоритмом работы:

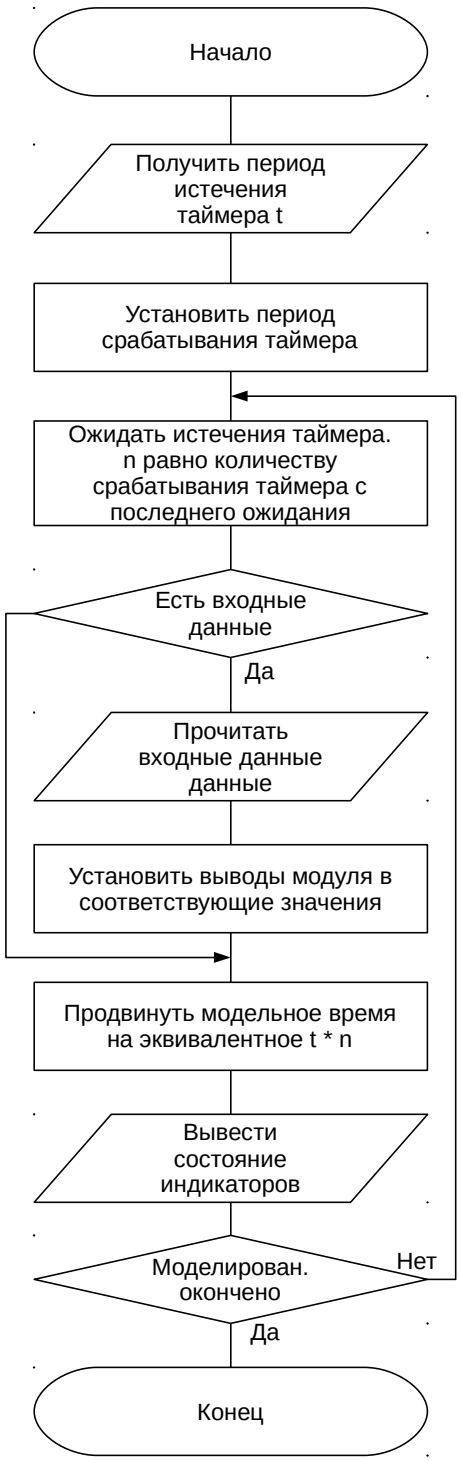

Рис. 2. Укрупненная блок схема моделирования

3.1.2. Взаимодействие с процессом моделирования.

Для данные передающиеся процессу моделирования vbproc и из представляют собой текстовые команды. Таких команд два типа:

1. команды предназначенные для изменения входных данных на портах моделируемого устройства.
2. команды предназначенные для изменения состояния моделирования.

Команды первого типа будем называть командами ввода данных, второго типа командами управления.

Команды ввода данных будут иметь следующий синтаксис:

```
-имя_порта:число
```

Команды управления будут иметь похожий синтаксис синтаксис:

```
#имя_команды:число
```

Данные которые возвращает vbproc данные так же в виде команд:

Команды вывода данных(синтаксис аналогичен командам ввода)

Команды вывода информации о моделировании(синтаксис аналогичен командам управления).

Команды ввода данных:

Имена входных портов для кнопок:

btn1, btn2, btn3, btn4, btn5

Значения: 0 — кнопка зажата, не равно 0 — кнопка отжата.

Имена команд управления:

- exit — завершает процесс vb_proc.

  Для команды exit через число передается код завершения процесса.
- start — возобновить моделирование.
- stop — остановить моделирование.

  Для данных команд числовое значение не используется.

Команды вывода данных:

Имена выходных портов для светодиодных индикаторов:

led1, led2, led3, led4, led5.

Значения числа: 0 — не горит, не равно 0 — горит.

Имена выходных портов для семисегментных индикаторов.

sevseg1, sevseg2.

В данном случае в виде числа передается двоичная маска, где первый бит управляет сегментом A, второй бит управляет сегментом B, и так далее.

Команды вывода информации о моделировании:

На данный момент единственной командой такого типа является команда time.

Значение числа: время моделирования в миллисекундах.

3.1.3. Ввод и вывод команд

Алгоритм ввода и вывода, при взаимодействии процессов между собой, у обоих схожий.

Для процесса моделирования ввод-вывод будет происходить следующим образом.

Ввод в программе выходных команд состоит из двух этапов:

1. Чтение списков команд ввода данных и команд управления из входящего потока;
2. Обработка команд из этих списков команд;

Вывод данных команд в стандартный поток вывода:

1. Формирования списка команд вывода данных и команд информации о моделировании.
2. Отправка списка команд в выходящий поток основному процессу.

Рассмотрим по подробнее ввод данных.

3.1.4. Чтение команд.

Прочитанные команды заносятся в список либо в список команд ввода либо в список команд управления. Блок-схема показана на рис. 3:

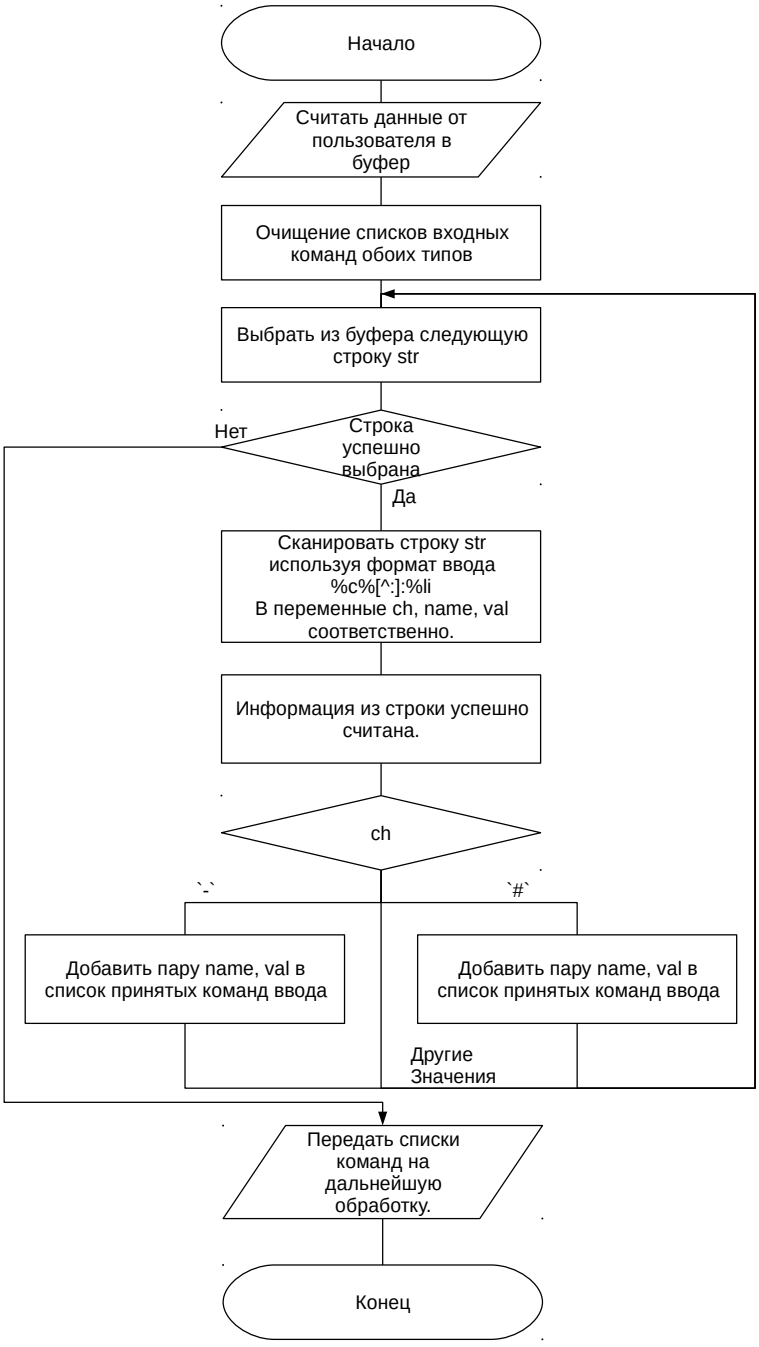

Рис. 3. Блок-схема алгоритма чтения команд.

3.1.5. Обработка поступающих команд.

Алгоритм обработки команд показан на рис 4.

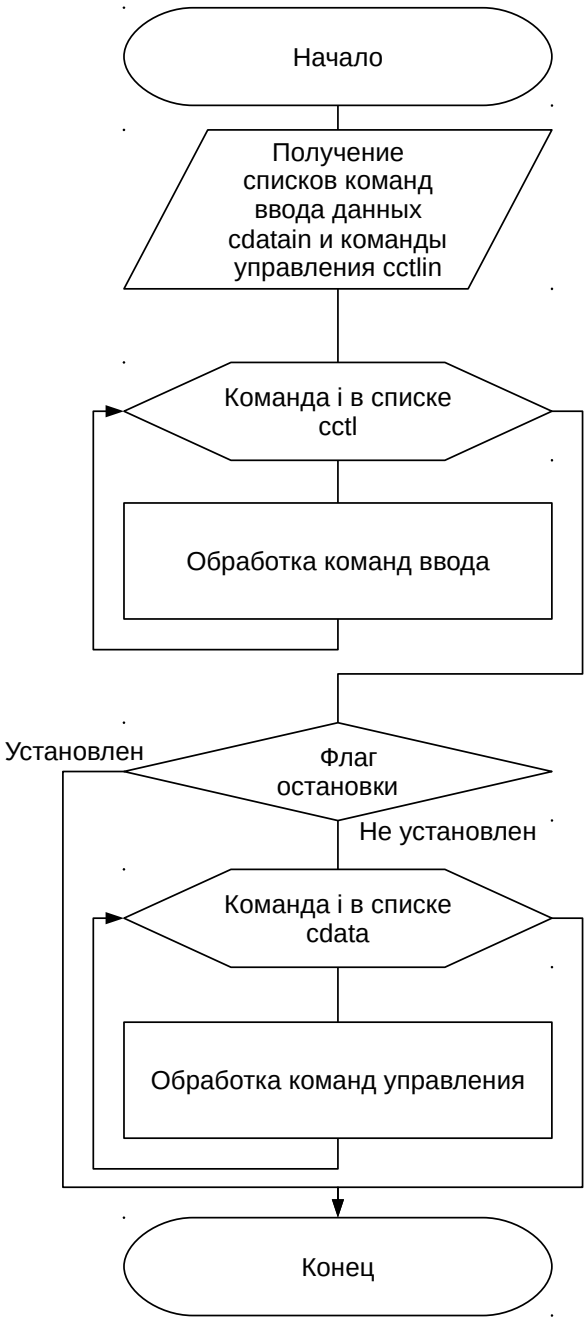

Рис. 4. Блок-схема алгоритма чтения команд.

3.1.6. Обработка команд ввода данных

Алгоритм обработки команд ввода данных.

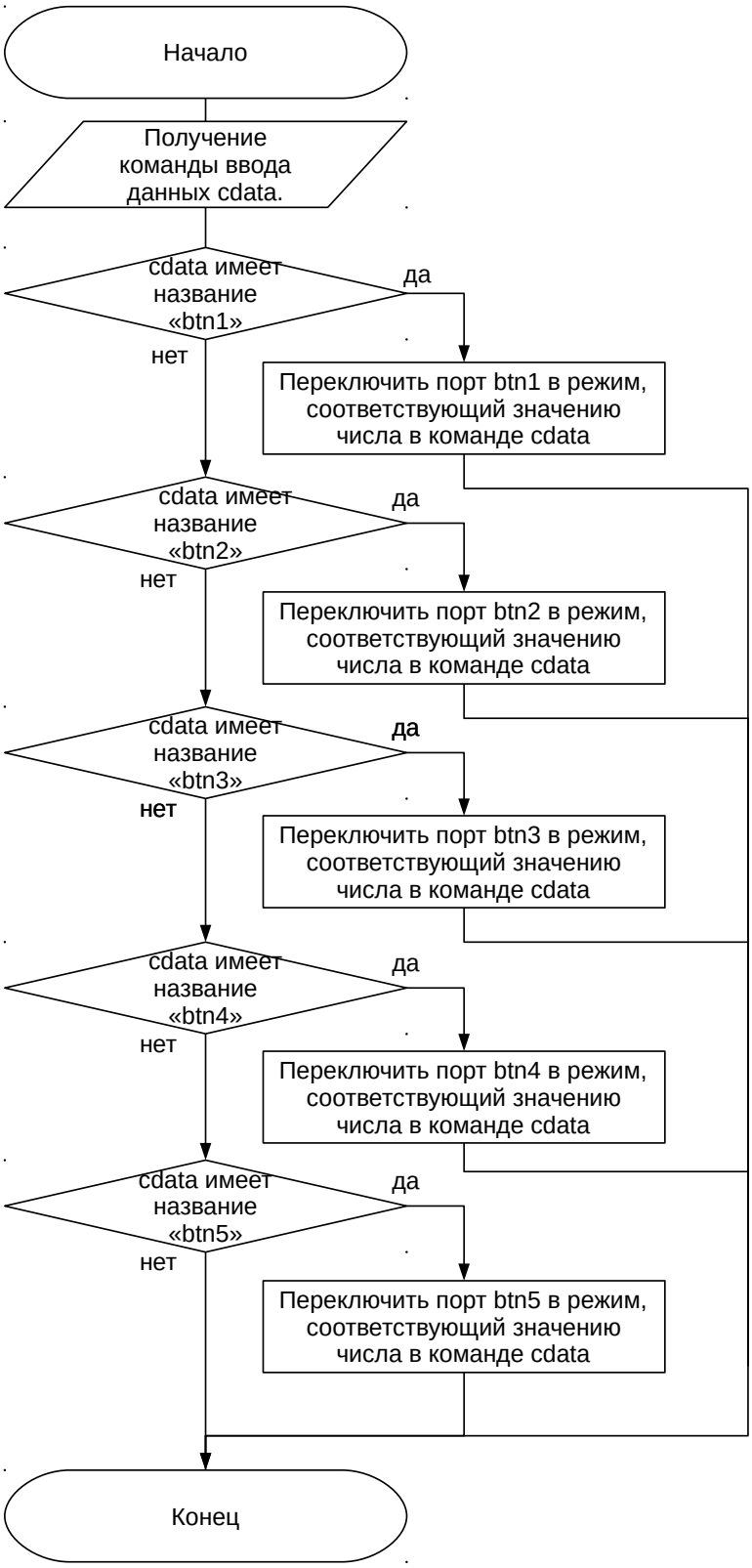

Рис. 5. Блок-схема алгоритма обработки команд ввода данных

3.1.7. Обработка команд управления.

Алгоритм обработки команд управления.

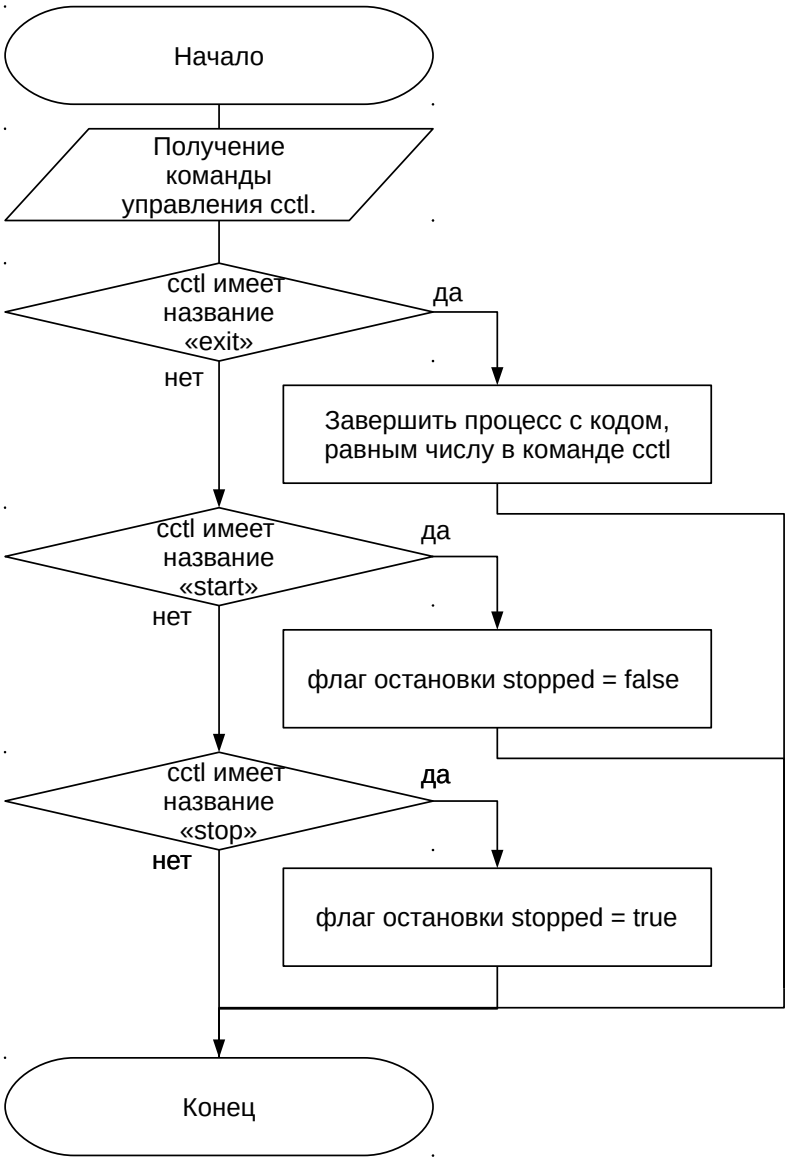

Рис. 6. Блок-схема алгоритма обработки команд ввода данных.

3.1.8. Вывод выходных команд.

Алгоритм вывода выходных команд.

Рис. 7. Блок-схема алгоритма обработки команд ввода данных.

3.1.9. Формирование списка команд вывода данных.

Алгоритм формирования списка команд вывода.

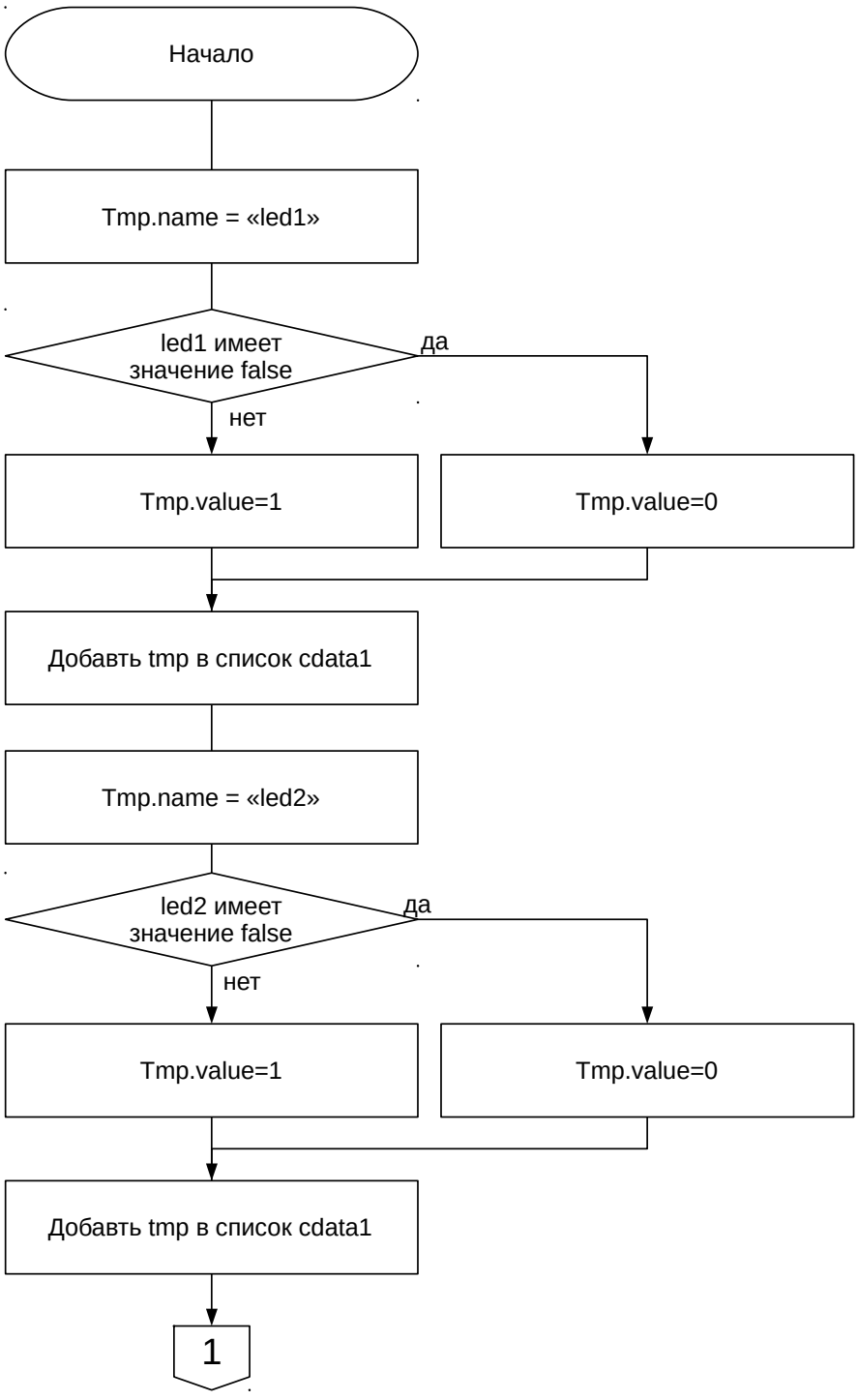

Рис. 8. Блок-схема алгоритма формирования списка команд вывода (часть. 1).

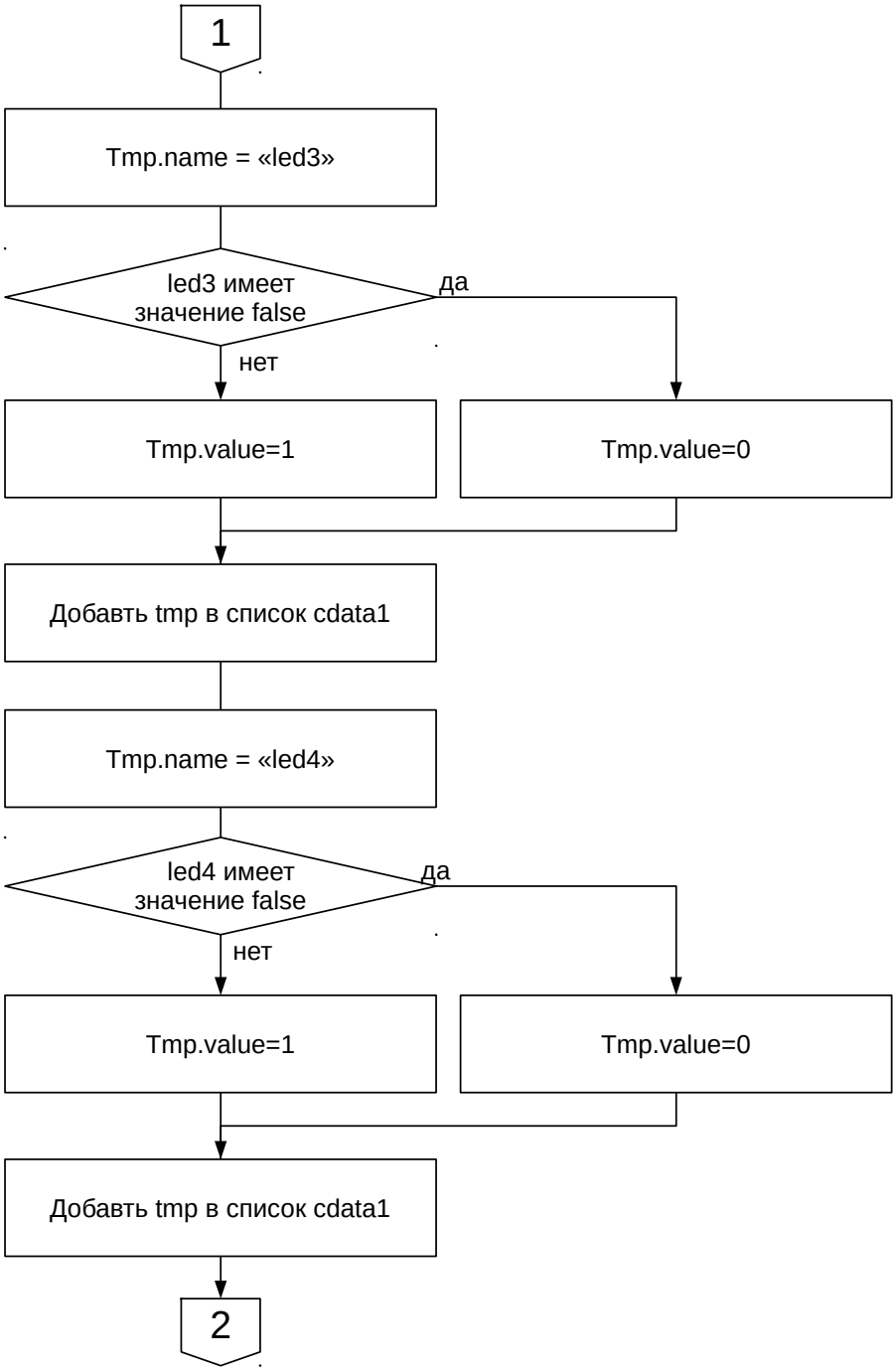

Рис. 9. Блок-схема алгоритма формирования списка команд вывода (часть. 2).

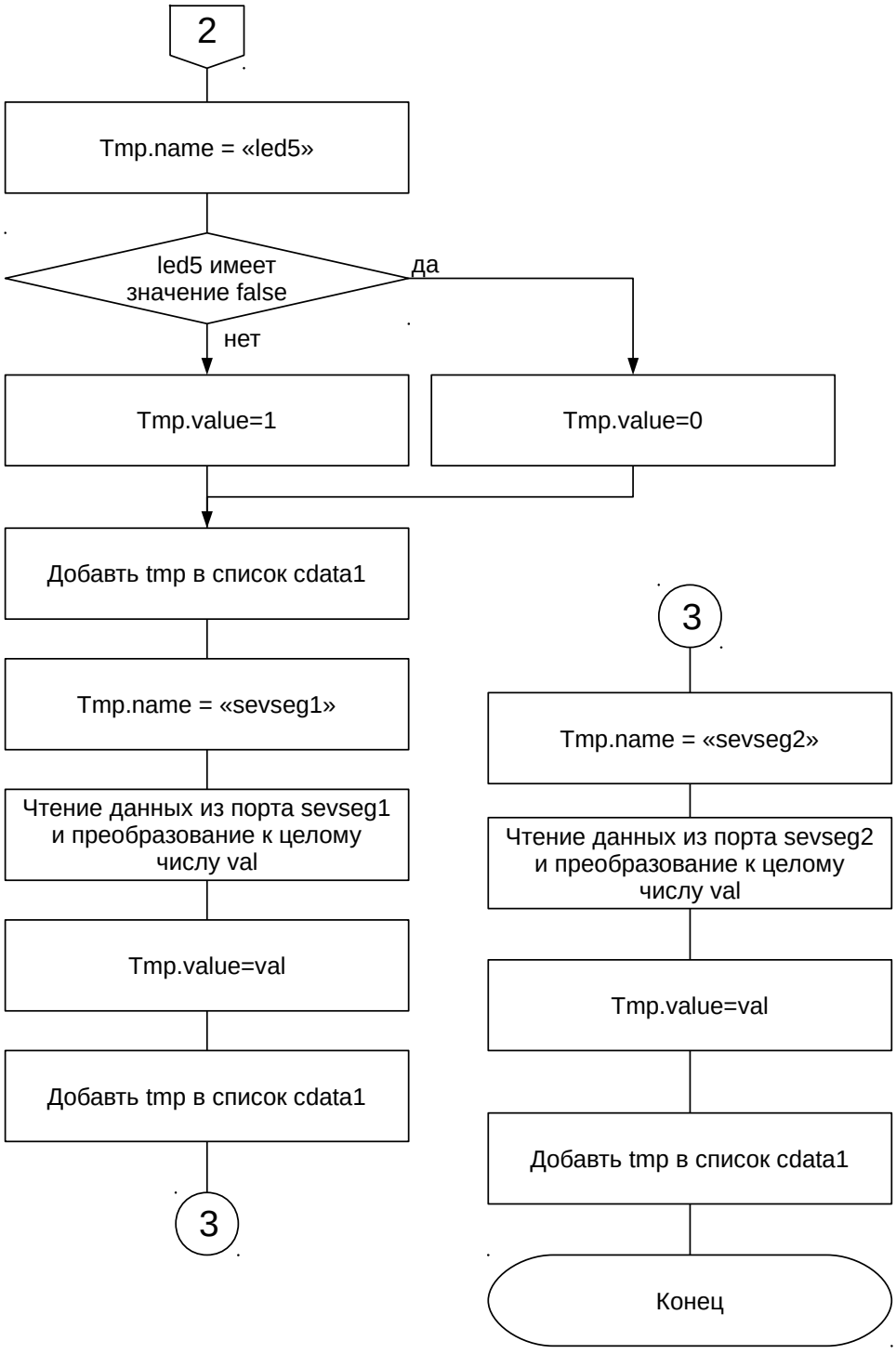

Рис. 10. Блок-схема алгоритма формирования списка команд вывода (часть. 3).

3.1.10. Формирование списка команд вывода информации о моделировании.

Алгоритм формирования списка команд вывода информации о моделировании.

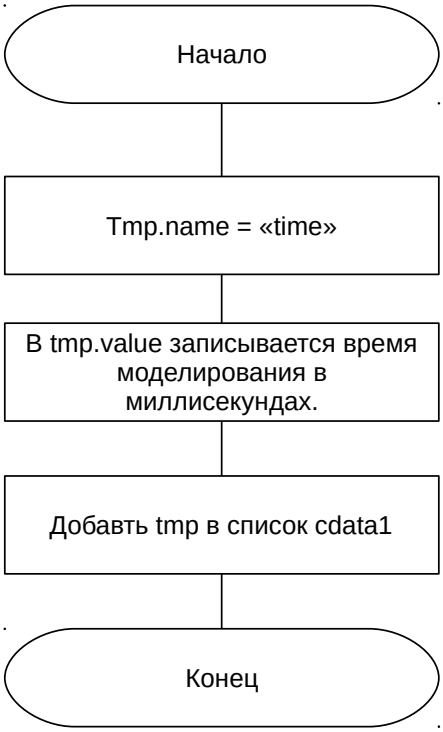

Рис. 11. Блок-схема алгоритма формирования списка команд вывода информации о моделировании.

3.1.11. Класс таймеров vb_timer.

Класс таймеров vb_timer в упрощенном виде имеет вид:

Класс таймеров
Свойства
(-)_timer (POSIX таймер) (-)timerid типа int(идентификатор таймера) (-)is_called типа bool (флаг срабатывания) (-)counter типа unsigned (количество истечений таймера)
Методы
(+)Дружественная функция vb_handler_timer() (+)void set(int millisec) (+)unsigned wait() (+)void stop() (+)vb_timer()(Конструктор) (+)~vb_timer()(Деструктор)

Метод set(int millisec) устанавливает время истечения таймера POSIX в миллисекундах.

Метод unsigned wait() ожидает истечения таймера POSIX. Возвращает количество раз, когда истекал таймер, с прошлого вызова wait().

Метод stop() останавливает таймер POSIX.

Дружественная функция vb_handler_timer() используется для обработки сигналов посылаемых таймеров POSIX.

3.1.12. Создание экземпляра класса таймеров vb_timer.

Алгоритм создания экземпляра класса таймера рис. 6

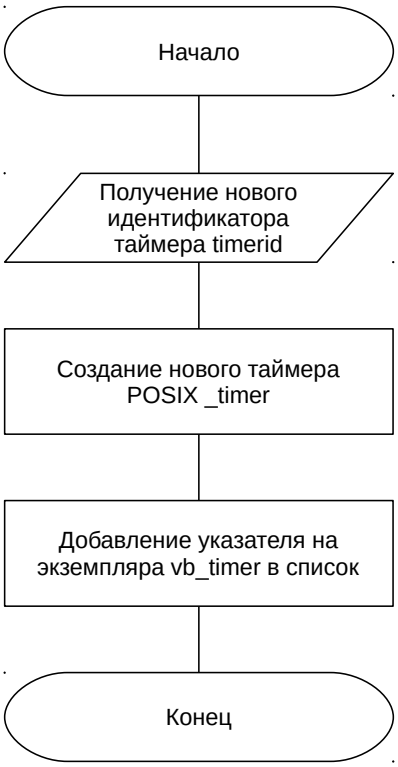

Рис. 12. Блок-схема алгоритма создания экземпляра класса таймеров vb_timer.

При создании POSIX таймера указывается что по истечению таймера он посылает сигнал с использованием структуры siginfo. В данной структуре указывается идентификатор timerid созданного класса.

3.1.13. Обработка сигналов таймера POSIX.

Алгоритм обработки сигнала от таймера POSIX показан на Рис. 7:

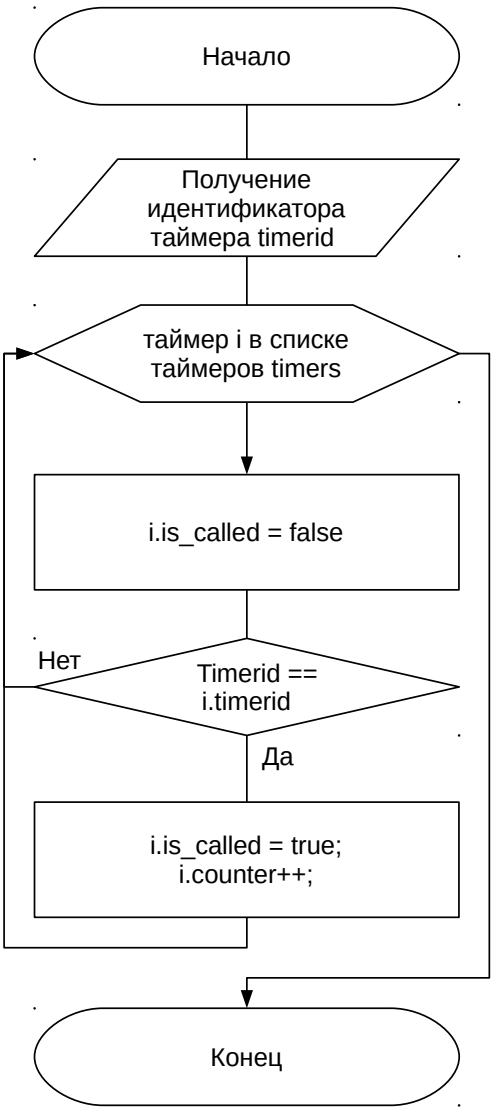

Рис. 13. Обработка сигналов таймера POSIX.

3.1.14. Компиляция исполняемого файла процесса моделирования.

Для запуска процесса моделирования необходимо предварительно скомпилировать исполняемый файл данного процесса.

Компиляция исполняемого файла процесса моделирования состоит из следующих этапов:

1. Компиляция описания модуля SystemC.
2. Лексический анализ описания модуля для выделения в нем наименования и портов.
3. Генерация файла на языке C++ с функцией sc_main() на основе данных лексического анализа.
4. Компиляция файла, полученного на предыдущем этапе.
5. Компоновка полученных откомпилированных файлов с библиотекой модуля-тестера и библиотекой SystemC в исполняемый файл.

3.1.15. Лексический анализ.

Назначение лексического анализатора в данной программе - это поиск в описании модуля: наименования модуля, входных и выходных портов. Это необходимо для интеграции описания устройства с основным ядром моделирования. Данные анализатора потом передаются для генерации sc_main файла в котором и осуществляется интеграция

Для лексического анализа применяется следующий алгоритм:

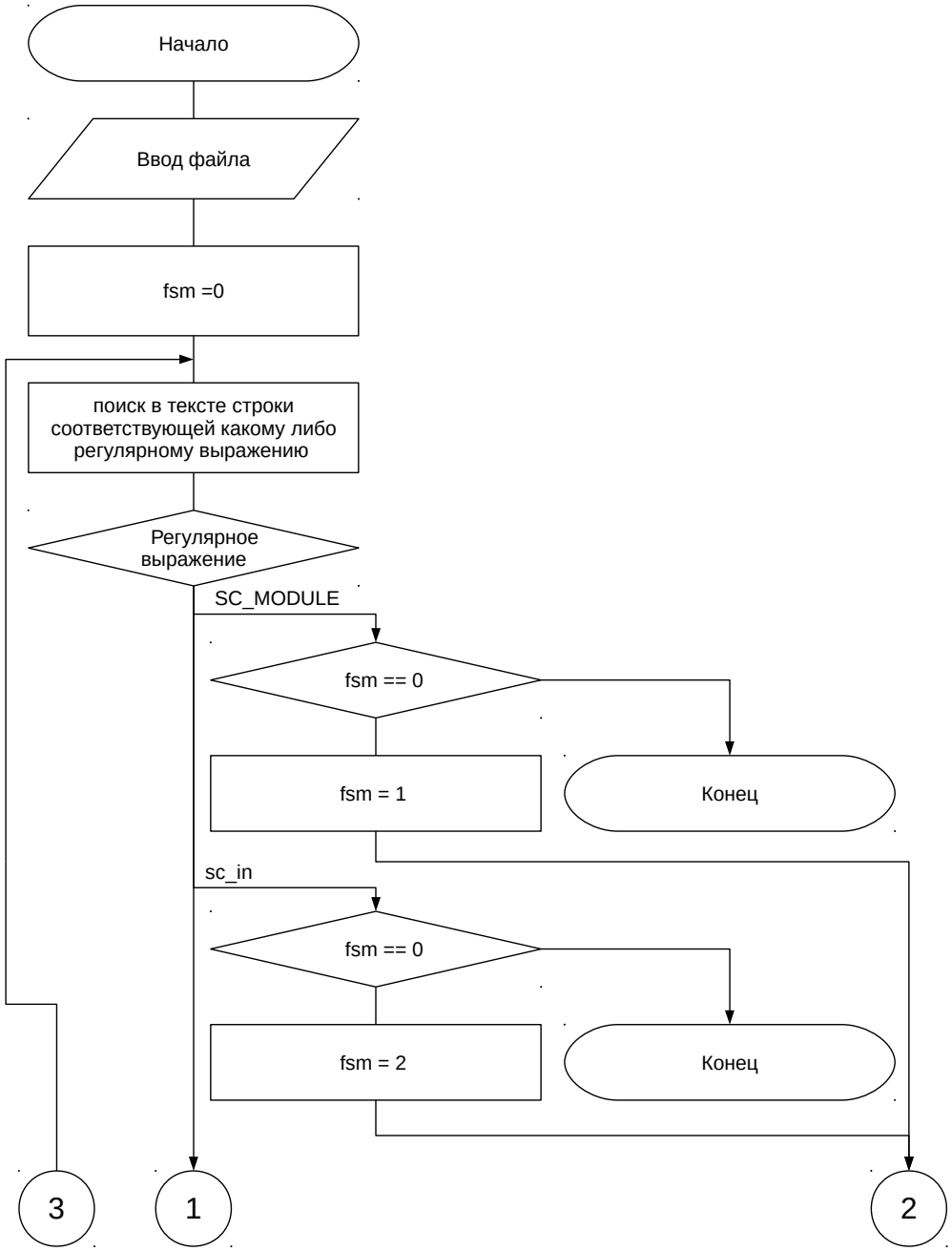

Рис. 14. Алгоритм лексического анализа (часть. 1).

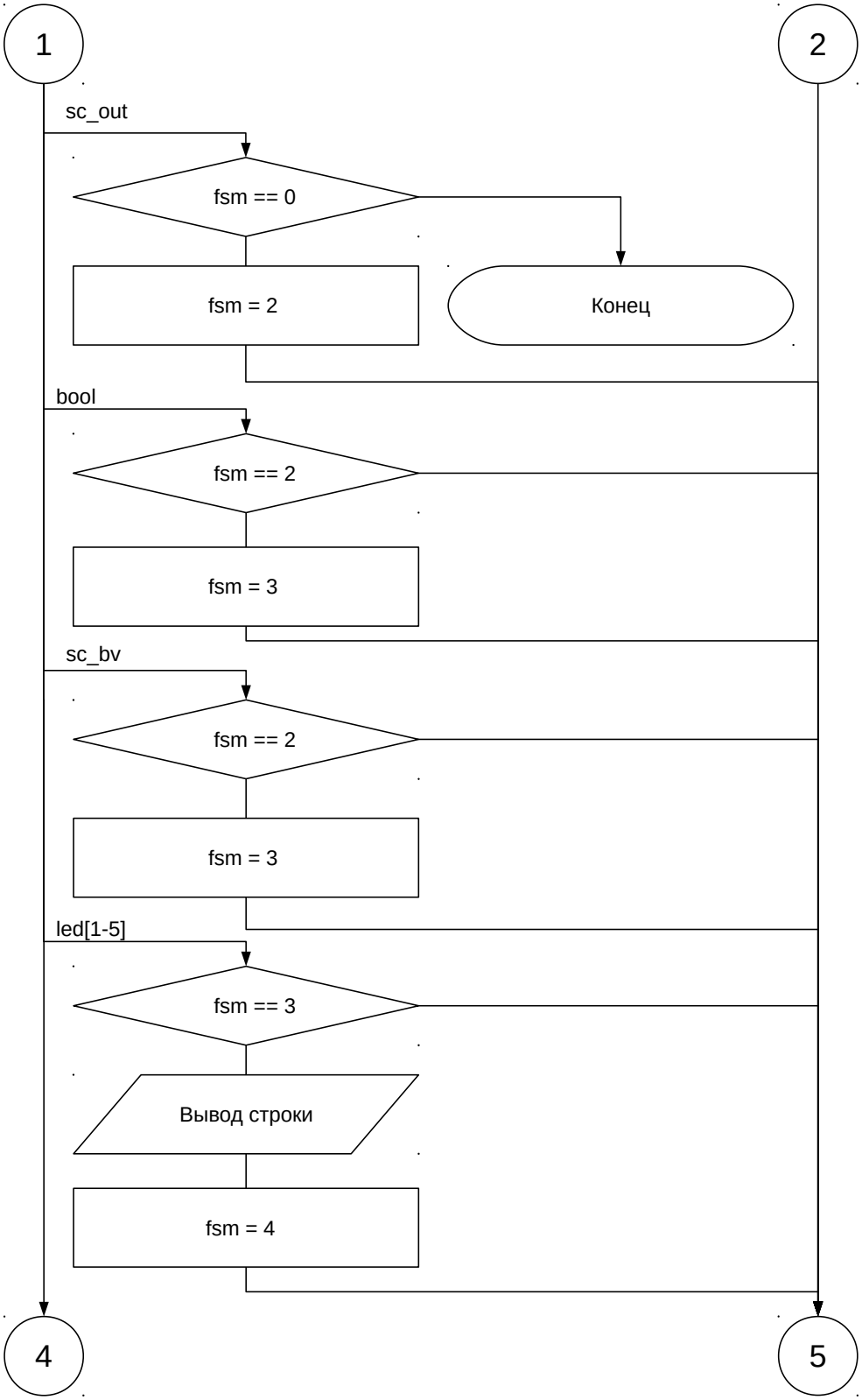

Рис. 15. Алгоритм лексического анализа (часть. 2).

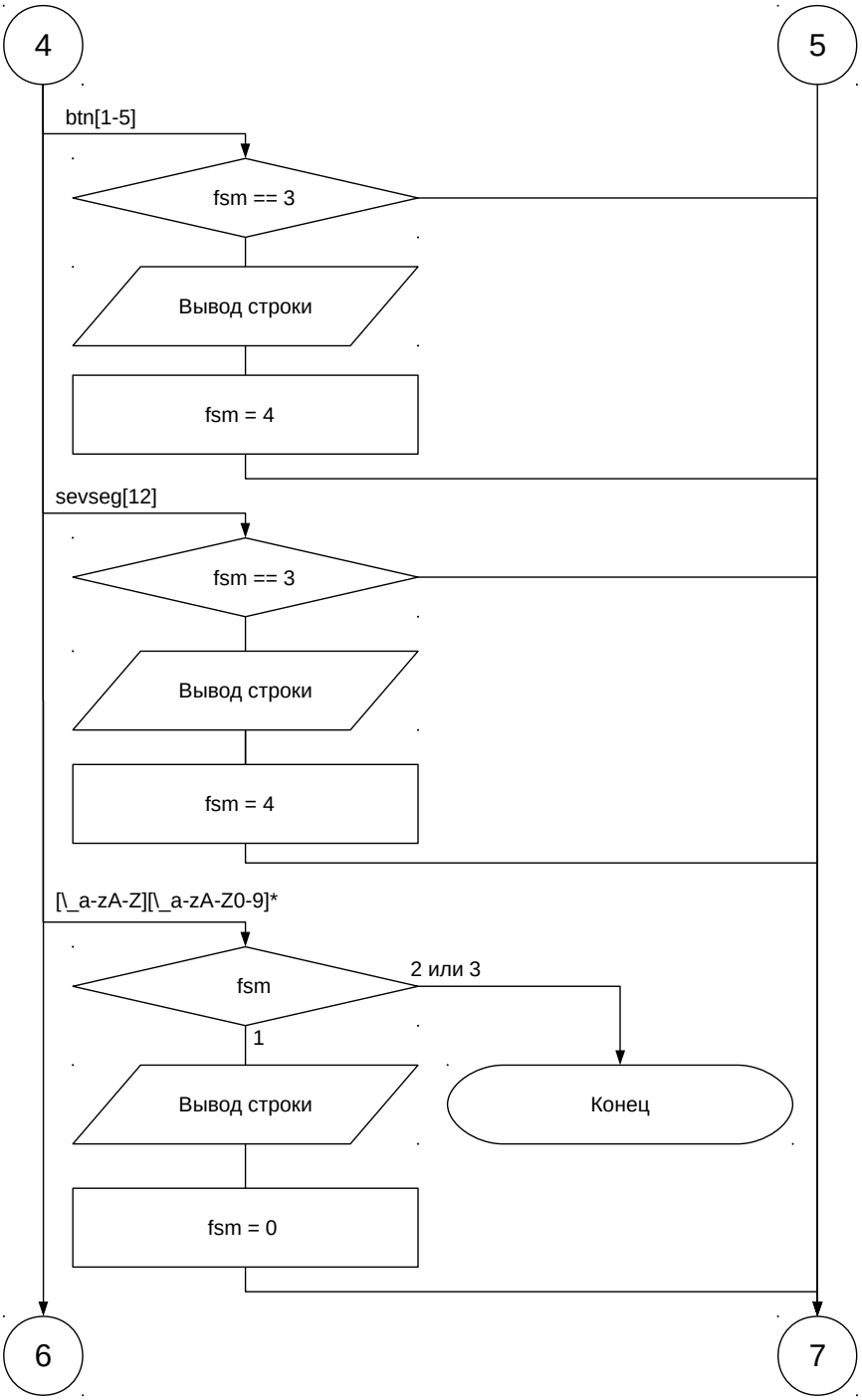

Рис. 16. Алгоритм лексического анализа (часть. 3).

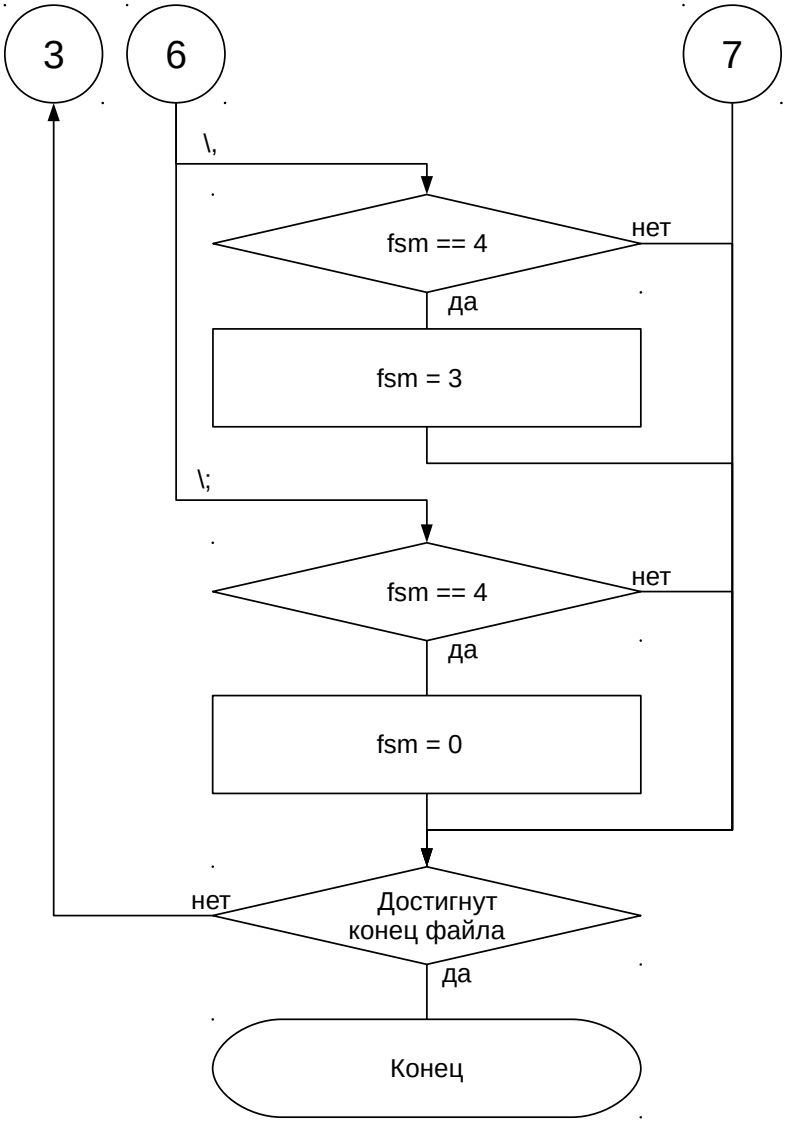

Рис. 17. Алгоритм лексического анализа (часть. 4).

3.1.16. Генерация файла содержащего функцию sc_main().

Такие действия как: объединение модулей верхнего уровня иерархии или запуск процесса моделирования в библиотеке SystemC осуществляется в функции sc_main(). Данная функция в библиотеке SystemC аналогична функции main языка C++. Генерация файла происходит на основе шаблона и состоит из следующих пунктов:

Получение от лексического анализатора наименования и используемых портов в тестируемом модуле.

Вставка в шаблон строки:

```
#include «имя_заголовочного_файла»
```

Вставка в шаблон строки внутри функции sc_main:

```
имя_модуля module(«MODULE»);
```

Вставка в шаблон для каждого найденного порта внутри функции sc_main() строки следующего содержания

```
module.имя_модуля(имя_модуля);
```

3.2. Разработка программ.

Программа состоит из следующих составных частей:

1. Основная программа qtvirtboard.
2. Библиотека ввода-вывода vb_io.
3. Библиотека модуля-тестера vb_testbench.
4. Программа лексического анализа vbllex, созданная с использованием утилиты flex
5. Скрипт компиляции и установки программы в системе.

3.2.1 Разработка библиотеки vb_io.

Библиотека vb_io используется для взаимодействия основной программы и модели, представляющей собой так же программу.

В данной библиотеки описаны типы:

- структура cmd.

Используется для хранения команд в программах.

- Тип vec_cmds представляющий собой контейнер vector для типа cmd.

Используется в при обработке данных.

В данной библиотеки описаны следующие функции:

- bool read_cmds(vec_cmds * cctl, vec_cmds * cdata);

Осуществляет чтение команд из стандартного потока ввода вывода и заносит их в списки на которые указывают аргументы cctl и cdata.

Возвращает false если новых данных не поступило true в ином случае.

- void write_cmds(vec_ctl_data * cmds, vec_ctl_data * vcd);

Осуществляет запись команд из списков на которые указывают аргументы cctl и cdata в стандартный поток ввода вывода.

- void vec_cmds2str(char * str, vec_cmds * cctl, vec_cmds * cdata);

Записывает команды из списков на которые указывают аргументы cctl и cdata в буфер srt,. Используется внутри функции write_cmds().

- void str2vec_cmds(vec_cmds * cctl, vec_cmds * cdata, char * str);

Заносит команды из буфер str в списки на которые указывают аргументы cmd и vcd. Используется внутри функции read_cmds().

3.2.2. Разработка библиотеки vb_testbench.

Данная библиотека нужна для управления моделируемым устройством и синхронизации модельного и реального времени.

В данную библиотеку входит класс vb_timer. Данный класс наследуется от структуры sc_module, являющаяся частью библиотеки SystemC.

Данный класс имеет следующие поля:

Частные поля:

Контейнер cdatain имеющий тип vec_cmds. Используется для хранения команд ввода данных.

Контейнер cctlin имеющий тип vec_cmds. Используется для хранения команд управления моделированием.

Контейнер cdataout имеющий тип vec_cmds. Используется для хранения команд вывода данных.

Контейнер cctlout имеющий тип vec_cmds. Используется для хранения команд

Переменная _model_step имеющая тип int. Используется для хранения минимального шага, с которым происходит взаимодействие с загруженным модулем.

Переменная stopped типа bool. Флаг остановки моделирования.

sync_timer который является экземпляром класса vb_timer.

Публичные поля:

Порты btn1, btn2, btn3, btn4 btn5 имеющие класс sc_out<bool>. Используются для подключения загруженного модуля к тестеру. Через данные порты в происходит воздействие пользователя на модуль.

Порты led1, led2, led3, led4, led5 имеющие класс sc_in<bool>. Используется для подключения загруженного модуля к тестеру. Эти порты соответствуют светодиодам на виртуальной отладочной плате.

sevseg1, sevseg2 имеющие класс sc_in<sc_bv<7>>. Используется для подключения загруженного модуля к тестеру. Эти порты соответствуют семисегментному индикаторам виртуальной отладочной платы.

Публичные методы данного класса:

```
virtual void handlcdata(const cmd & cdata);
```

Данный метод обрабатывает команды ввода данных из cdata.

virtual void handlcctl(const cmd & cctl);

Данный метод обрабатывает поступающие команды управления из cctl.

virtual void func_outcdata(vec_cmds * cdata);

Данный метод заносит команды выходных данных в cdata

```
virtual void func_outcctl(vec_cmds * cctl);
```

Данный метод заносит команды вывода информации о моделировании.

```
void setStopped(bool);
```

Метод установить флаг остановки.

void test_thread();

Метод который описывает являющийся процессом SC_THREAD. Описывает работу модуля-тестера.

vb_testbench(sc_module_name name,int millisec);

Конструктор класса vb_testbench. Через millisec указывается шаг моделирования в миллисекундах.

3.2.3. Разработка программы qtvirtboard.

Для основой программы разработаны следующие классы:

1. Класс основного окна — mainwindow;
2. Класс окна «Настройка» — settings_wind;
3. Класс окна «Открыть модуль» — windopenauto;
4. Класс светодиодных индикаторов — vbled;
5. Класс семисегментных индикаторов — sevsegitm;
6. Класс кнопок виртуальной отладочной платы vbbtn;

Класс mainwindow наследуется от класса окна QmainWindow.

Частные поля:

- Контейнер cDataIn имеющий тип vec_cmds. Используется для хранения команд ввода данных, которые посылаются программой qtvirt_board.

- Контейнер cCtlIn имеющий тип vec_cmds. Используется для хранения команд управления моделированием, которые посылаются программой qtvirt_board..
- Контейнер cDataOut имеющий тип vec_cmds. Используется для хранения команд вывода данных, которые принимаются программой qtvirt_board..
- Контейнер cCtlOut имеющий тип vec_cmds. Используется для хранения команд, которые принимаются программой qtvirt_board.
- Объект InPipe имеющий класс QByteArray. Используется при передачи данных для процесса моделирования в качестве буфера.
- Объект OutPipe имеющий класс QByteArray. Используется при получении данных от процесса моделирования в качестве буфера.
- Объект vbproc имеющий класс Qprocess. Используется для взаимодействия с процессом моделирования программы qtvirt_board.
- Объект namevb имеющий класс QString. Хранит имя исполняемого файла для процесса моделирования.
- Указатель на объект _windowOpenAuto класса WindOpenAuto. Используется для создания окна «Открыть модуль»
- Указатель на объект _settingsWind класса setting_wind. Используется для создания окна «Настройка».
- Указатели на объект led1, led2, led3, led4, led5 имеющие классы vbled. Используется для отображения светодиодных индикаторов на виртуальной отладочной платы.
- Указатели на объект btn1, btn2, btn3, btn4, btn5 имеющие класс vbbtn. Используется для отображения кнопок на виртуальной отладочной платы.
- Указатели на объект sevseg1, sevseg2 имеющие класс SevSegItm. Используются для отображения семисегментных индикаторов.

Частные слоты:

- Слот void updateSettings(). Данный слот обновляет настройки программы
- Слот void OpenModuleAuto(). Данный слот открывает окно «Открыть модуль».
- Слот void OpenSettings(). Данный слот открывает окно «Настройки».
- Слот void recieveData(). Предназначен для чтения команд от процесса моделирования.
- Слот void handlInCmd(). Предназначен для отправки команд управления процессу моделирования.
- Слот void handlInData(). Предназначен для отправки команд ввода данных процессу моделирования.
- Слот void handlOutCmd(). Предназначен для обработки команд вывода данных от процесса моделирования.
- Слот void handlOutData(). Предназначен для обработки команд вывода информации о моделировании.
- Слот void startModel(QString nm). Предназначен для запуска процесса моделирования. Через аргумент nm передается имя исполняемого файла программы.
- Слоты void btn1_clicked(); void btn2_clicked(); void btn3_clicked(); void btn4_clicked(); void btn5_clicked(). Данные слоты обрабатывают нажатия кнопок виртуальной отладочной платы.
- Слот on_start_btn_clicked(). Данный слот реагирует на нажатие кнопки «старт» и отправляет команду возобновления моделирования.
- Слот on_stop_btn_clicked(). Данный слот реагирует на нажатие кнопки «стоп» и отправляет команду остановки моделирования моделирования.

Класс setting_wind наследуется от класса диалоговых окон Qdialog.

Частные поля:

- Объект configFile имеет класс Qfile. С помощью данного класса происходит взаимодейсвия.

- Объект SystemCPath класса QString. Хранит имя каталога, в котором находится библиотека SystemC.
- Объект LexPath класса Qstring. Хранит имя каталога, в котором находится лексический анализатор.
- Объект GccPath класса Qstring. Хранит имя каталога, в котором находится компилятору
- Объект VirtBoardPath класс Qstring. Хранит имя каталога, в котором находится библиотека модуля тестера vb_testbench.
- Объект inclVirtBoard класса Qstring. Хранит имя каталога, в котором находятся заголовочные файлы тестера vb_testbench.
- Объект inclSystemC класса Qstring. Хранит имя каталога, в котором находятся заголовочные файлы библиотеки SystemC.

Публичные методы:

- Метод QString getSystemCPath(). Возвращает путь к библиотеке SystemC.
- Метод Qstring getGccPath(). Возвращает указанный в настройках путь к компилятору GCC.
- Метод Qstring getLexPath(). Возвращает указанный в настройках путь к лексическому анализатору vbllex.
- Метод Qstring getVirtBoardPath(). Возвращает указанный в настройках путь путь к библиотеке vb_testbench.
- Метод Qstring getIncPathVB(). Возвращает указанный в настройках путь к заголовочным файлам vb_testbench.
- Метод Qstring getIncPathSC(). Возвращает указанный в настройках путь к заголовочным файлам SystemC.

Частные слоты:

- Слот void on_OkBtn_clicked(). Обрабатывает нажатие на клавишу «Ок»
- Слот void on_cancelBtn_clicked(). Обрабатывает нажатие на клавишу «Отмена»

- Слот void readConf(). Читает конфигурационный файл .config.
- Слот void writeConf(). Записывает настройки в конфигурационный файл.

Сигналы Qt:

- Сигнал void settingChange(). Сигнал об изменении настроек.

Класс windopenauto. Наследуется от класса окна QmainWindow.

Частные поля:

- Объект nameGcc класса QString. Хранит имя компилятора GCC.
- Объект nameVBLex класса Qstring. Хранит имя лексического анализатора vbllex.
- Объект nameNoduleCpp класса Qstring. Хранит имя cpp файла модуля.
- Объект nameModuleH класса Qstring. Хранит имя h файла модуля.
- Объект nameModel класса QString. Хранит имя исполняемого файла процесса моделирования
- Объект nameObjMod класса QString. Хранит имя объектного файла модуля
- Объект nameModule класса Qstring. Хранит Название идентификатора модуля для генерации файла содержащего sc_main функцию.
- Объект nameSc_main класса QString. Хранит название файла содержащего функцию sc_main().
- Объект nameObjMain класса Qstring. Хранит откомпилированный объектный файл содержащий функцию sc_main().
- Объект SystemCPath класса QString. Хранит имя каталога, в котором находится библиотека SystemC, получаемое из окна «Настройки».
- Объект LexPath класса Qstring. Хранит имя каталога, в котором находится лексический анализатор, получаемое из окна «Настройки».
- Объект GccPath класса Qstring. Хранит имя каталога, в котором находится компилятор, получаемого из окна «Настройки».

- Объект VirtBoardPath класс Qstring. Хранит имя каталога, в котором находится библиотека модуля тестера vb_testbench, получаемого из окна «Настройки».
- Объект inclVirtBoard класса Qstring. Хранит имя каталога, в котором находятся заголовочные файлы тестера vb_testbench, получаемого из окна «Настройки».
- Объект inclSystemC класса Qstring. Хранит имя каталога, в котором находятся заголовочные файлы библиотеки SystemC, получаемого из окна «Настройки».

Частные методы:

- Метод QString getTempNameF(QString name, Qstring ext). Предназначен для генерации имени временного файла. Аргумент name — имя, аргумент ext — расширение файла.
- Метод Qstring getPathFile(QString dirname, Qstring file). Предназначен для добавляет формирования полного имени имени.
- Публичные методы:
- Метод void setSystemCPath(QString Path). Записывает данные в поле SystemCPath
- Метод QByteArray err_out(); Считывает данные об ошибке при работе компилятора.
- Метод void setLexPath(QString Path). Записывает данные в поле LexPath
- Метод void setGccPath(QString Path). Записывает данные в поле GccPath
- Метод void setVirtBoardPath(QString Path). Записывает данные в поле VirtBoardPath
- Метод void setInclPathVB(QString Path). Записывает данные в поле inclVirtBoard

- Метод void setInclPathSC(QString Path). Записывает данные в поле inclSystemC
- Метод void updatePathName(). Обновляет хранящиеся имена файлов имена файлов.

Класс vbled. Наследуется от класса QLabel

Частные поля:

- Объект imgon класса QPixmap Хранит изображение горящего индикатора.
- Объект imgoff класса QPixmap. Хранит изображение негорящего индикатора.
- Переменная state типа bool. Хранит текущее значение светодиода.

Публичные методы:

- Метод setState(bool st). устанавливает текущее состояние светодиода.

Класс vbbtn. Наследуется от класса абстрактной кнопки QabstractButton.

Частные поля и методы:

- Объект imgon класса QPixmap Хранит изображение нажатой кнопки.
- Объект imgoff класса QPixmap. Хранит изображение ненажатой кнопки.
- Метод void paintEvent(QPaintEvent * e). Отвечает за отображение кнопки на экране.

Класс SevSegItm. Наследуется от класса сцены для отображения графической информации QGraphicsScene.

Частные поля:

- Переменная state типа int. Хранит текущее состояние семисегментного индикатора.
- Объект Bon класса QBrush. Хранит раскраску активного сегмента индикатора.
- Объект Boff класса QBrush. Хранит раскраску неактивного сегмента индикатора.

- Объекты A, B, C, D, E, F, G класса QgraphicsRectItem. Данные объекты отвечают за отображение сегментов индикатора.

Публичные методы:

- Метод setState(bool st). устанавливает текущее состояние светодиода.

4. РАЗРАБОТКА ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ.

4.1. Описание применения.

4.1.1. Назначение программы

Программа qtvirtboard предназначена для моделирования взаимодействия человека и отладочной платы на компьютере. Данная программа позволяет загрузить описание устройства и взаимодействовать с ним в реальном времени.

Возможности программы qtvirtboard:

1. программа способна моделировать реакцию отладочной платы на нажатие кнопок.
2. программа моделирует взаимодействие отладочной платы с пользователем по средством индикаторов.

Ограничение программы:

1. Данная программа предназначена в первую очередь для использования в операционных системах поддерживающих стандарт POSIX.
2. Описание устройства не должно содержать подмодулей.

4.1.2. Условия применения

Минимальные требования к аппаратным средствам:

1. Процессор архитектуры x86 с тактовой частотой не менее 2 ГГц.
2. Объем оперативной памяти не менее 1 Гб.
3. Объем свободной памяти на диске не менее 256МБ.

Требования к программным средствам:

1. Операционная система поддерживающая стандарт POSIX (Например, Ubuntu 18.04, Debian GNU/Linux 9).
2. Наличие пакета GNU Compile Collection (набор компиляторов проекта GNU).
3. Интерпретатор командной строки bash.
4. Наличие библиотеки SystemC версии не ниже 2.2.
5. Наличие программы flex версии не ниже 2.6.
6. Наличие библиотеки Qt для разработчиков версии не ниже 5.9.

7. Пакет утилит GNU Binary Utilities.
8. Пакет утилит GNU Core Utilities
9. Программа Make версии не ниже 4.1
10. Программа-архиватор tar версии не ниже 1.29

4.1.3. Описание задачи.

qtvirtboard должна моделировать поведение отладочной платы с загруженным на нее описанием устройства написанным на языке SystemC.

4.1.4. Входные и выходные данные.

Входными данными являются описание устройства написанное для библиотеки SystemC написанный в двух файлах:

1. Заголовочный файл с расширением h содержащий описание класса модуля.
2. Файл реализации с расширением cpp.

Выходными данными является отображаемое через графический интерфейс состояние индикаторов виртуальной отладочной платы.

4.2. Руководство системного программиста

4.2.1. Общие сведения о программе.

Назначением программы qtvirtboard является моделирование взаимодействия человека и отладочной платы на компьютере. Данная программа позволяет загрузить описание устройства и взаимодействовать с ним в реальном времени.

Минимальные требования к аппаратным средствам:

1. Процессор архитектуры x86 с тактовой частотой не менее 2 ГГц.
2. Объем оперативной памяти не менее 1 Гб.
3. Объем свободной памяти на диске не менее 256МБ.

Требования к программным средствам:

1. Операционная система поддерживающая стандарт POSIX (Например, Ubuntu 18.04, Debian GNU/Linux 9).
2. Наличие пакета GNU Compile Collection (набор компиляторов проекта GNU).
3. Интерпретатор командой строки bash.
4. Наличие библиотеки SystemC версии не ниже 2.2.
5. Наличие программы flex версии не ниже 2.6.
6. Наличие библиотеки Qt для разработчиков версии не ниже 5.9.
7. Пакет утилит GNU Binary Utilities.
8. Пакет утилит GNU Core Utilities.
9. Программа Make версии не ниже 4.1
10. Программа-архиватор tar версии не ниже 1.29

4.2.2. Структура программы.

Весть программный пакет состоит из следующих составных частей

1. Библиотека тестера vb_testbench. Обеспечивает управление моделируемым устройством, так же синхронизирует реальное и модельное время.
2. Библиотека ввода-вывода vb_io.

3. Программа qtvirtboard с графическим интерфейсом пользователя имеет следующие составные части:
   - Основное окно программы. Предназначено для визуализации нескольких составных частей программы использует библиотеку vb_io.
   - Окно «Открыть модуль». Предназначено для выбора файлов моделируемого описания устройства, запуска процесса компиляции модели устройства, передачи информации для модели устройства основному окну.
   - Окно «Настройки». Предназначено для указания параметров системы.
4. Программа лексического анализа текста vbllex. Используется на стадии компиляции модели.

4.2.3. Настройка и установка программы.

Установочный комплект распространяется в виде архива virtboard.tar.gz.

Состав установочного комплекта:

1. Исходный код библиотеки vb_testbench.
2. Исходный код библиотеки vb_io.
3. Исходный код программы qtvirtboard.
4. Исходный код программы vbllex.
5. Пример модуля SystemC testmod.

Установка программы:

Для установки программы нужно выполнить следующие действия:

1. Установить пакеты GNU Core Utilities, GNU Binary Utilities, bash, flex, gcc, Qt и.т
2. Загрузить архив с исходными текстами библиотеки SystemC с официального сайта(**https://www.accellera.org/downloads/standards/systemc**)
3. Запустить интерпретатор командной строки bash
4. Распаковать архив с текстами командой

```
tar -xzvf <имя_архива_библиотеки_SystemC>
```

5. Перейти в распакованый каталог.
6. Установить SystemC согласно официальной документации.
7. Распаковать архив с программой qtvirtboard командой
   ```
   tar -xzvf virtboard.tar.gz
   ```
8. Перейти в распакованный каталог `virtboard` командой:
   ```
   cd virtboard/
   ```
9. Запустить с скрипт установки командной:
   ```
   ./install.sh /путь/для/установки/
   ```

Настройка программы:

Для настройки программы используется файл `.config` настроек располагающийся в директории с исполняемым файлом. В файле указываются пути к необходимым для работы программам и библиотека.

Пример файла:

```
SystemCPath:
VirtBoardPath:/home/kost/virt_board/libvb/lib
inclVirtBoard:/home/kost/virt_board/libvb/include
inclSystemC:
LexPath:/home/kost/virt_board/libvb/
GccPath:
```

,Где

`SystemCPath` — путь к библиотеке SystemC.

`inclSystemC` — путь к заголовочным файлам библиотеки SystemC.

`VirtBoardPath` — путь к библиотекам vb_testbench и vb_io.

`inclVirtBoard` — путь к заголовочным файлам библиотек vb_testbench и vb_io.

`LexPath` — путь к программе лексического анализа текста.

`GccPath` — путь к компилятору GCC.

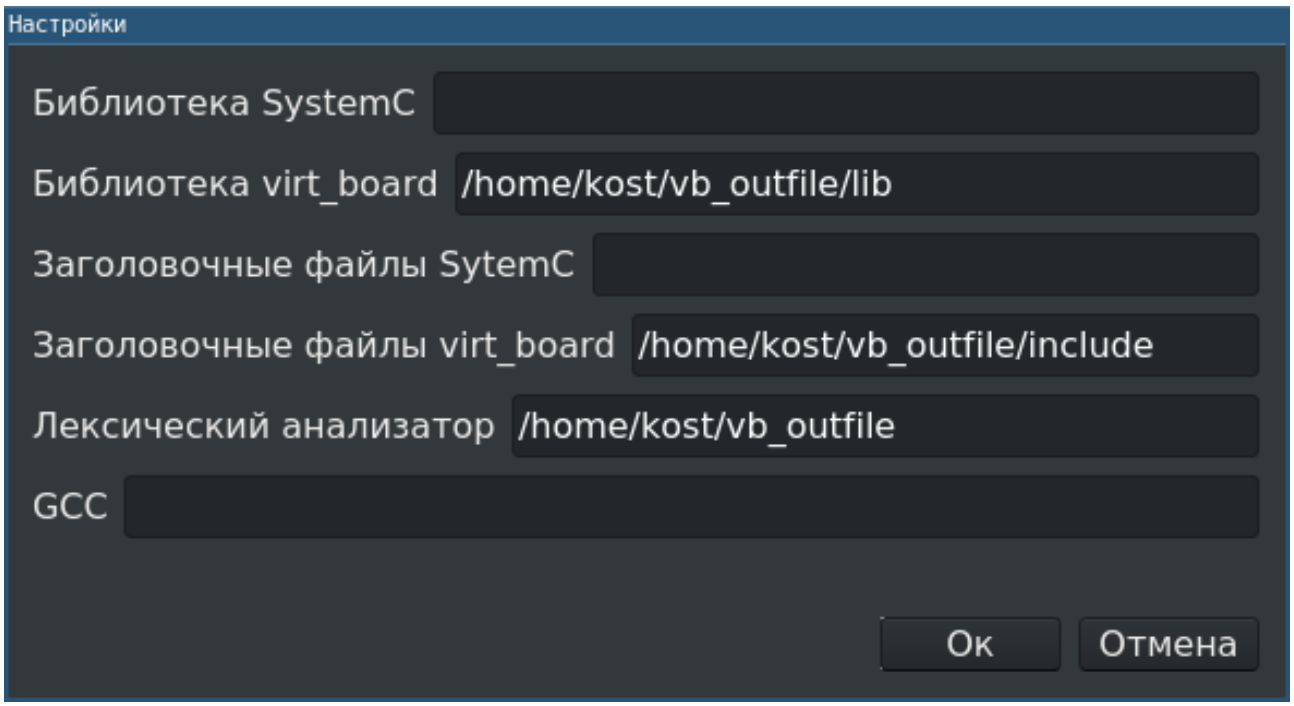

Рис. 18. Внешний вид окна «Настройки»

Так же программу можно настроить в самой программе qtvirtboard в окне «Настройки». Данное окно можно вызвать и меню «Правка»

Расположение файлов библиотеки SystemC и компилятора GCC можно не указывать, если они установлены в системе в стандартных каталогах FHS.

4.2.4. Проверка программы.

Для проверки работоспособности программы используется описание testmod входящее в установленный комплект.

Для проверки нужно выполнить следующие действия:

1. Загрузить описание из распакованного каталога с программой. Для этого нужно открыть окно «Открыть модуль» в меню «Файл».

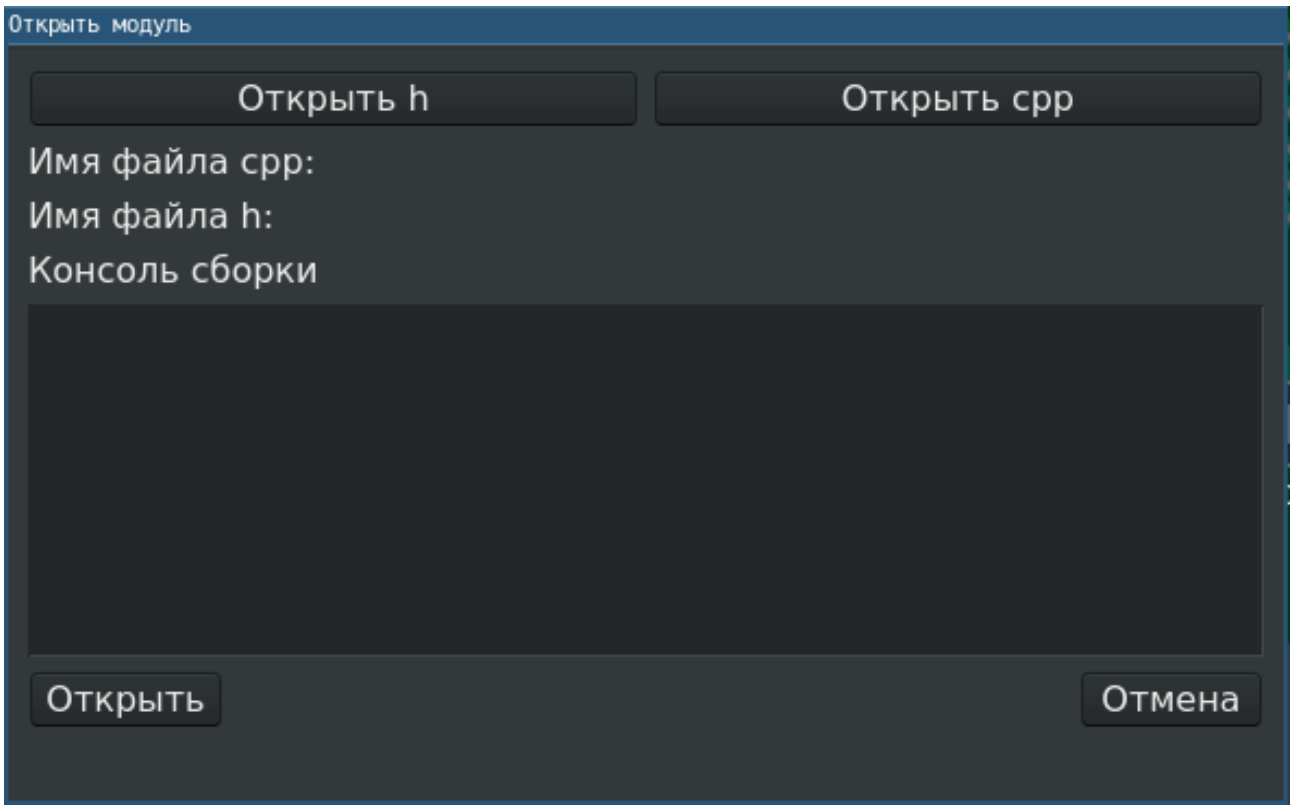

Рис. 19. Внешний вид окна «Открыть модуль».

2. Запустить моделирование нажатием на кнопку «Старт».
3. При этом этом на в основном окне загорится индикатор led1, а на обоих семисегментных индикаторах будет отображаться значение 0.
4. 10 раз нажать и отжать значение btn1. При этом значение на семисегментных индикаторах будет увеличиваться на 1 при каждом нажатии кнопки. Также при каждом нажатии кнопки будет загораться светодиодный индикатор справа от того, который горит сейчас. Если такого нет, то должен загореться led1.
5. 10 раз нажать и отжать значение btn5. При этом значение на семисегментных индикаторах будет увеличиваться на 1 при каждом нажатии кнопки. Также при каждом нажатии кнопки будет загораться светодиодный индикатор слева от того, который горит сейчас. Если такого нет, то должен загореться led5.

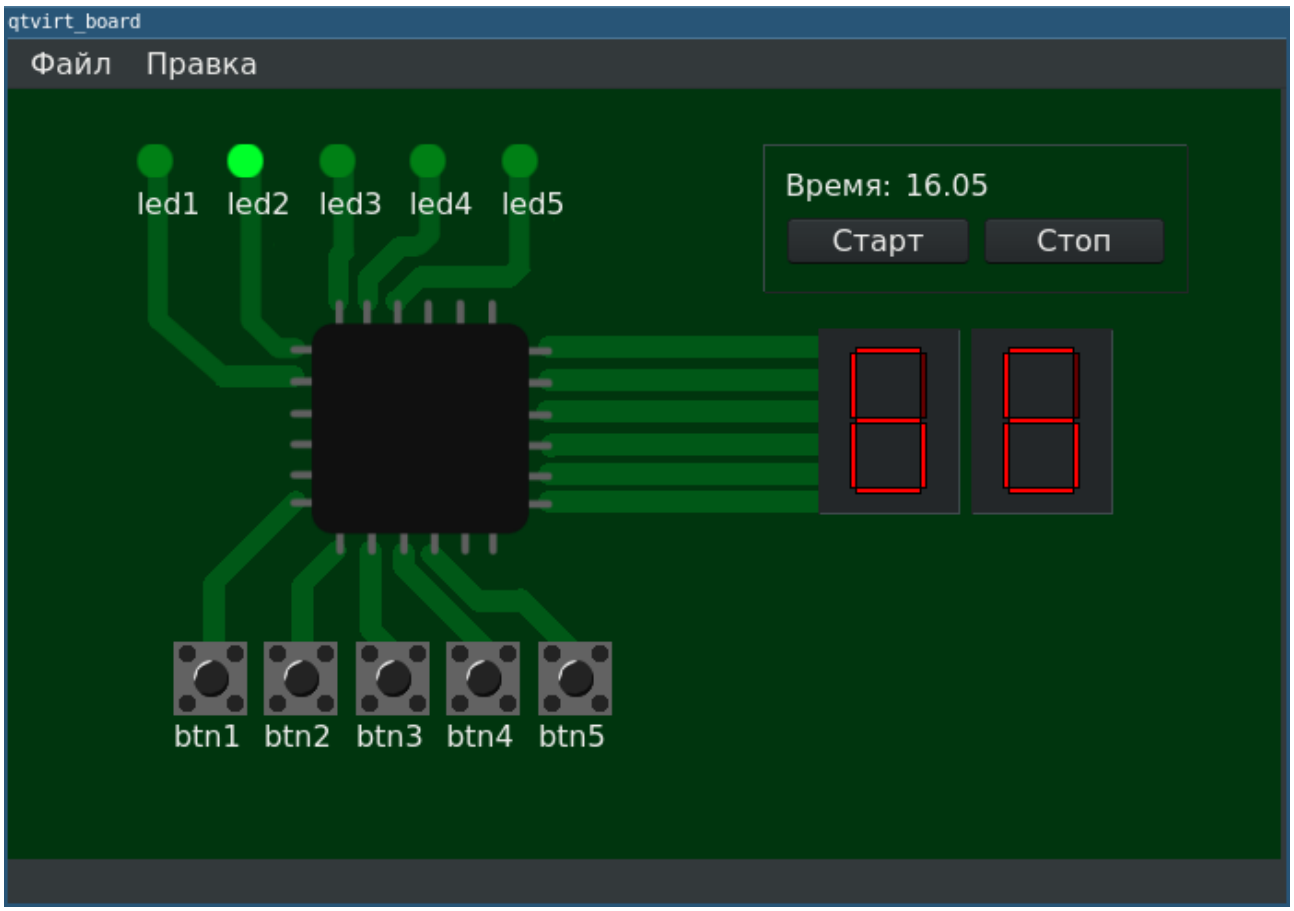

Рис. 20. Внешний вид окна основного окна программы.

4.2.5. Сообщения системному программисту.

В данной программе сообщения пользователю посылаются в случае ошибки в виде диалоговых окон.

Перечень ошибок:

«Ошибка компиляции: выбраны не все файлы.»

Данная ошибка появляется в том случае, если не выбран заголовочный файл или файла реализации описания загружаемого устройства.

«Ошибка компиляции: ошибка при компиляции файла модуля»

Данная ошибка появляется в том случае если в файлах описания имеются синтаксические ошибки. Так же причиной может быть неправильно указанный путь к заголовочным файлам. Возможно не указан путь к компилятору GCC. В случае ошибки необходимо просмотреть в текстовом редакторе файл error_compile располагающийся в директории для временных файлов системы(В ОС Linux такой директорией является /tmp/).

«Ошибка генерации: Ошибка при генерации файла main»

Данная ошибка появляется в том случае, если не удалось сгенерировать файл содержащий функцию sc_main. Такая ошибка может возникнуть, когда не удалось распознать порты заголовочном файле описания с расширением h. Причиной может быть неправильно указанный при настройке путь лексическому анализатору vbllex

«Ошибка компиляции: Ошибка при компиляции файла main»

Данная ошибка появляется в том случае, если не удалось откомпилировать файл содержащий функцию sc_main. В случае ошибки необходимо просмотреть в текстовом редакторе файл error_compile располагающийся в директории для временных файлов системы.

«Ошибка компиляции: Ошибка при компиляции модели»

Данная ошибка появляется в том случае, не удалось скомпилировать исполняемый файл для запуска процесса моделирования. Такая ошибка может происходить в случае если неправильно указан путь к библиотекам vb_testbench и vb_io. Так же может быть не найдена библиотека SystemC. Требуется настроить программу в окне «Настройки». В случае ошибки необходимо просмотреть в текстовом редакторе файл error_compile располагающийся в директории для временных файлов системы.

4.3. Руководство программиста

4.3.1. Назначение программы.

Программа qtvirtboard предназначена для моделирования взаимодействия человека и отладочной платы на компьютере. Данная программа позволяет загрузить описание устройства и взаимодействовать с ним в реальном времени.

Минимальные требования к аппаратным средствам:

1. Процессор архитектуры x86 с тактовой частотой не менее 2 ГГц.
2. Объем оперативной памяти не менее 1 Гб.
3. Объем свободной памяти на диске не менее 256МБ.

Требования к программным средствам:

1. Операционная система поддерживающая стандарт POSIX (Например, Ubuntu 18.04, Debian GNU/Linux 9).
2. Наличие пакета GNU Compile Collection (набор компиляторов проекта GNU).
3. Интерпретатор командой строки bash.
4. Наличие библиотеки SystemC версии не ниже 2.2.
5. Наличие программы flex версии не ниже 2.6.
6. Наличие библиотеки Qt для разработчиков версии не ниже 5.9.
7. Пакет утилит GNU Binary Utilities.
8. Пакет утилит GNU Core Utilities.
9. Программа Make версии не ниже 4.1
10. Программа-архиватор tar версии не ниже 1.29

4.3.2. Характеристика программы.

Данная программа для моделирования взаимодействует с различными программными средствами. В первую очередь с компилятором. Это необходимо для осуществления компиляции модели описания устройства. Так же программа взаимодействует с лексическим анализатором. Лексический анализатор возвращает информацию о портах, которые есть в описании устройства и используется на стадии компиляции. Время работы совместной

обоих программ может меняться в зависимости от сложности описания SystemC. Обычно это время составляет несколько секунд.

Модель представляет собой программу запускаемую параллельно основной. Данная программа с периодичностью в 50 мс посылает основной информацию о состоянии модуля SystemC внутри модели. Программа имеет два режима работы:

- моделирование запущено;
- моделирование остановлено.

Основная программа qtvirtboard имеет следующие режимы работы:

- «Модуль не загружен» - при запуске программы.
- «Компиляция модели» - ожидание окончания процесса компиляции модели.
- «Модуль загружен» - основной режим работы, при котором происходит моделирование.

4.3.3. Обращение к программе.

Обращение к программе может происходить через командную строку с помощью команды

```
/путь/к/программе/qtvirt_board
```

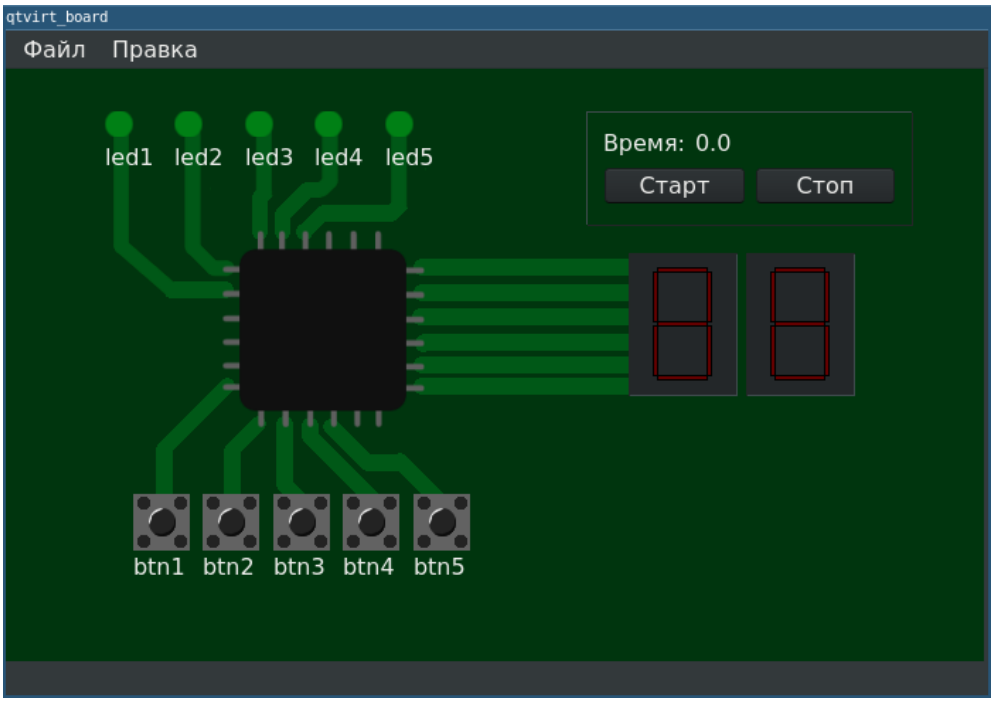

Рис. 21. Внешний вид основного окна программы.

Для загрузки описания устройства в программе есть окно «Открыть модуль», расположенное в меню «Файл», где нужно указать:

Имя заголовочного файла описания с расширением h.

Имя файла реализации описания с расширение cpp.

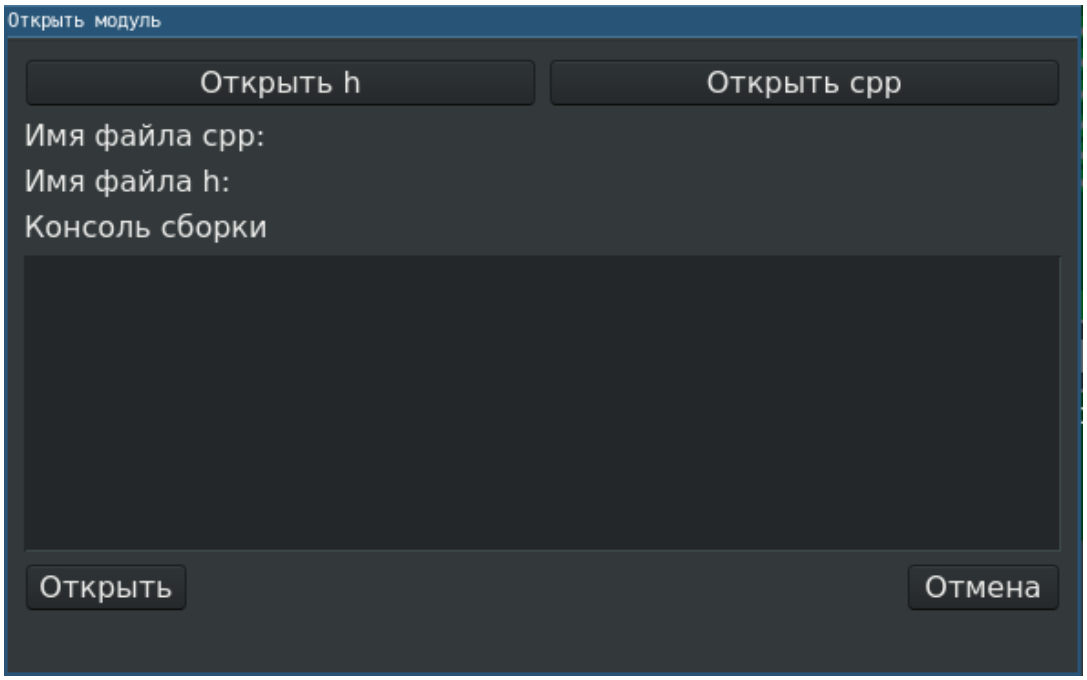

Рис. 22. Внешний вид окна «Открыть модуль».

4.3.4. Входные и выходные данные.

Входными данными являются описание устройства написанное для библиотеки SystemC написанный в двух файлах:

1. Заголовочный файл с расширением h содержащий описание класса модуля.
2. Файл реализации с расширением cpp.

Требования к файлам описания SystemC:

1. Описание не должно содержать подмодулей.
2. Входные порты должны быть типа bool и могут называться только btn1, btn2, btn3, btn4, btn5.(Порты будут соответствовать кнопкам)
3. Выходные порты могут быть либо bool и могут называться led1, led2, led3, led4, led5 (Порты будут соответствовать светодиодным индикаторам), либо sc_bv<7> и могут называться sevseg1,

sevseg2(Порты будут соответствовать семисегментным индикаторам индикаторам).

Программа взаимодействует с пользователем по средством графического интерфейса при моделировании:

1. Кнопки btn1, btn2, btn3, btn4, btn5.
2. Светодиодные индикаторы led1, led2, led3, led4, led5.
3. Семисегментные индикаторы sevseg1, sevseg2.
4. Табло отображения времени.
5. Кнопки «Старт» и «Стоп» для управления процессом моделирования.

4.3.5. Сообщения.

В данной программе сообщения пользователю посылаются в случае ошибки в виде диалоговых окон

Перечень ошибок:

«Ошибка компиляции: выбраны не все файлы.»

Данная ошибка появляется в том случае, если не выбран заголовочный файл или файла реализации описания загружаемого устройства.

«Ошибка компиляции: ошибка при компиляции файла модуля»

Данная ошибка появляется в том случае если в файлах описания имеются синтаксические ошибки. Так же причиной может быть неправильно указанный путь к заголовочным файлам. Возможно не указан путь к компилятору GCC. В случае ошибки необходимо более подробно изучить сообщения в консоли сборки.

«Ошибка генерации: Ошибка при генерации файла main»

Данная ошибка появляется в том случае, если не удалось сгенерировать файл содержащий функцию sc_main. Такая ошибка может возникнуть, когда не удалось распознать порты заголовочном файле описания с расширением h. Причиной может быть неправильно указанный при настройке путь лексическому анализатору vbllex

«Ошибка компиляции: Ошибка при компиляции файла main»

Данная ошибка появляется в том случае, если не удалось откомпилировать файл содержащий функцию sc_main. В случае ошибки необходимо более подробно изучить сообщения в консоли сборки.

«Ошибка компиляции: Ошибка при компиляции модели»

Данная ошибка появляется в том случае, не удалось скомпилировать исполняемый файл для запуска процесса моделирования. Такая ошибка может происходить в случае если неправильно указан путь к библиотекам vb_testbench и vb_io. Так же может быть не найдена библиотека SystemC. Требуется настроить программу в окне «Настройки». В случае ошибки необходимо более подробно изучить сообщения в консоли сборки.

4.4 Руководство оператора.

4.4.1. Назначение программы.

Программа qtvirtboard предназначена для моделирования взаимодействия человека и отладочной платы на компьютере. Данная программа позволяет загрузить описание устройства и взаимодействовать с ним в реальном времени.

Возможности программы qtvirtboard:

1. Программа способна моделировать реакцию отладочной платы на нажатие кнопок.
2. Программа моделирует взаимодействие отладочной платы с пользователем по средством индикаторов.

Ограничение программы:

1. Данная программа предназначена в первую очередь для использования в операционных системах поддерживающих стандарт POSIX.
2. Описание устройства не должно содержать подмодулей.

4.4.2. Условия применения.

Минимальные требования к аппаратным средствам:

1. Процессор архитектуры x86 с тактовой частотой не менее 2 ГГц.
2. Объем оперативной памяти не менее 1 Гб.
3. Объем свободной памяти на диске не менее 256МБ.

Требования к программным средствам:

1. Операционная система поддерживающая стандарт POSIX (Например, Ubuntu 18.04, Debian GNU/Linux 9).
2. Наличие пакета GNU Compile Collection (набор компиляторов проекта GNU).
3. Интерпретатор командной строки bash.
4. Наличие библиотеки SystemC версии не ниже 2.2.
5. Наличие программы flex версии не ниже 2.6.
6. Наличие библиотеки Qt для разработчиков версии не ниже 5.9.
7. Пакет утилит GNU Binary Utilities.

8. Пакет утилит GNU Core Utilities
9. Программа Make версии не ниже 4.1
10. Программа-архиватор tar версии не ниже 1.29

4.4.3. Выполнение программы.

Исполняемый файл программы `qtvirt_board` находится в каталоге, в который его установил администратор.

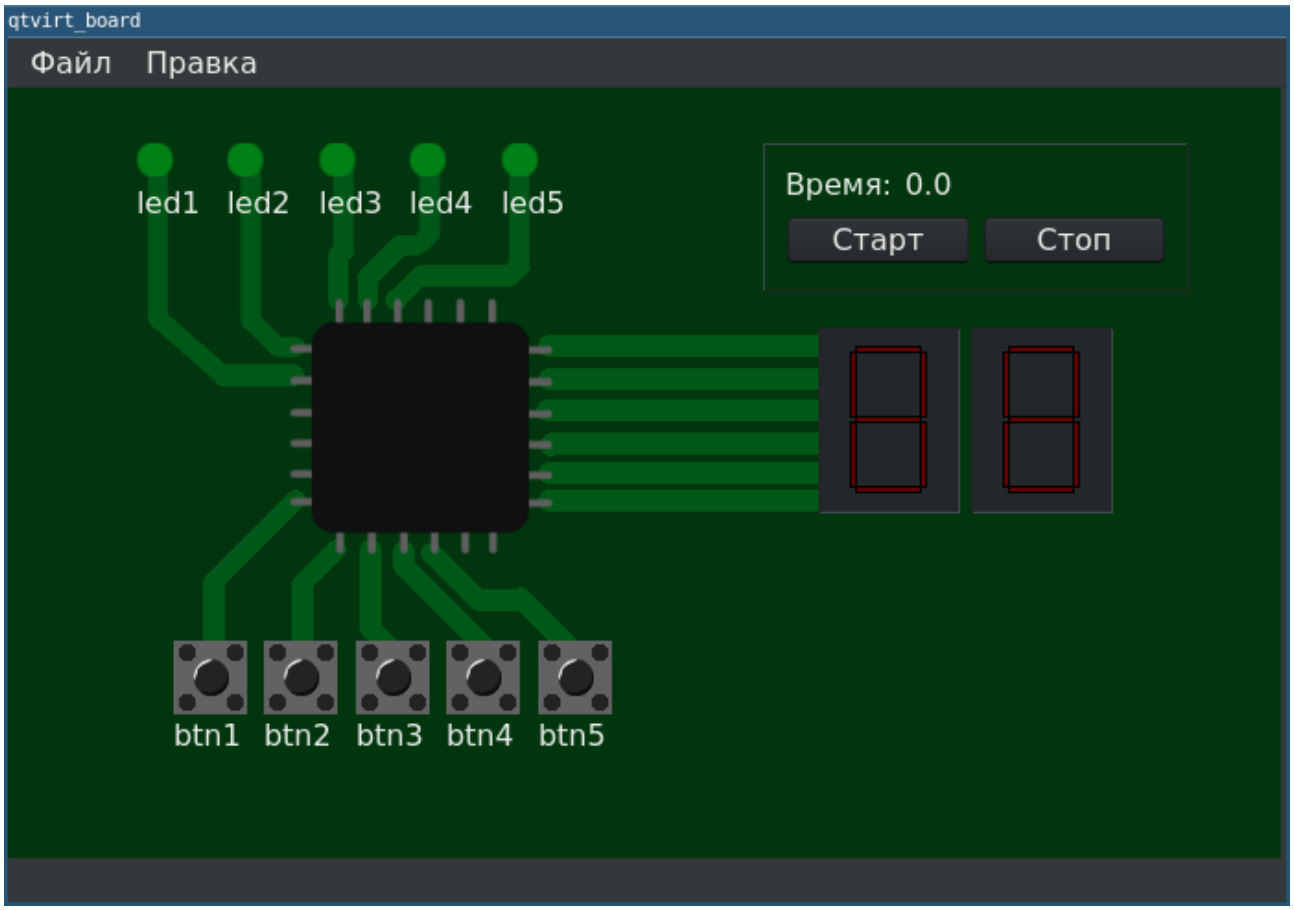

Рис. 23. Внешний вид основного окна программы.

Для загрузки описания устройства в программе есть окно «Открыть модуль», расположенное в меню «Файл», где нужно указать:

Имя заголовочного файла описания с расширением h.

Имя файла реализации описания с расширение cpp.

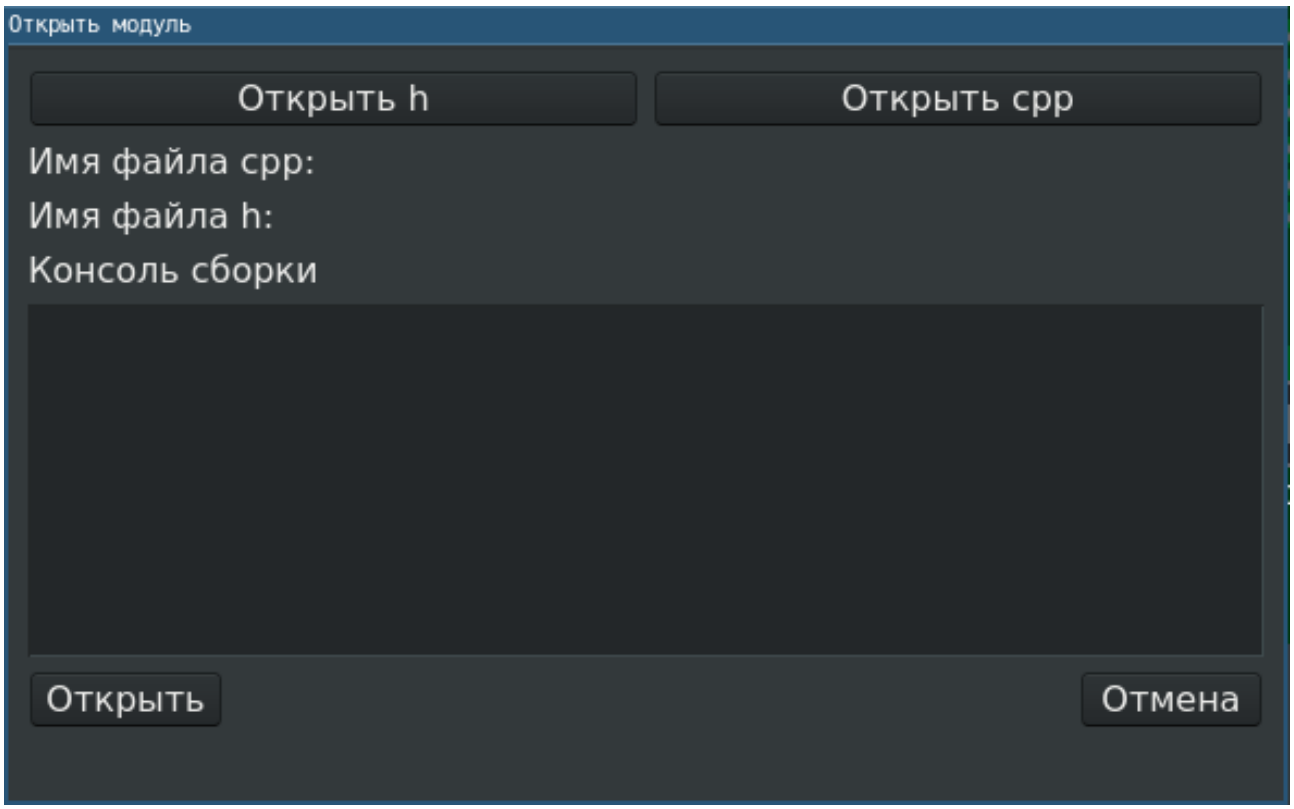

Рис. 24. Внешний вид окна «Открыть модуль».

Требования к файлам описания SystemC:

1. Описание не должно содержать подмодулей.
2. Входные порты должны быть типа bool и могут называться только btn1, btn2, btn3, btn4, btn5.(Порты будут соответствовать кнопкам)
3. Выходные порты могут быть либо bool и могут называться led1, led2, led3, led4, led5 (Порты будут соответствовать светодиодным индикаторам), либо sc_bv<7> и могут называться sevseg1, sevseg2(Порты будут соответствовать семисегментным индикаторам индикаторам).

Программа для взаимодействия с оператором использует:

1. Кнопки btn1, btn2, btn3, btn4, btn5. Данные кнопки предназначены для При нажатии данные кнопки зажимаются и отжимаются при повторном нажатии. Кнопка зажата соответствует значению true, кнопка отжата соответствует значению false.

2. Светодиодные индикаторы led1, led2, led3, led4, led5. Горящий светодиод соответствует значению true, потухший светодиод соответствует значению false.
3. Семисегментные индикаторы sevseg1, sevseg2. Каждый сегмент данного индикатора управляется отдельным битом в описании. Первый бит управляет сегментом A, второй бит управляет сегментом B, и так далее.
4. Табло отображения времени.
5. Кнопки «Старт» и «Стоп» для управления процессом моделирования.

4.4.4. Сообщения оператору.

В данной программе сообщения оператору посылаются в случае ошибки в виде диалоговых окон

Перечень ошибок:

«Ошибка компиляции: выбраны не все файлы.»

Данная ошибка появляется в том случае, если не выбран заголовочный файл или файла реализации описания загружаемого устройства.

«Ошибка компиляции: ошибка при компиляции файла модуля»

Данная ошибка появляется в том случае если в файлах описания имеются синтаксические ошибки. Так же причиной может быть неправильно указанный путь к заголовочным файлам. Возможно не указан путь к компилятору GCC. В случае ошибки необходимо более подробно изучить сообщения в консоли сборки.

«Ошибка генерации: Ошибка при генерации файла main»

Данная ошибка появляется в том случае, если не удалось сгенерировать файл содержащий функцию sc_main. Такая ошибка может возникнуть, когда не удалось распознать порты заголовочном файле описания с расширением h. Причиной может быть неправильно указанный при настройке путь лексическому анализатору vbllex

«Ошибка компиляции: Ошибка при компиляции файла main»

Данная ошибка появляется в том случае, если не удалось откомпилировать файл содержащий функцию sc_main. В случае ошибки необходимо более подробно изучить сообщения в консоли сборки.

«Ошибка компиляции: Ошибка при компиляции модели»

Данная ошибка появляется в том случае, не удалось скомпилировать исполняемый файл для запуска процесса моделирования. Такая ошибка может происходить в случае если неправильно указан путь к библиотекам vb_testbench и vb_io. Так же может быть не найдена библиотека SystemC. Требуется настроить программу в окне «Настройки». В случае ошибки необходимо более подробно изучить сообщения в консоли сборки.

5. КОНТРОЛЬНЫЕ ПРИМЕРЫ.

В качестве контрольных примеров выберем два модуля написанных на языке SystemC:

- Модуль моделирующий элемент 4ИЛИ-НЕ nor4
- Модуль testmod.

5.1. Контрольный пример: элемент 4ИЛИ-НЕ.

Описание модуля nor4:

Таблица истинности 4ИЛИ-НЕ:

A	B	C	D	F
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	0
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	0

nor4.h:

```
#include <systemc.h>

SC_MODULE(nor4){
    sc_in<bool> btn1, btn2, btn3, btn4;
    sc_out<bool> led3;
```

```
    void do_nor4();
    SC_CTOR(nor4){
        SC_METHOD(do_nor4);
        sensitive << btn1 << btn2
                  << btn3 << btn4;
    }
};
```

nor4.cpp:

```
#include "nor4.h"

void nor4::do_nor4(){
    led3.write(!(btn1.read() || btn2.read() ||
                btn3.read() || btn4.read()));
}
```

Откроем модуль в окне «Открыть модуль»:

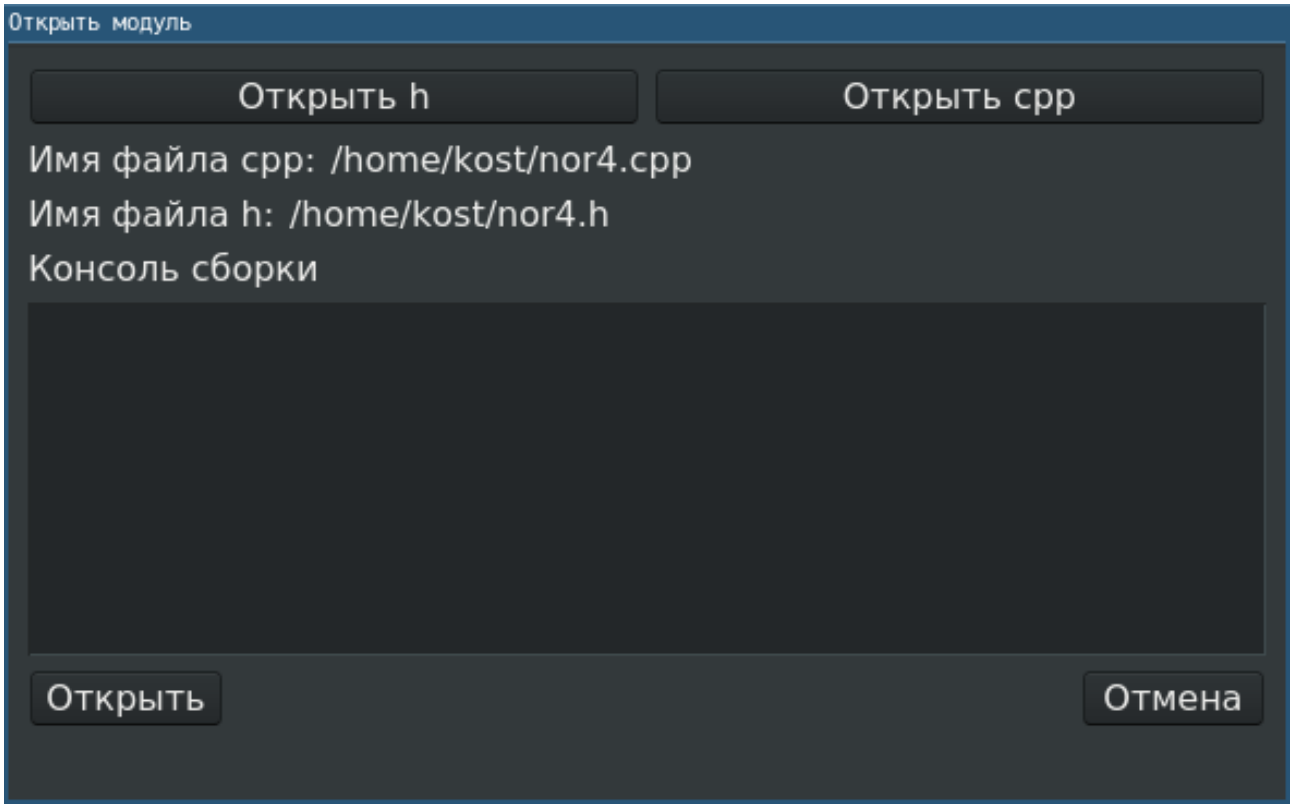

Рис. 25. Открытия нового модуля.

И начнем проверку нескольких нескольких случаев из таблицы истинности:

btn1 = 0, btn2 = 0, btn3 = 0, btn4 = 0; led3 = 1.

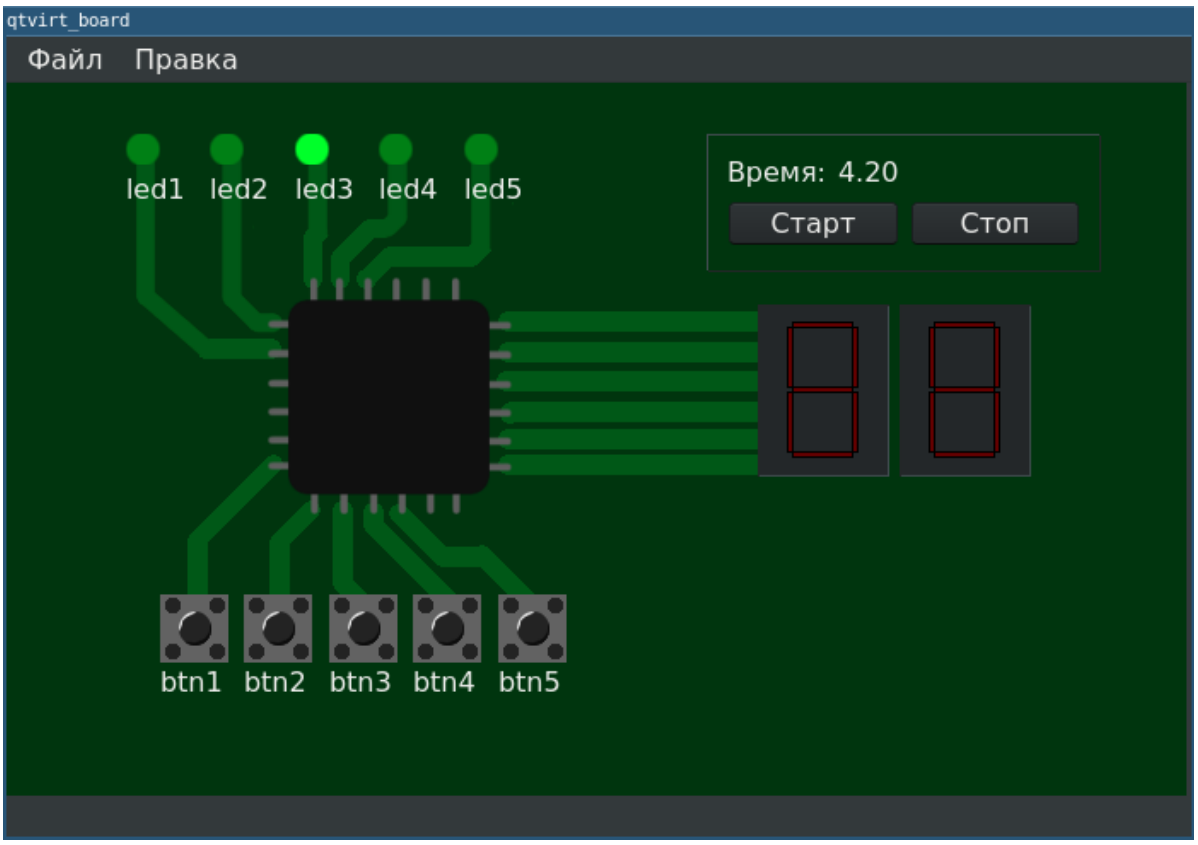

Рис. 26. Моделирование модуля nor4(первый случай).

btn1 = 1, btn2 = 1, btn3 = 0, btn4 = 1; led3 = 0.

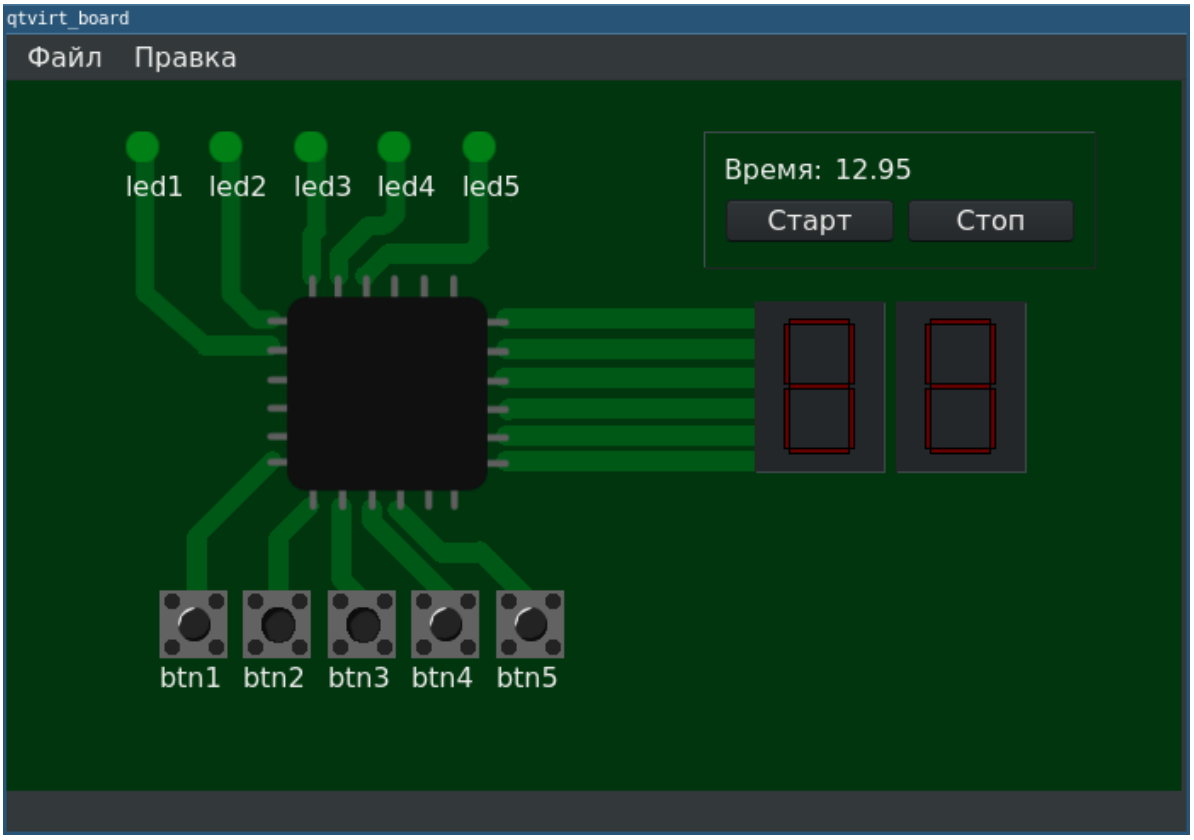

Рис. 27. Моделирование модуля nor4(второй случай).

Контрольный пример: модуль из комплекта поставки testmod.

Описание работы:

По переднему фронту сгнала btn1(При зажатии) переменная, хранящая внутреннее состояние модуля увеличивается на 1;

По переднему фронту сгнала btn1(При зажатии) переменная, хранящая внутреннее состояние модуля уменьшается на 1;

Последняя цифра переменной внутреннего состояния отображается на семисегментных индикаторах.

Так же зависимости от значения последней цифры внутреннего состояния зависит, то какой светодиод загорится: led1 при 0 и 5, led2 при 1 и 6. led3 при 2 и 7, led4 при 3 и 8, led5 при 4 и 9.

Описание данного модуля на языке SystemC находится можно посмотреть в листинге программы.

Загрузим несколько файлы описания модуля testmod:

Нажмем и отожмем 3 раза кнопку btn1. При этом у виртуальной отладочной платы будет следующее состояние:

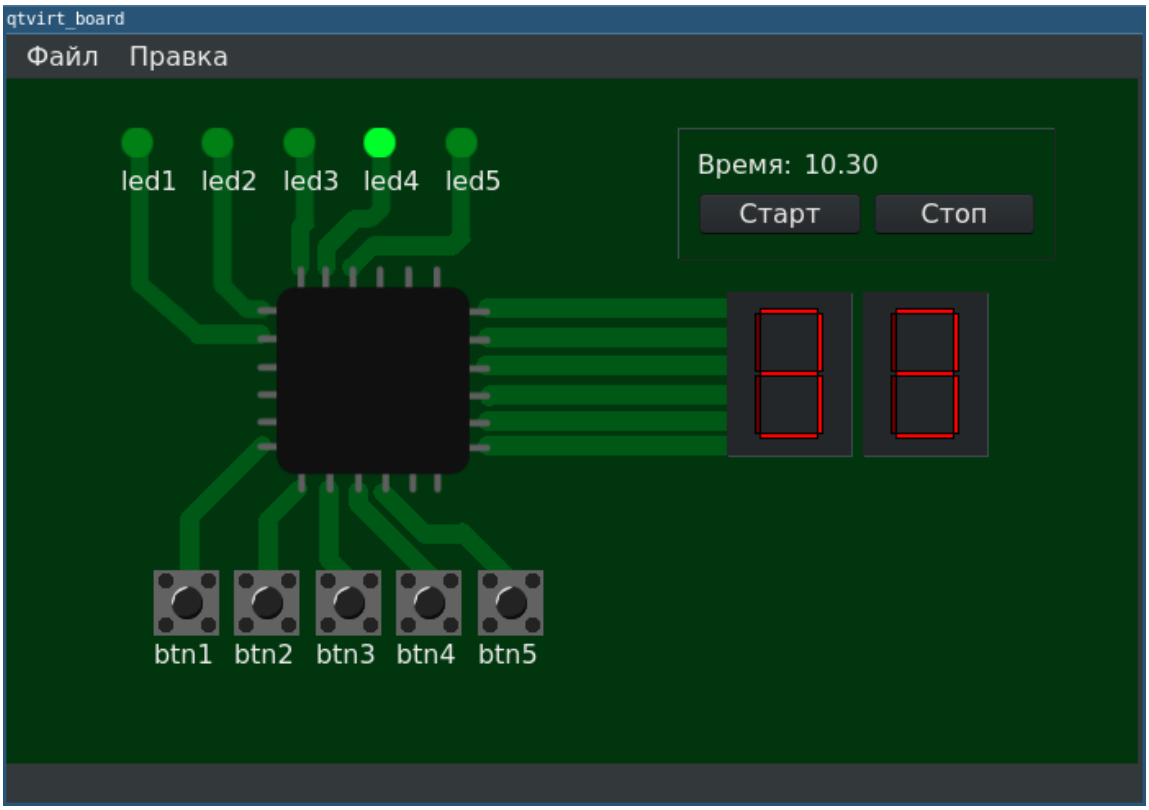

Рис. 28. Моделирование модуля testmod(первый случай).

Нажмем и отожмем 1 раза кнопку btn1. При этом у виртуальной отладочной платы будет следующее состояние:

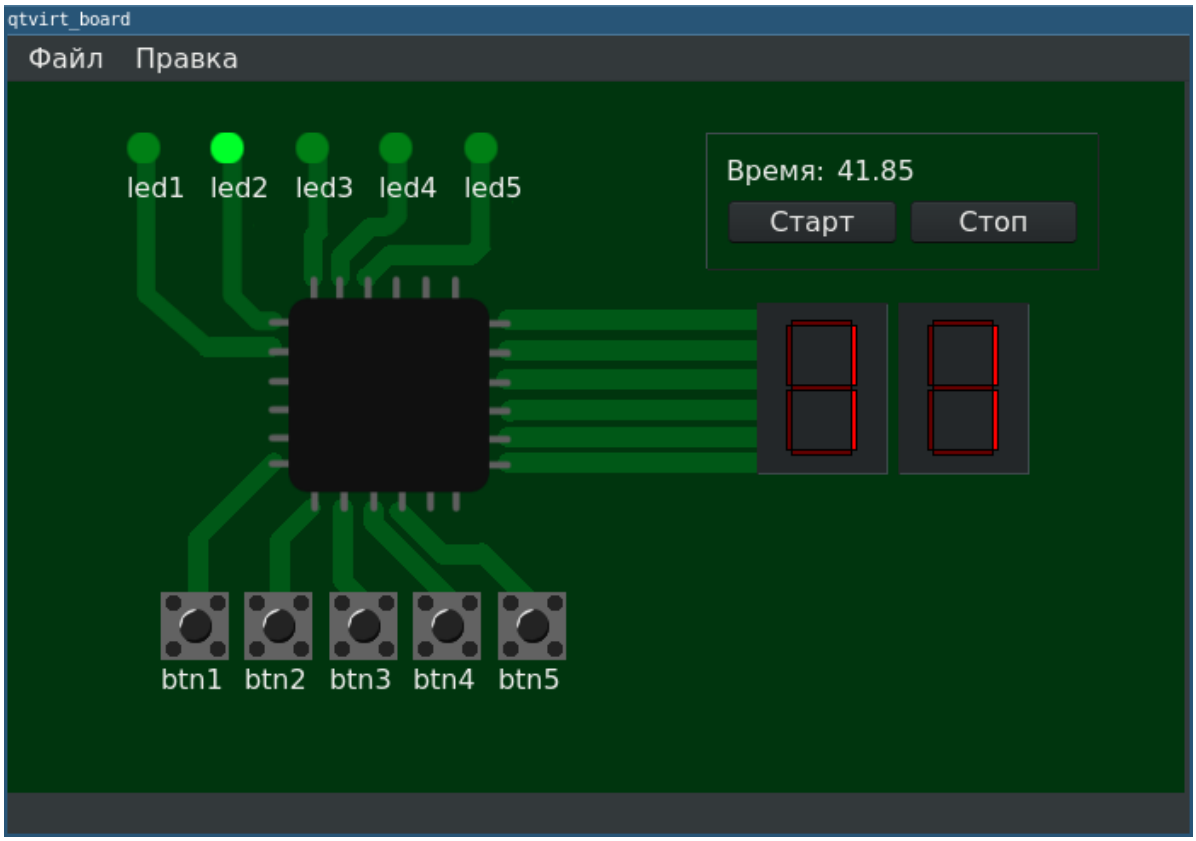

Рис. 29. Моделирование модуля testmod(второй случай).

ЗАКЛЮЧЕНИЕ

В данной работе была разработана программа, которая облегчает моделирование устройств написанных на языке SystemC для операционных систем POSIX. При этом взаимодействие в с моделью устройства и пользователя происходит через понятный интерфейс, имитирующий взаимодействие с макетной платой.

Так же в процессе разработки были углублены знания по работе с библиотекой SystemC. Была проделана работа по изучению стандартов в UNIX-подобных системах. Были изучены принципы взаимодействия программ между собой.

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ.

1. Лав, Р. Linux. Системное программирование. 2-е изд. — СПб.: Питер, 2014. — 448 с.
2. UNIX. Профессиональное программирование. 3-е изд. — /У. Ричард Стивенс, Стивен А. Раго — СПб.: Питер, 2018. — 944 с.
3. Программирование на языке C++ в среде Qt Creator. — / Е.Р. Алексеев, Г.Г. Злобин, Д.А Костюк. О.В. Чеснокова, А.С Чмыхало — М.: ALT Linux. 2015 — 448с.
4. Hardware description language[Электронный ресурс] — режим доступа: https://en.wikipedia.org/wiki/Hardware_description _language (Дата обращения 23.05.2019)
5. SystemC [Электронный ресурс] — режим доступа: https://en.wikipedia.org/wiki/SystemC (Дата обращения 23.05.2018)
6. Verilog [Электронный ресурс] — режим доступа: https://en.wikipedia.org/wiki/Verilog (Дата обращения 23.05.2019)
7. VHDL [Электронный ресурс] — режим доступа: https://en.wikipedia.org/wiki/VHDL (Дата обращения 23.05.2019)
8. Spatran-6 family overview [Электронный ресурс] — режим доступа: https://www.xilinx.com/support/documentation/ data_sheets/ds160.pdf (Дата обращения 26.05.2019)
9. Xilinx Spartan 6 FPGA Boards [Электронный ресурс] — режим доступа: https://numato.com/docs/mimas-v2-spartan-6-fpga-development-board-with-ddr-sdram/ (Дата обращения 26.05.2019)
10. Functional specification for SystemC 2.0, 2002, 135p
11. SystemC 2.0 User’s Guide, 2002, 198p
12. Cygwin [Электронный ресурс] — режим доступа: https://www.cygwin.com/ (Дата обращения 27.05.2019)
13. POSIX® 1003.1 Frequently Asked Questions (FAQ Version 1.16) [Электронный ресурс] — режим доступа: http://www.opengroup.org/austin/papers/posix_faq.html (Дата обращения 27.05.2019)

14. Lexical Analysis With Flex, for Flex [Электронный ресурс] — режим доступа: https://github.com/westes/flex/blob/master/doc/flex.texi (Дата обращения 27.05.2019)

ПРИЛОЖЕНИЯ

Приложение А1. Исходный код vb_io.h

```
#ifndef VB_IO_H
#define VB_IO_H

#include <string>
#include <vector>
#include "vb_error/vb_error.h"//В заголовочном файле описана обработка
ошибок

//Структура для передачи команд
struct cmd{
    std::string name;//Имя команды
    long value;//Операнд
};

//Массив команд
typedef std::vector<cmd> vec_cmds;

/* Чтенеие данных и команд из stdin */
bool read_cmds(vec_cmds * cctl, vec_cmds * cdata);

/* Запись данных и команд в stdout */
void write_cmds(vec_cmds * cmds,vec_cmds * cdata);

/*Функция преобразует команды в строку */
void vec_cmds2str(char * str,vec_cmds * cctl, vec_cmds * cdata);

/*Функция преобразует строку в команды */
void str2vec_cmds(vec_cmds * cctl, vec_cmds * vcd, char * str);

#endif
```

Приложение А2. Исходный код vb_io.cpp

```
#include <sys/select.h>//В заголовочном файле описана функция select()
#include <unistd.h>/*В данном заголовочном файле описаны функции
                     *стандартные функции UNIX*/
#include <string.h>
#include "vb_io/vb_io.h"
#include <time.h>//В данном заголовочном файле описаны POSIX таймеры
#include <stdio.h>

//Длины для различных буферов
#define LEN_BUF 8192
#define LEN_NAME 1024

char inbuf[LEN_BUF];//Входной буфер
char outbuf[LEN_BUF];//Выходной буфер

/* Чтенеие данных и команд из stdin */
bool read_cmds(vec_cmds * cctl, vec_cmds * cdata){
    //Очистить массивы команд
    cctl->clear();
    cdata->clear();
    //Инициализация функции select()
    struct timeval zerotime = {0, 0};//Время ожидания у функции select
```

```
    int ret;
    fd_set stdin_set;
    FD_ZERO(&stdin_set);
    FD_SET(STDIN_FILENO,&stdin_set);
    /*Функция select ожидает появлениянновых данных в потоке
     *в течение некоторого времени timerval*/
    ret = select(STDIN_FILENO + 1,
                 &stdin_set,
                 NULL, NULL, &zerotime);
    if(ret == -1) vb_error(VB_IO);//Ошибка в функции select
    if(ret != 0){
        //Считать данные из stdin если есть данные
        ret = read(STDIN_FILENO,inbuf,LEN_BUF);
        if(ret == -1) vb_error(VB_IO);
        //Преобразовать команды
        str2vec_cmds(cctl,cdata,inbuf);
        return true;
    }
    if((cctl->size() == 0)&&(cdata->size() == 0))
        //Если данных нет
        return false;
}

/*Функция преобразует данные и команды в строку */
void vec_cmds2str(char * str, vec_cmds * cctl, vec_cmds * cdata){
    vec_cmds::iterator i;
    str[0] = '\0';
    char strcmd[LEN_NAME];
    strcmd[0] = '\0';
    //Цикл в котором в строку добавляются команды
    for(i = cctl->begin();
        i != cctl->end();
        i++)
    {
        sprintf(strcmd,"#%s:%li\n",(*i).name.c_str(),(*i).value);
        strcat(str,strcmd);
    }
    vec_cmds::iterator j;
    //Цикл в котором в стоку добавляются данные
    for(j = cdata->begin();
        j != cdata->end();
        j++)
    {
        sprintf(strcmd,"-%s:%li\n",(*j).name.c_str(),(*j).value);
        strncat(str,strcmd,LEN_BUF);
    }
}

/*Функция преобразует строку в данные и команды */
void str2vec_cmds(vec_cmds * cctl, vec_cmds * cdata, char * str){
    cctl->clear();
    cdata->clear();
    //Указатель p1 указывает на еще не прочитанную часть буфера
    char * p1 = str;
    //Указатель p2 нужен для поиска очередной строки в буфере
    char * p2 = p1;
    char namebuf[LEN_NAME];
```

```
    long value;
    char type_cmd;
    cmd tmp;
    while(true){
        //Нахождение окончания строки
        p2 = index(p1,'\n');
        if(p2 == NULL) break;
        *p2 = '\0';
        if(sscanf(p1,"%c%[^:]:%li",
                  &type_cmd,namebuf,&value) == 3){
            //Если строка прочитана корректно
            tmp.name = namebuf;
            tmp.value = value;
            switch(type_cmd){
                case '#':
                    //Добавление в список команд
                    cctl->push_back(tmp);
                    break;
                case '-':
                    //Добавление с список данных
                    cdata->push_back(tmp);
                    break;
            }
        }
        //переход к следующей строке
        p1 = p2 + 1;
    }
}

/*Функция преобразует строку в данные и команды */
void write_cmds(vec_cmds * cctl, vec_cmds * cdata){
    vec_cmds2str(outbuf,cctl,cdata);
    std::cout << outbuf << std::endl;
    cctl->clear();
    cdata->clear();
}
```

Приложение А3. Исходный код vb_testbench.h

```
/* vb_testbench.h
 * Заголовочный файл тестера*/

#ifndef VB_TESTBENCH_H
#define VB_TESTBENCH_H

#include <systemc.h>
#include "vb_timer/vb_timer.h"
#include "vb_io/vb_io.h"
#include <stdlib.h>

#define DEF_STEP_MODEL 100 //Переодичность опросов по умолчанию
#define TM_MS(s) sc_time( (s), SC_MS) //Макрос для перевода int в sc_time

class vb_testbench : public sc_module{
   private:
        vec_cmds cdatain;//Список команд ввода данных
        vec_cmds cdataout;//Список команд управления
        vec_cmds cctlin;//Список команд вывода данных
```

```
        vec_cmds cctlout;//Список команд вывода информации о
моделировании
        int _model_step;//Переодичность опроса в мс
        bool stopped = true;//Флаг остановки моделирования
    public:
        vb_timer sync_timer;/*vb_timer с помощью которого осуществляется
                             *синзронизация реального и модельного
времени*/
        sc_out<bool> btn1, btn2, btn3, btn4, btn5;//Порты для кнопок
        sc_in<bool> led1, led2, led3, led4, led5;/*Порты для светодиодных
                                                  *индикаторов*/
        sc_in<sc_bv<7>> sevseg1, sevseg2;/*Порты для семисегментных
                                          *индикаторов*/
        virtual void handlcctl(const cmd & cctl);/*Обработка команд
                                                  *управления от
пользователя*/
        virtual void func_outcctl(vec_cmds * cctl);/*Составление списка
                                                    *выходных команд
для пользователя*/
        virtual void handlcdata(const cmd & cdata);/*Обработка входных
                                                    *данных от
пользователя*/
        virtual void func_outcdata(vec_cmds * cdata);/*Составление списка
                                                      *выходных данных для
пользователя*/
        void setStopped(bool);//Функция для остоновки/возобновления
моделирования
        void test_thread();/*Процесс SC_THREAD который передает данные от
                            *пользователя в тестируемый модуль*/
        typedef vb_testbench SC_CURRENT_USER_MODULE;//Служебная строка
для библиотеки SystemC
        vb_testbench(sc_module_name,int); //Конструктор класса
vb_testbench
};
#endif
```

Приложение А4. Исходный код vb_testbench.cpp

```
/* vb_testbench.cpp
 * исходный код тестера*/

#include "vb_testbench/vb_testbench.h"
#include "vb_error/vb_error.h"
#include "vb_timer/vb_timer.h"
#include "vb_io/vb_io.h"

 /*Процесс SC_THREAD который передает данные от
  *пользователя в тестируемый модуль*/
void vb_testbench::test_thread(){
    sync_timer.set(_model_step);//Установка переодичности опроса
    unsigned time_timer;/*Количество истечений таймера с прошлого
                         *вызова vb_timer::wait()*/
     while(1){//Бесконечный цикл
        time_timer = sync_timer.wait();//Ожидание истечения таймера
        if(read_cmds(&cctlin,&cdatain)){//Считывание данных
            vec_cmds::iterator i;
            for(i = cctlin.begin();//Цикл, в котором обрабатываются
                i != cctlin.end(); //поступающие команды
```

```
                i++)
            {
                handlcctl(*i);

            }
            if(stopped == false){//Проверка флага остановки моделирования
                vec_cmds::iterator j;
                for(j = cdatain.begin();//Цикл, в котором обрабатываются
                    j != cdatain.end(); //Поступающие данные
                    j++)
                {
                    handlcdata(*j);
                }
            }
        }
        if(stopped == false)
            wait(TM_MS(time_timer * _model_step));/*продвижение
модельного
                                                    *времени */
        func_outcctl(&cctlin);//Составление выходных команд
        if(stopped == false)
            func_outcdata(&cdatain);//Составление выходных данных
        write_cmds(&cctlin,&cdatain);//Вывод данных и команд
    }
}

/*Функция для остоновки/возобновления моделирования*/
void vb_testbench::setStopped(bool flg){
    stopped = flg;
}

/*Обработка входных данных от пользователя*/
void vb_testbench::handlcdata(const cmd & cdata){
    /* Здесь происходит проверка - какому порту предназначались данные*/
    if(cdata.name.compare("btn1") == 0){
        btn1.write((cdata.value == 0) ? false : true);
    }else
    if(cdata.name.compare("btn2") == 0){
        btn2.write((cdata.value == 0) ? false : true);
    }else
    if(cdata.name.compare("btn3") == 0){
        btn3.write((cdata.value == 0) ? false : true);
    }else
    if(cdata.name.compare("btn4") == 0){
        btn4.write((cdata.value == 0) ? false : true);
    }else
    if(cdata.name.compare("btn5") == 0){
        btn5.write((cdata.value == 0) ? false : true);
    }
}

/*Обработка входных команд от пользователя*/
void vb_testbench::handlcctl(const cmd & cctl){
    /* Здесь происходит проверка - какая именно команда поступила*/
    if(cctl.name.compare("quit") == 0){
        exit(cctl.value);
    }else
```

```
    if(cctl.name.compare("start") == 0){
        setStopped(false);
    }else
    if(cctl.name.compare("stop") == 0){
        setStopped(true);
    }
}

/* Функция котороая преобразует тип sc_bv<7> к типу long*/
long bv7_to_long(sc_bv<7> val){
    long retval = 0;//Возвращемое значение устанвливается в 0
    for(char i = 0; i < 7; i++){
        retval |= ((val[6-i]=='1') ? 1 : 0) << i; /*Если i-ый бит
sc_bv<7>
                    установлен в '1' то соответсвующий бит retval
                    устанавливается 1 иначе 0*/
    }
    return retval;
}

/*Составление списка выходных данных для пользователя*/
void vb_testbench::func_outcdata(vec_cmds * cdata){
    cmd tmp;
    tmp.name = "led1";
    tmp.value = (led1.read() == false) ? 0 : 1;
    cdata->push_back(tmp);
    tmp.name = "led2";
    tmp.value = (led2.read() == false) ? 0 : 1;
    cdata->push_back(tmp);
    tmp.name = "led3";
    tmp.value = (led3.read() == false) ? 0 : 1;
    cdata->push_back(tmp);
    tmp.name = "led4";
    tmp.value = (led4.read() == false) ? 0 : 1;
    cdata->push_back(tmp);
    tmp.name = "led5";
    tmp.value = (led5.read() == false) ? 0 : 1;
    cdata->push_back(tmp);
    tmp.name = "sevseg1";
    tmp.value = bv7_to_long(sevseg1.read());
    cdata->push_back(tmp);
    tmp.name = "sevseg2";
    tmp.value = bv7_to_long(sevseg2.read());
    cdata->push_back(tmp);
}

/*Составление списка выходных данных для пользователя*/
void vb_testbench::func_outcctl(vec_cmds * cctl){
    cmd tmp;
    tmp.name = "time";
    tmp.value = (long)(sc_time_stamp().to_seconds() * 1000);
    cctl->push_back(tmp);
}

//Конструктор класса vb_testbench
vb_testbench::vb_testbench(sc_module_name nm,
                           int model_step = DEF_STEP_MODEL):
```

```
sc_module(nm){
    _model_step = model_step;//Установка времени моделирования
    SC_THREAD(test_thread);/*Регистрация функции test_thread как
SC_THREAD
                            *в ядре моделирования*/
}
```

Приложение А5. Исходный код lex.gen

```
%{
    #include<stdio.h>
    int fsm = 0;
%}

%%


SC_MODULE   if( fsm == 0){ fsm = 1;}else exit(1);
sc_in       if( fsm == 0){ fsm = 2;}else exit(2);
sc_out      if( fsm == 0){ fsm = 2;}else exit(3);
bool        if( fsm == 2){ fsm = 3;}
sc_bv       if( fsm == 2){ fsm = 3;}
led[1-5]    if( fsm == 3){printf("%s\n",yytext);fsm = 4;}
btn[1-5]    if( fsm == 3){printf("%s\n",yytext);fsm = 4;}
sevseg[12]  if( fsm == 3){printf("%s\n",yytext);fsm = 4;}
[\_a-zA-Z][\_a-zA-Z0-9]* if( fsm == 1){printf("%s\n",yytext);fsm=0;}else
if(fsm == 2)exit(1); else if(fsm == 3)exit(4);
\,          if( fsm == 4){fsm = 3;}
\;          if( fsm == 4){fsm = 0;}
.
\n
```

Приложение А6. Исходный код vb_timer.h

```
/* vb_timer.h
 * Заголовочный файл для таймера*/

#ifndef VB_TIMER_H
#define VB_TIMER_H

#include <time.h>//В данном заголовочном файле описаны POSIX таймеры
#include <signal.h>//В данном заголовчном файле описаны POSIX сигналы

#define MAX_NUM_TIMERS 256 //Максимальное количесво таймеров
#define INIT_ID 1000 //Начальный id номер таймера

//extern struct sigaction sigact;/*Данная структура нужна чтобы назначить
//                                *для сигналов POSIX функцию
обработчик*/
//extern sigset_t no_msk;//sigset_t представляет собой набор нескольких
сигналов

/* Функция инициализации таймеров */
void use_vb_timers();

/* Класс описывающий таймеры vb_timer */
class vb_timer{
        int timerid;// id номер таймера
        timer_t _timer;//Таймер POSIX
```

```
        unsigned counter = 0;//Количество истечений таймера
        struct sigevent sigevt;/*Стректрура с помощью которой
настаиваются
                                *сигналы POSIX от таймера;*/
        struct itimerspec itimer;/*Структура предназначена для того,
чтобы
                                  *назначить таймеру время истечения*/
        bool is_called = false; /*Если установлено значение true,то
значит
                                 *таймер истек*/
    public:
       friend void vb_timer_handler(int,
                                    siginfo_t *,
                                    void *);/*Функция обработчик сигналов
                                              *имеет доступ частным данным
                                              класса*/
        vb_timer();//Конструктор класса vb_timer
        ~vb_timer();//Деструктор класса vb_timer
        //void init();
        void set(int milisec);/*Метод устанавливает таймеру
                               *интервал в милисекундах*/
        unsigned wait();/*Метод приостанвливает выполнение программы
                         *до истечения таймера. Возвращает количество
                          истечений таймера с предыдущего вызова wait*/
        void stop();/*Метод останавливает таймер*/
};

#endif
```

Приложение А7. Исходный код vb_timer.cpp

```
#include "vb_timer/vb_timer.h"/*В данном заголовочном файле
                               *описан класс vb_vector*/
#include "vb_error/vb_error.h"//В заголовочном файле описана обработка
ошибок
#include <vector> //В данном заголовочном файле описан контейнер vector
#include <unistd.h>/*В данном заголовочном файле описаны функции
                    *стандартные функции UNIX*/

unsigned next_timerid = INIT_ID;/*Переменная хранит значения, которое
                                 *будет присвоено новому экземпляру
vb_timer*/
struct sigaction sigact;;/*Данная структура нужна чтобы назначить
                          *для сигналов POSIX функцию обработчик*/

sigset_t no_msk;;/*Данная переменная нужна чтобы установить
                  *пустую маску для сигналов*/

std::vector<vb_timer *> timers_ref;/*В данном контейнере храянтся
                                    *указатели на созданные таймеры*/

/*Функция обрабатывает сигналы от таймеров*/
void vb_timer_handler(int signo,
                      siginfo_t * siginfo,
                      void * ucontext){
    for(int i = 0; i < timers_ref.size(); i++){
        timers_ref[i]->is_called = false;//Сброс значения
        if (timers_ref[i]->timerid == siginfo->si_int){
```

```
            /*Выполняется если в структуре siginfo, предназначеной для
переди
            *инофрмации через сигналы, хранится значение равное
            *id номеру текущего таймера*/
            timers_ref[i]->counter++;
            timers_ref[i]->is_called = true;
        }
    }
}

/* Функция инициализации таймеров */
void use_vb_timers(){
    int ret;//Переменная в которой хранятся значения стандартных функций
UNIX
    memset(&sigact, 0,sizeof(sigact));//Сброс sigact
    sigact.sa_sigaction = vb_timer_handler;/*Назначить функцию-обработчик
                                           *сигналов*/
    sigact.sa_flags = SA_SIGINFO;/*Использовать структуру siginfo
                                 *при передачи сигналов POSIX*/
    ret = sigemptyset(&no_msk);/*Создать пустую маску
                               *блокировки сигналов*/
    if(ret == -1) vb_error(VB_TIMER);/*Обработка ошибок в фукции
                                       *sigemptyset*/
    sigact.sa_mask = no_msk;/*Назначить маску блокировки сигналов*/
    ret = sigaction( SIGALRM, &sigact, NULL);/*Назначение примененных
                                       *параметров сигналу
SIGALRM*/
    if(ret == -1) vb_error(VB_TIMER);//Аналогично
    timers_ref.reserve(MAX_NUM_TIMERS);/*Установить максимальное
                                      *количество таймеров*/
}


//Конструктор класса таймеров vb_timer
vb_timer::vb_timer(){
    int ret;
    timerid = next_timerid;//Назначение id номера таймеру
    timers_ref.push_back(this);//Добавление таймера в массив
    //Настройка действтий которые будет выполнятся по истечению таймера
    sigevt.sigev_notify = SIGEV_SIGNAL;//Будет посылаться сигнал
    sigevt.sigev_signo = SIGALRM;//Будет использоваться сигнал SIGALRM
    sigevt.sigev_value.sival_int = timerid;//Передаваемое значение для
сигнала
    next_timerid++;//Инкрементация следующего значения id номера
    //Применение значений
    ret = timer_create( CLOCK_MONOTONIC,
                        &sigevt,  &_timer);
    if(ret == -1) vb_error(VB_TIMER);
}

/*Метод устанавливает таймеру интервал в милисекундах*/
void vb_timer::set(int milisec){
    int ret;
    time_t sec = milisec/1000;//Количество целых секунд секунд
    long nsec = (milisec % 1000) * 1000000;//Количество наносекунд
    //Присваивание получившихся значений к структуре itimer
    itimer.it_interval.tv_sec = sec;
```

```
    itimer.it_interval.tv_nsec = nsec;
    itimer.it_value.tv_sec = sec;
    itimer.it_value.tv_nsec = nsec;
    //Установка интервала POSIX таймера
    ret = timer_settime( _timer, 0, &itimer, NULL);
    if(ret == -1) vb_error(VB_TIMER);
}

/*Метод останавливает таймер*/
void vb_timer::stop(){
    int ret;
    /*Присваивание всем полям структуры itimer нулевых значений. Так как
     *При этих значениях таймер останавливается*/
    itimer.it_interval.tv_sec = 0;
    itimer.it_interval.tv_nsec = 0;
    itimer.it_value.tv_sec = 0;
    itimer.it_value.tv_nsec = 0;
    ret = timer_settime( _timer, 0, &itimer, NULL);
    if(ret == -1) vb_error(VB_TIMER);
}

/*Метод приостанвливает выполнение программы до истечения таймера.
 *Возвращает количество истечений таймера с предыдущего вызова wait*/
unsigned vb_timer::wait(){
    int ret;
    while(true){
        pause();//Остановка выполнения программы до любого сигнала
        /* Выйти из цикла, если был сигнал был послан этим таймером */
        if(is_called == true) break;
    }
    is_called = false;//Срос значения
    unsigned ret_coint = counter;
    counter = 0;//Сброс количества истечений
    return ret_coint;
}

//Деструктор класса таймеров vb_timer
vb_timer::~vb_timer(){
    int ret;
    ret = timer_delete(_timer);//Удаление POSIX таймера
    if(ret == -1) vb_error(VB_TIMER);
    std::vector<vb_timer *>::iterator i;
    //Цикл по timers_ref
    for(i = timers_ref.begin();
        i != timers_ref.end();
        i++)
    {
        /*Если id номер таймера равен номеру этого таймера,
         *то удалить его из массива*/
        if((*i)->timerid == timerid){
            timers_ref.erase(i);
            return;
        }
    }
}
```

Приложение А8. Исходный код vb_error.h

```
#ifndef VB_ERROR_H
#define VB_ERROR_H

#include <stdio.h>
#include <errno.h>/*Данный заголовочный файл нужен для обработки
                   *ошибок стандартных функций*/
#include <string>
#include <string.h>
#include <iostream>

//В каком файле произошла ошибка
enum vb_part_id{
    VB_TIMER, //Ошибка в vb_timer.cpp
    VB_IO //Ошибка в vb_io.cpp
};

/* Функция обработки ошибок */
void vb_error( vb_part_id part_id);
```

Приложение А9. Исходный код vb_error.cpp

```
#include "vb_error/vb_error.h"

//Строковые константы для каждого файла
const std::string msg_vb_none = "error: ";
const std::string msg_vb_timer = "error in vb_timer: ";
const std::string msg_vb_io = "error in vb_io: ";

//Буфер для чтения ошибок стандартных функций
char err_buf[128];

/* Функция обработки ощибок */
void vb_error( vb_part_id part_id){
    std::string where;
    std::string error_msg;
    switch(part_id){
        case VB_TIMER:
            where = msg_vb_timer;
            break;
        case VB_IO:
            where = msg_vb_io;
            break;
    }
    error_msg = strerror_r(errno,err_buf,128);
    error_msg = err_buf;
    std::cerr << where << error_msg << std::endl;
    exit(-1);
}
```

Приложение А10. Исходный код mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QProcess>
#include <QIODevice>
#include <QTimer>
```

```
#include <windopenauto.h>
#include "vb_io/vb_io.h"
#include "vbled.h"
#include "vbbtn.h"
#include "sevsegitm.h"
#include <QGraphicsScene>
#include "settings_wind.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    QList<QByteArray> ListStr;
    vec_cmds cDataIn;
    vec_cmds cCtlIn;
    vec_cmds cDataOut;
    vec_cmds cCtlOut;
    QByteArray InPipe;
    QByteArray OutPipe;
    QProcess vbproc;
    QString namevb;
    Ui::MainWindow *ui;
    WindOpenAuto * _windOpenAuto;
    settings_wind * _settingsWind;
    vbled * led1,* led2,* led3,* led4,* led5;
    vbbtn * btn1,* btn2,* btn3,* btn4,* btn5;
    QLabel lal1, lal2, lal3, lal4, lal5;
    QLabel lab1, lab2, lab3, lab4, lab5;
    SevSegItm * sevseg1;
    SevSegItm * sevseg2;
private slots:
    void updateSettings();
    void OpenModuleAuto(bool);
    void OpenSettings();
    void recieveData();
    void handlInCmd();
    void handlOutCmd();
    void handlInData();
    void handlOutData();
    void startModel(QString);
    void btn1_clicked();
    void btn2_clicked();
    void btn3_clicked();
    void btn4_clicked();
    void btn5_clicked();
    void on_start_btn_clicked();
    void on_stop_btn_clicked();
};
```

```
#endif // MAINWINDOW_H
```

Приложение А11. Исходный код mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QFile>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    QProcessEnvironment sc_mult_drivers =
QProcessEnvironment::systemEnvironment();
    sc_mult_drivers.insert("SC_SIGNAL_WRITE_CHECK","DISABLE");
    vbproc.setProcessEnvironment(sc_mult_drivers);
    _windOpenAuto = new WindOpenAuto(this);
    _settingsWind = new settings_wind(this);
    led1 = new vbled(this); led2 = new vbled(this);
    led3 = new vbled(this); led4 = new vbled(this);
    led5 = new vbled(this);
    btn1 = new vbbtn(this); btn2 = new vbbtn(this);
    btn3 = new vbbtn(this); btn4 = new vbbtn(this);
    btn5 = new vbbtn(this);
    ui->setupUi(this);
    sevseg1 = new SevSegItm();
    ui->sevseg1_view->setScene(sevseg1);
    sevseg2 = new SevSegItm();
    ui->sevseg2_view->setScene(sevseg2);
    lal1.setText("led1");
    ui->ll1->addWidget(led1);
    ui->ll1->addWidget(&lal1);
    lal2.setText("led2");
    ui->ll2->addWidget(led2);
    ui->ll2->addWidget(&lal2);
    lal3.setText("led3");
    ui->ll3->addWidget(led3);
    ui->ll3->addWidget(&lal3);
    lal4.setText("led4");
    ui->ll4->addWidget(led4);
    ui->ll4->addWidget(&lal4);
    lal5.setText("led5");
    ui->ll5->addWidget(led5);
    ui->ll5->addWidget(&lal5);
    lab1.setText("btn1");
    ui->lb1->addWidget(btn1);
    ui->lb1->addWidget(&lab1);
    lab2.setText("btn2");
    ui->lb2->addWidget(btn2);
    ui->lb2->addWidget(&lab2);
    lab3.setText("btn3");
    ui->lb3->addWidget(btn3);
    ui->lb3->addWidget(&lab3);
    lab4.setText("btn4");
    ui->lb4->addWidget(btn4);
    ui->lb4->addWidget(&lab4);
```

```
    lab5.setText("btn5");
    ui->lb5->addWidget(btn5);
    ui->lb5->addWidget(&lab5);
    connect(ui->actModAuto, &QAction::triggered,
            this, &MainWindow::OpenModuleAuto);
    connect(_windOpenAuto, &WindOpenAuto::vbprocCompilied,
            this,           &MainWindow::startModel);
    connect(ui->actSettings, &QAction::triggered,
            this,            &MainWindow::OpenSettings);
    connect(_settingsWind, &settings_wind::settingChange,
            this,          &MainWindow::updateSettings);
    updateSettings();
    connect(btn1, &vbbtn::clicked,
            this, &MainWindow::btn1_clicked);
    connect(btn2, &vbbtn::clicked,
            this, &MainWindow::btn2_clicked);
    connect(btn3, &vbbtn::clicked,
            this, &MainWindow::btn3_clicked);
    connect(btn4, &vbbtn::clicked,
            this, &MainWindow::btn4_clicked);
    connect(btn5, &vbbtn::clicked,
            this, &MainWindow::btn5_clicked);
    connect(&vbproc,&QProcess::readyReadStandardOutput,
            this,   &MainWindow::recieveData);
}

MainWindow::~MainWindow()
{
    //_windOpenAuto->hide();
    //delete _windOpenAuto;
    delete led1; delete led2; delete led3;
    delete led4; delete led5;
    delete btn1; delete btn2; delete btn3;
    delete btn4; delete btn5;
    delete sevseg1; delete sevseg2;
    disconnect(&vbproc,&QProcess::readyReadStandardOutput,
               this,   &MainWindow::recieveData);
    vbproc.kill();
    delFile(namevb);
    delete ui;
}

void MainWindow::recieveData(){
    OutPipe = vbproc.readAllStandardOutput();
    str2vec_cmds(&cCtlOut,&cDataOut,OutPipe.data());
    handlOutCmd();
    handlOutData();
    cCtlOut.clear();
    cDataOut.clear();
}

void MainWindow::handlOutCmd(){
    vec_cmds::iterator i;
    QString text;
    for (i = cCtlOut.begin();
         i != cCtlOut.end();
         i++){
```

```
        if ((*i).name.compare("time")==0)

ui->time_label->setText(text.sprintf("%.2f",(*i).value/1000.0));
    }
}

void MainWindow::handlOutData(){
    vec_cmds::iterator i;
    for( i = cDataOut.begin();
         i != cDataOut.end();
         i++){
        if((*i).name.compare("led1")==0){
            if((*i).value == 0) led1->setState(false);
            else led1->setState(true);
        }else if((*i).name.compare("led2") == 0){
            if((*i).value == 0) led2->setState(false);
            else led2->setState(true);
        }else if((*i).name.compare("led3") == 0){
            if((*i).value == 0) led3->setState(false);
            else led3->setState(true);
        }else if((*i).name.compare("led4") == 0){
            if((*i).value == 0) led4->setState(false);
            else led4->setState(true);
        }else if((*i).name.compare("led5") == 0){
            if((*i).value == 0) led5->setState(false);
            else led5->setState(true);
        }else if((*i).name.compare("sevseg1") == 0){
            sevseg1->setState((*i).value);
        }else if((*i).name.compare("sevseg2") == 0){
            sevseg2->setState((*i).value);
        }
    }
}

void MainWindow::handlInCmd(){
    InPipe.clear();
    vec_cmds::iterator i;
    for( i = cCtlIn.begin();
         i != cCtlIn.end();
         i++){
        InPipe.append("#" + QString((*i).name.c_str()) + ":" +
                      QString::number((*i).value) +
                      "\n");
    }
    qDebug() << InPipe;
    cCtlIn.clear();
    vbproc.write(InPipe);
}

void MainWindow::handlInData(){
    InPipe.clear();
    vec_cmds::iterator i;
    for( i = cDataIn.begin();
         i != cDataIn.end();
         i++){
        InPipe.append("-" + QString((*i).name.c_str()) + ":" +
                      QString::number((*i).value) + "\n");
```

```
    }
    qDebug() << InPipe;
    cDataIn.clear();
    vbproc.write(InPipe);
}

void MainWindow::OpenModuleAuto(bool i){
    _windOpenAuto->show();
}

void MainWindow::startModel(QString nm){
    if(vbproc.state()!=QProcess::NotRunning){
        vbproc.kill();
        vbproc.waitForFinished(-1);
    }
    namevb = nm;
    vbproc.start(namevb + " 50");
    qDebug() << vbproc.state();
    qDebug() << namevb;
}

void MainWindow::btn1_clicked(){
    cmd tmp;
    tmp.name = "btn1";
    if(btn1->isChecked()){
        tmp.value = 1;
    }else{
        tmp.value = 0;
    }
    cDataIn.push_back(tmp);
    handlInData();
}

void MainWindow::btn2_clicked(){
    cmd tmp;
    tmp.name = "btn2";
    if(btn2->isChecked()){
        tmp.value = 1;
    }else{
        tmp.value = 0;
    }
    cDataIn.push_back(tmp);
    handlInData();
}

void MainWindow::btn3_clicked(){
    cmd tmp;
    tmp.name = "btn3";
    if(btn3->isChecked()){
        tmp.value = 1;
    }else{
        tmp.value = 0;
    }
    cDataIn.push_back(tmp);
    handlInData();
}
```

```
void MainWindow::btn4_clicked(){
    cmd tmp;
    tmp.name = "btn4";
    if(btn4->isChecked()){
        tmp.value = 1;
    }else{
        tmp.value = 0;
    }
    cDataIn.push_back(tmp);
    handlInData();
}

void MainWindow::btn5_clicked(){
    cmd tmp;
    tmp.name = "btn5";
    if(btn5->isChecked()){
        tmp.value = 1;
    }else{
        tmp.value = 0;
    }
    cDataIn.push_back(tmp);
    handlInData();
}

void MainWindow::on_start_btn_clicked()
{
    cmd tmp;
    tmp.name = "start";
    tmp.value = 0;
    cCtlIn.push_back(tmp);
    handlInCmd();
}

void MainWindow::on_stop_btn_clicked()
{
    cmd tmp;
    tmp.name = "stop";
    tmp.value = 0;
    cCtlIn.push_back(tmp);
    handlInCmd();
}

void MainWindow::OpenSettings(){
    _settingsWind->show();
}

void MainWindow::updateSettings(){
    _windOpenAuto->setSystemCPath(_settingsWind->getSystemCPath());
    _windOpenAuto->setLexPath(_settingsWind->getLexPath());
    _windOpenAuto->setGccPath(_settingsWind->getGccPath());
    _windOpenAuto->setVirtBoardPath(_settingsWind->getVirtBoardPath());
    _windOpenAuto->setInclPathVB(_settingsWind->getInclPathVB());
    _windOpenAuto->setInclPathSC(_settingsWind->getInclPathSC());
    _windOpenAuto->updatePathName();
}
```

Приложение А12. Исходный код settings_wind.h

```
#ifndef SETTINGS_WIND_H
#define SETTINGS_WIND_H

#include <QDialog>
#include <QFile>

namespace Ui {
class settings_wind;
}

class settings_wind : public QDialog
{
    Q_OBJECT

public:
    explicit settings_wind(QWidget *parent = 0);
    ~settings_wind();
    QFile configFile;
    QString getSystemCPath();
    QString getLexPath();
    QString getGccPath();
    QString getVirtBoardPath();
    QString getInclPathVB();
    QString getInclPathSC();
private slots:
    void on_OkBtn_clicked();
    void readConf();
    void writeConf();
    void on_cancelBtn_clicked();
private:
      QString SystemCPath;//Имя каталога в котором хранится библиотека
SystemC
    QString LexPath;//Путь к лексическому анализатору
    QString GccPath;//Путь к компилятору
      QString VirtBoardPath;//Путь к библиотеке виртуальной отладочной
платы
      QString inclVirtBoard;//Заголовочные файлы виртуальной отладочной
платы
    QString inclSystemC;//Заголовочные файлы библиотеки SystemC
    Ui::settings_wind *ui;
signals:
    void settingChange();
};

#endif // SETTINGS_WIND_H
```

Приложение А13. Исходный код settings_wind.cpp

```
#include "settings_wind.h"
#include "ui_settings_wind.h"
#include <QDebug>

settings_wind::settings_wind(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::settings_wind),
    configFile(".config")
```

```
{
    ui->setupUi(this);
    readConf();
    SystemCPath = ui->SystemCPath->text();
    LexPath = ui->LexPath->text();
    GccPath = ui->GccPath->text();
    VirtBoardPath = ui->VirtBoardPath->text();
    inclVirtBoard = ui->inclVirtBoard->text();
}

settings_wind::~settings_wind()
{
    delete ui;
}

void settings_wind::on_OkBtn_clicked()
{
    writeConf();
    SystemCPath = ui->SystemCPath->text();
    LexPath = ui->LexPath->text();
    GccPath = ui->GccPath->text();
    VirtBoardPath = ui->VirtBoardPath->text();
    inclVirtBoard = ui->inclVirtBoard->text();
    inclSystemC = ui->inclSystemC->text();
    emit settingChange();
    hide();
}

void settings_wind::readConf(){
    QString line;
    QStringList data;
    if(configFile.open(QIODevice::ReadOnly | QIODevice::Text)){
        while (!configFile.atEnd()) {
            line = configFile.readLine();
            line = line.left(line.size()-1);
            data = line.split(':');
            if(data.size() == 2){
                if(data[0].compare("SystemCPath") == 0)
                    ui->SystemCPath->setText(data[1]);
                else if(data[0].compare("VirtBoardPath") == 0)
                    ui->VirtBoardPath->setText(data[1]);
                else if(data[0].compare("inclVirtBoard") == 0)
                    ui->inclVirtBoard->setText(data[1]);
                else if(data[0].compare("inclSystemC") == 0)
                    ui->inclSystemC->setText(data[1]);
                else if(data[0].compare("LexPath") == 0)
                    ui->LexPath->setText(data[1]);
                else if(data[0].compare("GccPath") == 0)
                    ui->GccPath->setText(data[1]);
            }
        }
    }
    configFile.close();
}

void settings_wind::writeConf(){
    if(configFile.open(QIODevice::WriteOnly | QIODevice::Text)){
```

```
        configFile.write("SystemCPath:" + ui->SystemCPath-
>text().toUtf8() + "\n");
        configFile.write("VirtBoardPath:" + ui->VirtBoardPath-
>text().toUtf8() + "\n");
        configFile.write("inclVirtBoard:" + ui->inclVirtBoard-
>text().toUtf8() + "\n");
        configFile.write("inclSystemC:" + ui->inclSystemC-
>text().toUtf8() + "\n");
        configFile.write("LexPath:" + ui->LexPath->text().toUtf8() + "\
n");
        configFile.write("GccPath:" + ui->GccPath->text().toUtf8() + "\
n");
    }
    configFile.close();
}

void settings_wind::on_cancelBtn_clicked()
{
    hide();
}

QString settings_wind::getGccPath(){
    return GccPath;
}

QString settings_wind::getLexPath(){
    return LexPath;
}

QString settings_wind::getSystemCPath(){
    return SystemCPath;
}

QString settings_wind::getVirtBoardPath(){
    return VirtBoardPath;
}

QString settings_wind::getInclPathVB(){
    return inclVirtBoard;
}

QString settings_wind::getInclPathSC(){
    return inclSystemC;
}
```

Приложение А14. Исходный код windopenauto.h

```
#ifndef WINDOPENAUTO_H
#define WINDOPENAUTO_H

#include <QMainWindow>
#include <QFileDialog>
#include <QDebug>
#include <QProcess>
#include <QFileInfo>
#include <QFile>
#include <QMessageBox>
```

```
void delFile(QString);//Удалить файл

namespace Ui {
class WindOpenAuto;
}

class WindOpenAuto : public QMainWindow
{
    Q_OBJECT

public:
    explicit WindOpenAuto(QWidget *parent = 0);
    ~WindOpenAuto();
    void setSystemCPath(QString);
    void setLexPath(QString);
    void setGccPath(QString);
    void setVirtBoardPath(QString);
    void setInclPathVB(QString);
    void setInclPathSC(QString);
    void updatePathName();
private slots:

    void on_CompileButton_clicked();
    void on_OpenH_clicked();
    void on_OpenCPP_clicked();
    void on_Cancel_clicked();

private:
    QString getTempNameF(QString,QString);//Получить имя временного файла
    QString getPathFile(QString,QString);//Получить полный путь к файлу
    bool compileModule();//Откомпилироваить модуль
    bool genSc_main();//Сгенерировать sc_main файл
    bool AnalyzeModule();//Анализировать код модуля
    bool compileMain();//Откомпилировать sc_main
    bool compileModel();//Откомпилировать программу моделирования
    QStringList RecognPorts();//Распознать порты модуля
    QByteArray err_out();
    QStringList vblex_data;
    QString nameGcc;//Имя компилятора
    QString nameVBLex;//Имя лексического анализатора
    QString nameNoduleCpp;//Путь к cpp файлу модуля
    QString nameModuleH;//Путь к h файлу модуля
    QString nameModel;//Имя процесса моделирования
    QString nameObjMod;//Имя объектного файла модуля
    QString nameModule;//Имя модуля
    QString nameErr_file;
    QString nameSc_main;
    QString nameObjMain;
    Ui::WindOpenAuto *ui;
    //Внешние переменные:
      QString SystemCPath;//Имя каталога в котором хранится библиотека
SystemC
    QString LexPath;//Путь к лексическому анализатору
    QString GccPath;//Путь к компилятору
      QString VirtBoardPath;//Путь к библиотеке виртуальной отладочной
платы
```

```
      QString inclVirtBoard;//Заголовочные файлы виртуальной отладочной
платы
    QString inclSystemC;//Заголовочные файлы библиотеки SystemC
signals:
    void vbprocCompilied(QString);
};

#endif // WINDOPENAUTO_H
```

Приложение А15. Исходный код windopenauto.cpp

```
#include "windopenauto.h"
#include "ui_windopenauto.h"

WindOpenAuto::WindOpenAuto(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::WindOpenAuto)
{
    ui->setupUi(this);
}

WindOpenAuto::~WindOpenAuto(){
    delFile(nameModel);
    delFile(nameObjMain);
    delFile(nameObjMod);
    delFile(nameSc_main);
    delete ui;
}

QByteArray WindOpenAuto::err_out(){
    QByteArray tmp;
    QFile errFile(QDir::tempPath() + "/errors_compile");
    if(errFile.open(QIODevice::ReadOnly | QIODevice::Text)){
        tmp = errFile.readAll();
    }
    return tmp;
}

void WindOpenAuto::updatePathName(){
    nameGcc = getPathFile(GccPath,"g++");//Имя компилятора
         nameVBLex  =  getPathFile(LexPath,"vbllex");//Имя  лексического
анализатора
    nameModel = getTempNameF(QDir::tempPath() +
                             "/" + "vb_proc",
                                "");  //Генерирование имени процесса
моделирования
    nameObjMod = getTempNameF(QDir::tempPath() +
                             "/" + "module",
                                 ".o");//Генерирование имени объектного
файла объектного кода модуля
    nameObjMain = getTempNameF(QDir::tempPath() +
                              "/" + "main",
                              ".o");
    nameSc_main = getTempNameF(QDir::tempPath() +
                              "/" + "main",
                              ".cpp");
    nameObjMain = getTempNameF(QDir::tempPath() +
```

```
                                "/" + "main",
                                ".o");
    nameSc_main = getTempNameF(QDir::tempPath() +
                                "/" + "main",
                                ".cpp");
    nameErr_file = QDir::tempPath() + "/errors_compile";
}

QString WindOpenAuto::getPathFile(QString path, QString name){//Функция
для получения пути к файлу
    QString PathFile;
    if(path.isEmpty()) PathFile = name;
    else PathFile = path + "/" + name;
    return PathFile;
}

QString WindOpenAuto::getTempNameF(QString str1, QString str2){//Функция
получения имени временного файла
    QString tempname = str1 + str2;
    if(QFileInfo::exists(tempname)){
        unsigned long i = 0;
        do{
            tempname = str1 + QString::number(i) + str2;
            i++;
        }while (QFileInfo::exists(tempname));
    }
    return tempname;
}

void delFile(QString nameObj){//Функция удаления файла с именем nameObj
    if(!nameObj.isEmpty()){
        QFile delfile(nameObj);
        delfile.remove();
    }
}

void WindOpenAuto::on_CompileButton_clicked(){
    ui->textBrowser->clear();
    if(nameNoduleCpp.isEmpty() || nameModuleH.isEmpty()){
        QMessageBox::warning(this,"Ошибка компиляции",
                             "Выбраны не все файлы",QMessageBox::Ok);
        return;
    }
    if(!compileModule()){
        QMessageBox::warning(this,"Ошибка компиляции",
                             "Ошибка при компиляции файла модуля\n" +
                             nameNoduleCpp,QMessageBox::Ok);
        ui->textBrowser->setText(err_out());
        delFile(nameObjMod);
        return;
    }
    if(!genSc_main()){
        QMessageBox::warning(this,"Ошибка генерации",
                             "Ошибка при генерации файла main\n" +
                             nameSc_main,QMessageBox::Ok);
        delFile(nameObjMod);
        return;
```

```
    }
    if(!compileMain()){
        QMessageBox::warning(this,"Ошибка компиляции",
                             "Ошибка при компиляции файла main\n" +
                             nameSc_main,QMessageBox::Ok);
        ui->textBrowser->setText(err_out());
        delFile(nameObjMod);
        delFile(nameObjMain);
        return;
    }else{
        delFile(nameSc_main);
    }
    if(!compileModel()){
        QMessageBox::warning(this,"Ошибка компиляции",
                                    "Ошибка при компиляции программы для
моделирования"
                             ,QMessageBox::Ok);
        ui->textBrowser->setText(err_out());
        delFile(nameObjMain);
        delFile(nameObjMod);
        return;
    }
    delFile(nameObjMain);
    delFile(nameObjMod);
    delFile(nameErr_file);
    emit vbprocCompilied(nameModel);
    hide();
}

bool WindOpenAuto::compileModule(){
    QProcess CompModule;//Процесс компиляциии модуля
    qDebug() << nameGcc;
    QStringList args;
    args << "-g" << "-c";
    if(!(inclVirtBoard.isEmpty()))
        args << ("-I" + inclVirtBoard);
    if(!(inclSystemC.isEmpty()))
        args << ("-I" + inclSystemC);
    args << nameNoduleCpp << "-o" << nameObjMod;
    qDebug() << args;
    CompModule.setStandardErrorFile(nameErr_file);
    CompModule.start(nameGcc,args);
    CompModule.waitForFinished(-1);
    if(CompModule.exitCode()!=0){
        qDebug() << "Error compile:" << nameObjMod;
        return false;
    }
    return true;
}

bool WindOpenAuto::AnalyzeModule(){
    QProcess AnalyzeLex;
    AnalyzeLex.setStandardInputFile(nameModuleH);
    qDebug() << nameModuleH;
    QString outFileName = nameObjMod + ".txt";
    AnalyzeLex.setStandardOutputFile(outFileName);
    qDebug() << nameVBLex;
```

```
    AnalyzeLex.start(nameVBLex);
    AnalyzeLex.waitForFinished(-1);
    if(AnalyzeLex.exitCode()!=0){
        qDebug() << "Error analyze:" << nameModuleH;
        delFile(outFileName);
        return false;
    };
    QFile outFile(outFileName);
    outFile.open(QIODevice::ReadOnly | QIODevice::Text);
    QString buf = outFile.readAll();
    qDebug() << buf;
    outFile.close();
    delFile(outFileName);
    vblex_data = buf.split('\n');
    return true;
}

void WindOpenAuto::on_OpenH_clicked(){
    nameModuleH = QFileDialog::getOpenFileName(this,"Open .cpp file",
                                                QDir::homePath(),
                                                     "Module (*.h);;All
(*.*)");
    ui->nameLabelH->setText(nameModuleH);
}

void WindOpenAuto::on_OpenCPP_clicked(){
    nameNoduleCpp = QFileDialog::getOpenFileName(this,"Open .h file",
                                                QDir::homePath(),
                                                    "Module (*.cpp);;All
(*.*)");
    ui->nameLabelCPP->setText(nameNoduleCpp);
}

QStringList WindOpenAuto::RecognPorts(){
    QStringList::iterator curWord;
    curWord = vblex_data.begin();
    nameModule = *curWord;
    curWord++;
    QStringList rec_port;
    for(;curWord != vblex_data.end();
         curWord++){
        if(!(*curWord).isEmpty())
            rec_port.append(*curWord);
    }
    return rec_port;
}

bool WindOpenAuto::genSc_main(){
    if(!AnalyzeModule())return false;
    QStringList rec_ports = RecognPorts();
    QString _BEGIN =
        "#include <systemc.h>\n\
         #include \"vb_testbench/vb_testbench.h\"\n\
         #include \"vb_timer/vb_timer.h\"\n\
         #include \"" + nameModuleH + "\"\n\
         int sc_main(int argc, char ** argv){\
            long timestep;\
```

```
            if(argc != 1) timestep = atoi(argv[1]);\
            else timestep = 10000;\
            use_vb_timers();\
            sc_signal<bool> led1,led2,led3,led4,led5;\
            sc_signal<bool> btn1,btn2,btn3,btn4,btn5;\
            sc_signal<sc_bv<7>> sevseg1,sevseg2;\
            vb_testbench vb(\"TestB\",timestep);\
            vb.led1(led1); vb.led2(led2);\
            vb.led3(led3);vb.led4(led4);\
            vb.led5(led5);\
            vb.btn1(btn1); vb.btn2(btn2);\
            vb.btn3(btn3); vb.btn4(btn4);\
            vb.btn5(btn5);\
            vb.sevseg1(sevseg1);\
            vb.sevseg2(sevseg2);";
    QString _MIDDLE = nameModule + " modlule(\"MODULE\");";
    QStringList::iterator curPort;
    for(curPort = rec_ports.begin();
        curPort != rec_ports.end();
        curPort++){
        _MIDDLE += ("modlule." + (*curPort) + "(" + (*curPort) + ");\n");
    }
    QString _END = "sc_start();return 0;}";
    QFile mainFile(nameSc_main);
    if(mainFile.open(QIODevice::WriteOnly | QIODevice::Text)){
        mainFile.write(_BEGIN.toUtf8());
        mainFile.write(_MIDDLE.toUtf8());
        mainFile.write(_END.toUtf8());
        qDebug() << _MIDDLE;
    }
    mainFile.close();
    return true;
}

bool WindOpenAuto::compileMain(){
    QProcess CompMain;
    QStringList args;
    args << "-g" << "-c";
    if(!(inclVirtBoard.isEmpty()))
        args << ("-I" + inclVirtBoard);
    if(!(inclSystemC.isEmpty()))
        args << ("-I" + inclSystemC);
    args << nameSc_main << "-o" << nameObjMain;
    qDebug() << args;
    CompMain.setStandardErrorFile(nameErr_file);
    CompMain.start(nameGcc,args);
    CompMain.waitForFinished(-1);
    if(CompMain.exitCode()!=0){
        qDebug() << "Error compile:" << nameObjMain;
        return false;
    }
    return true;
}

bool WindOpenAuto::compileModel(){
    QProcess CompModel;
    QStringList args;
```

```
    args << "-g";
    if(!(inclVirtBoard.isEmpty()))
        args << ("-L" + VirtBoardPath);
    if(!(inclSystemC.isEmpty()))
        args << ("-L" + SystemCPath);
    args << nameObjMod << nameObjMain;
    args << "-lvb_testbench" << "-lsystemc" << "-lrt";
    args << "-o" << nameModel;
    qDebug() << args;
    CompModel.setStandardErrorFile(nameErr_file);
    CompModel.start(nameGcc,args);
    CompModel.waitForFinished(-1);
    if(CompModel.exitCode()!=0){
        qDebug() << "Error Compile";
        return false;
    }
    return true;
}

void WindOpenAuto::on_Cancel_clicked()
{
    hide();
}

void WindOpenAuto::setGccPath(QString str){
    GccPath = str;
}

void WindOpenAuto::setLexPath(QString str){
    LexPath = str;
}

void WindOpenAuto::setInclPathSC(QString str){
    inclSystemC = str;
}

void WindOpenAuto::setInclPathVB(QString str){
    inclVirtBoard = str;
}

void WindOpenAuto::setSystemCPath(QString str){
    SystemCPath = str;
}

void WindOpenAuto::setVirtBoardPath(QString str){
    VirtBoardPath = str;
}
```

Приложение А16. Исходный код vbbtn.h

```
#ifndef VBBTN_H
#define VBBTN_H

#include <QObject>
#include <QWidget>
#include <QAbstractButton>
#include <QPainter>
```

```
class vbbtn : public QAbstractButton
{
    Q_OBJECT
    QPixmap imgon, imgoff;
    void paintEvent(QPaintEvent *e);
public:
    explicit vbbtn(QWidget * parent = 0);
};

#endif // VBBTN_H
```

Приложение А17. Исходный код vbbtn.cpp

```
    #include "vbbtn.h"

    vbbtn::vbbtn(QWidget * parent) : QAbstractButton(parent),
        imgon("vbbtnon.png"), imgoff("vbbtnoff.png")
    {
        setCheckable(true);
        setChecked(false);
        setMinimumSize(QSize(40,40));
    }

    void vbbtn::paintEvent(QPaintEvent *e){
        Q_UNUSED(e);
        QPainter painter(this);
        if(isChecked()==false){
            painter.drawPixmap(0,0,40,40,imgoff);
        }else{
            painter.drawPixmap(0,0,40,40,imgon);
        }
    }
```

Приложение А18. Исходный код vbled.h

```
#ifndef VBLED_H
#define VBLED_H

#include <QObject>
#include <QWidget>
#include <QLabel>
#include <QPixmap>

class vbled : public QLabel
{
    Q_OBJECT
    QPixmap imgon, imgoff;
    bool state;
public:
    explicit vbled(QWidget * parent = 0);
    void setState(bool);
};

#endif // VBLED_H
```

Приложение А19. Исходный код vbled.cpp

```
#include "vbled.h"

vbled::vbled(QWidget * parent) : QLabel(parent),
    imgoff("vbledoff.png"), imgon("vbledon.png")
```

```
{
    state = false;
    setPixmap(imgoff);
}

void vbled::setState(bool flg){
    state = flg;
    if(state)
        setPixmap(imgon);
    else
        setPixmap(imgoff);
}
```

Приложение А20. Исходный код sevsegitm.h

```
#ifndef SEVSEGITM_H
#define SEVSEGITM_H

#include <QObject>
#include <QGraphicsItem>
#include <QGraphicsRectItem>
#include <QGraphicsScene>

class SevSegItm : public QGraphicsScene
{
    Q_OBJECT
    QGraphicsRectItem A,B,C,D,E,F,G;
    QBrush Bon, Boff;
    int state;
public:
    explicit SevSegItm(QObject * parent = 0);
    void setState(int);
};

#endif // SEVSEGITM_H
```

Приложение А21. Исходный код sevsegitm.cpp

```
#include "sevsegitm.h"

SevSegItm::SevSegItm(QObject * parent)
        : QGraphicsScene(parent)
{
    Boff.setStyle(Qt::SolidPattern);
    Bon.setStyle(Qt::SolidPattern);
    Boff.setColor(QColor(100,0,0));
    Bon.setColor(QColor(255,0,0));
    A.setRect(15,0,35,3);
    B.setRect(50,3,3,35);
    C.setRect(50,41,3,35);
    D.setRect(15,76,35,3);
    E.setRect(12,41,3,35);
    F.setRect(12,3,3,35);
    G.setRect(15,38,35,3);
    A.setBrush(Boff);
    B.setBrush(Boff);
    C.setBrush(Boff);
    D.setBrush(Boff);
    E.setBrush(Boff);
```

```
    F.setBrush(Boff);
    G.setBrush(Boff);
    addItem(&A);
    addItem(&B);
    addItem(&C);
    addItem(&D);
    addItem(&E);
    addItem(&F);
    addItem(&G);
}

void SevSegItm::setState(int val){
    state = val;
    if((state &(1<<0))!=0)
        A.setBrush(Bon);
    else
        A.setBrush(Boff);
    if((state &(1<<1))!=0)
        B.setBrush(Bon);
    else
        B.setBrush(Boff);
    if((state &(1<<2))!=0)
        C.setBrush(Bon);
    else
        C.setBrush(Boff);
    if((state &(1<<3))!=0)
        D.setBrush(Bon);
    else
        D.setBrush(Boff);
    if((state &(1<<4))!=0)
        E.setBrush(Bon);
    else
        E.setBrush(Boff);
    if((state &(1<<5))!=0)
        F.setBrush(Bon);
    else
        F.setBrush(Boff);
    if((state &(1<<6))!=0)
        G.setBrush(Bon);
    else
        G.setBrush(Boff);
}
```