Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

М. И. Дмитриченко, А. А. Шалыто

Имитатор микроволновой печи

Объектно-ориентированное программирование
с явным выделением состояний

Проектная документация

Проект создан в рамках

«Движения за открытую проектную документацию»

http://is.ifmo.ru

Санкт-Петербург

2004

Содержание

Введение

Для алгоритмизации и программирования задач логического управления применяются различные подходы и технологии. В данной работе для задачи такого рода была использована SWITCH-технология, предложенная А. А. Шалыто [1] и развитая им совместно с Н. И. Туккелем [2]. Более подробно ознакомиться с этой технологией можно на сайтах http://is.ifmo.ru.

В данной работе совместно применяются объектно-ориентированное и автоматное программирование, что названо "*объектно-ориентированным программированием с явным выделением состояний*" [2].

Целью работы является разработка имитатора микроволновой печи на основе SWITCH-технологии.

Программа написана на языке программирования *C++* [3] и разработана с помощью среды *Microsoft Visual C++ 6.0* с использованием библиотеки классов *MFC*.

1. Постановка задачи

Как отмечалось во введении, целью данной работы является создание имитатора микроволновой печи (далее – печь). Имитатор должен удовлетворять следующим требованиям.

1. Работа печи происходит лишь при закрытой двери.
2. Управление печью происходит с использованием приборной панели (рис.1), на которой размещены:
   - экран для отображения информации;
   - клавиша режима авторазморозки;
   - клавиша для выбора массы продуктов, подвергающихся авторазморозке;
   - три клавиши для выбора длительности работы и установки часов;
   - клавиша установки часов;
   - клавиша выбора мощности работы;
   - клавиша отмены действий;
   - клавиша начала работы или добавления 30 с к длительности выполнения текущего задания.

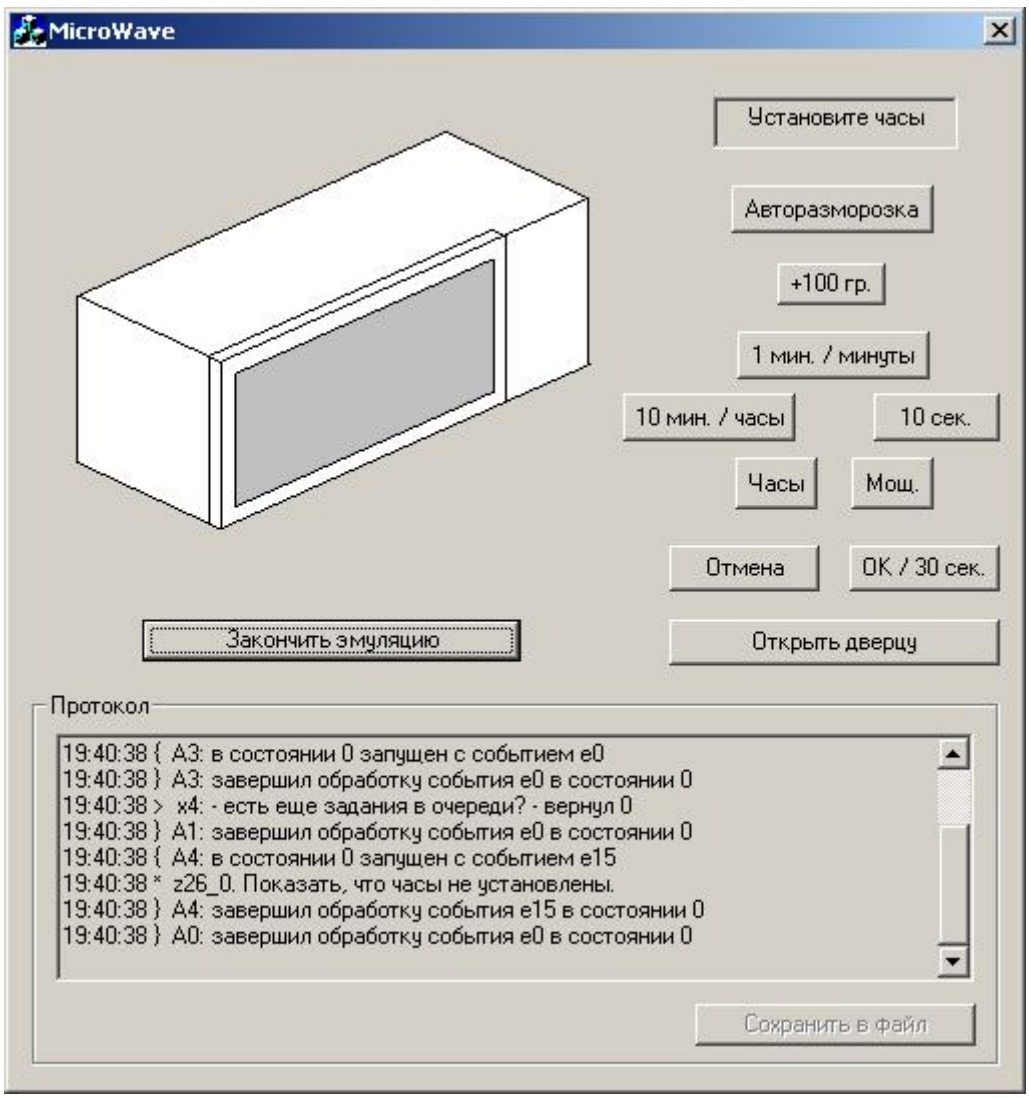

Рис.1. Внешний вид имитатора микроволновой печи

3. Имеется кнопка открытия двери.
4. Установка часов производится нажатием клавиши **«Часы»**. После этого печь переходит в состояние ”Установка часов”. Последующие нажатия этой клавиши приводят к переключению режима часов между 24-часовым и 12-часовым форматом. Нажатие одной из клавиш **«10 мин./часы»** и **«1 мин./минуты»** начинают стадию выбора текущего времени. Добавление одного часа к текущему времени производится нажатием клавиши **«10 мин./часы»**, а одной минуты – нажатием клавиши **«1 мин./минуты»**. Нажатие клавиши **«Часы»** завершает установку часов и делает выбранное время текущим. В любой момент времени нажатие клавиши **«Отмена»** приводит к возврату в первоначальное состояние и отмене все выбранных установок времени.
5. Программирование режима работы начинается с установки мощности нажатием клавиши **«Мощ.»**. Повторным нажатием этой клавиши пользователь может выбрать один из приведенных ниже уровней мощности: 750 Вт; 600 Вт; 450 Вт; 300 Вт; 180 Вт и 100 Вт.

   После выбора нужного варианта спустя две секунды или по нажатию одной из клавиш выбора длительности работы пользователю будет предложено задать время работы. Нажатие клавиши **«10 мин./часы»** добавляет к выбранному времени 10 мин., клавиши **«1 мин./минуты» -** 1 мин., а клавиши **«10 сек.» -** 10 с. В любой момент времени нажатие клавиши **«Отмена»** приводит к возврату в первоначальное состояние и отмене выбранных условий приготовления пищи. Нажатие клавиши **«ОК/30 сек.»** приведет к началу работы печи (но только в случае, если закрыта дверца).
6. Для размораживания пищи можно использовать режим «авторазморозки». В этом случае по заданной массе пищи печь сама рассчитает необходимое время работы на минимальной мощности (100 Вт). Для использования этого режима необходимо нажать клавишу **«Авторазморозка».** После этого пользователю будет предложено установить массу размораживаемых продуктов. Установка массы происходит повторным нажатием клавиши **«+100 гр.»** в интервале от 0 до 1500 гр. После выбора нужной массы нажатие клавиши **«ОК/30 сек.»** приведет к началу размораживания. В любой момент времени нажатие клавиши **«Отмена»** приводит к возврату в первоначальное состояние и отмене выбранных условий размораживания пищи.
7. Печь может быть запрограммирована для приготовления пиши в несколько этапов - в два или три этапа. Первый этап может быть (и должен быть, если приготовление идет в три этапа) режимом авторазморозки, а последующие могут иметь индивидуально заданные мощность и время работы. Каждый этап программируется аналогично пункту 5 или 6. Переход к программированию следующего этапа осуществляется нажатием клавиши **«Мощ.»** вместо клавиши **«ОК/30 сек.».** Нажатие клавиши **«ОК/30 сек.»** приводит к началу выполнения запрограммированных этапов приготовления пищи. В любой момент времени нажатие клавиши **«Отмена»** приводит к возврату в первоначальное состояние и отмене всех этапов.
8. Каждое нажатие клавиши **«ОК/30 сек.»** во время выполнения задания добавит к длительности его выполнения 30 с. Нажатие этой клавиши в режиме ожидания приведет к немедленному началу разогрева пищи на мощности в 750 Вт длительностью 30 с.
9. Нажатие клавиши **«Отмена»** или кнопки открытия двери во время работы приведет прибор в состояние паузы, но не отменит задание. Для того чтобы продолжить приготовление нужно закрыть дверь, если ее открыли, и нажать клавишу **«ОК/30 сек.».** Нажатие клавиши **«Отмена»** в состоянии паузы отменяет задание.

10. Во время эмуляции работы печи в нижней части окна ведется протокол, который после завершения эмуляции можно сохранить в файл.
11. Работа начинается (заканчивается) с нажатия кнопки **«Начать эмуляцию»,** которая превращается в кнопку **«Закончить эмуляцию»** и наоборот.

2. Диаграмма классов

В программе основными являются классы:

- `CMicroWaveDlg`, унаследованный от класса `CDialog` библиотеки классов *MFC* и предназначенный для визуализации работы с микроволновой печью;
- `Dispatcher`, который служит дополнительным слоем между средой и автоматами, хранит в себе текущие установки печи, экземпляры классов автоматов, предоставляет автоматам интерфейс протоколирования, а также реализует входные переменные и выходные воздействия;
- `Automata`, от которого наследуются другие классы автоматов;
- A0, A1, A2, A3, A4 представляют собой реализации конкретных классов автоматов.

Помимо основных классов, программа содержит вспомогательный класс `CMicroWaveApp`, который унаследован от класс `CWinApp` библиотеки классов *MFC*. Он подготавливает приложение к работе и инициирует работу основного окна приложения.

Диаграмма классов приведена на рис. 2.

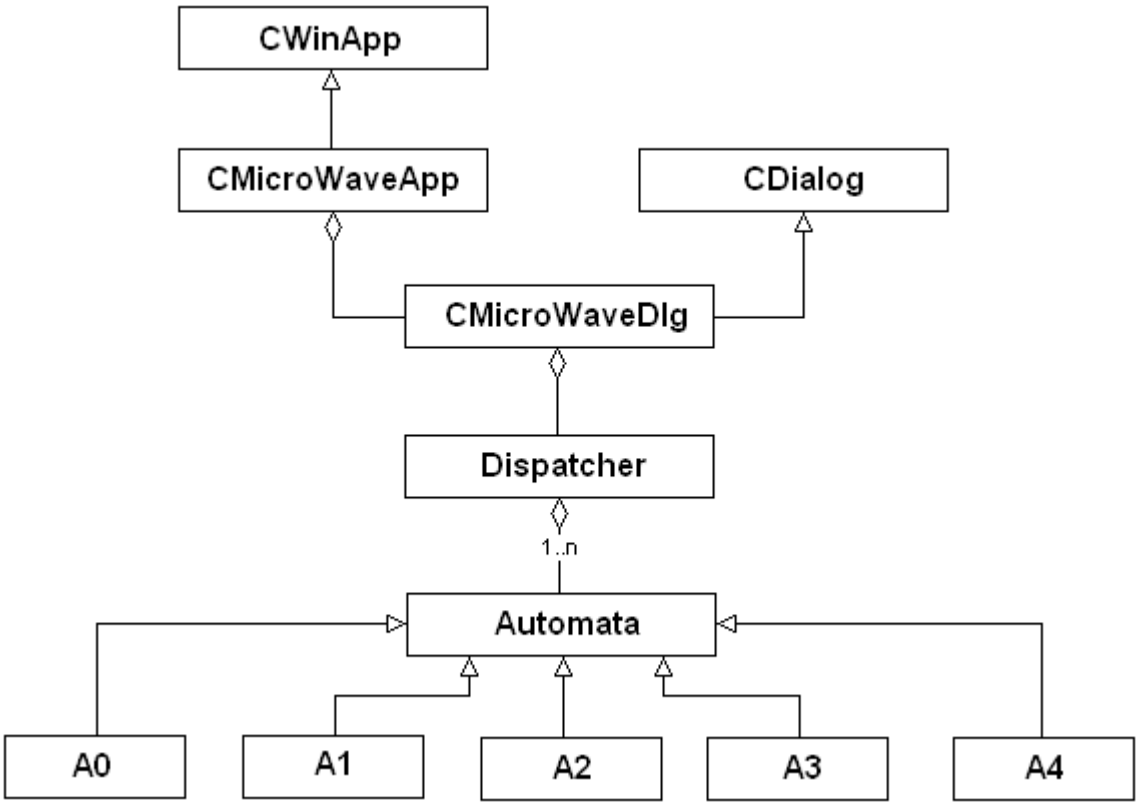

Рис. 2. Диаграмма классов

3. Класс `CMicroWaveDlg`

3.1. Словесное описание

Класс `CMicroWaveDlg` является наследником класса `CDialog`. Он обеспечивает интерфейс с пользователем и первичную обработку различных событий, поступающих от операционной системы. Он также взаимодействует с автоматами, используя диспетчер.

3.2. Краткое описание методов

- `CMircoWaveDlg()` – конструктор класса;
- `DoDataExchange()` – метод библиотеки классов *MFC*, предназначенный для взаимодействия с элементами пользовательского интерфейса;
- `OnInitDialog()` – метод, выполняющий инициализацию диалогового окна;
- `OnClose()` – метод обработки закрытия диалогового окна;
- `OnDestroy()` – метод обработки уничтожения диалогового окна;
- `OnAuto()` – метод обработки нажатия клавиши **«Авторазморозка»;**
- `OnCancel()` – метод обработки нажатия клавиши **«Отмена»**;
- `OnDoor()` – метод обработки нажатия клавиши **«Открытие/закрытие дверцы»**;
- `OnEmulation()` – метод обработки нажатия клавиши **«Начать/закончить эмуляцию»**;
- `OnMass()` – метод обработки нажатия клавиши **«+100 гр.»**;
- `OnOK()` – метод обработки нажатия клавиши **«OK/30 сек.»**;
- `OnOnemin()` – метод обработки нажатия клавиши **«1 мин./минуты»**;
- `OnSaveLog()` – метод обработки нажатия клавиши **«Сохранить в файл»**;
- `OnSetclock()` – метод обработки нажатия клавиши **«Часы»**;
- `OnSetpower()` – метод обработки нажатия клавиши **«Мощ.»**;
- `OnTenmin()` – метод обработки нажатия клавиши **«10 мин./часы»**;
- `OnTensec()` – метод обработки нажатия клавиши **«1 сек.»**;
- `OnTimer()` – метод обработки событий различных таймеров.

4. Класс `Dispatcher`

4.1. Словесное описание

Класс `Dispatcher` является слоем взаимодействия автоматов со средой, которая в данном случае является диалоговым окном. Этот класс служит сразу нескольким целям:

- создает и хранит экземпляры классов автоматов;
- хранит параметры состояния микроволновой печи, которые представляется не удобным кодировать в состояниях автоматов (например, текущее время);
- реализует входные переменные и выходные воздействия и хранит информацию, которая необходима для их работы (например, указатель на экземпляр класса диалогового окна);
- реализует функции протоколирования.

Это делает реализацию автоматов системонезависимой.

4.2 Краткое описание методов

- `Dispatcher()` – конструктор класса;
- `~Dispatcher()` – деструктор класса;
- `log_enter()` – протоколирование запуска автомата;
- `log_exit()` – протоколирование завершения обработки автоматом события;
- `log_trans()` – протоколирование изменения состояния автомата;
- `log_input()` – протоколирование проверки входных переменных;
- `log_output()` – протоколирование выходных воздействий;
- `x1()-x5()` – реализация входных переменных;
- `z1()-z32()` – реализация выходных воздействий.

5. Класс `Automata`

5.1. Словесное описание

Класс `Automata` реализует общие для всех автоматов функции и описывает интерфейс, который должен предоставлять каждый автомат. Класс `Automata` – абстрактный класс и предназначен только для того, чтобы от него наследовались реализации конкретных автоматов.

5.2. Краткое описание методов

- `Automata()` – конструктор класса;
- `~Automata()` – деструктор класса;
- `wake()` – метод, реализующий вызов автомата согласно алгоритму реализации графов перехода [2];
- `get_state()` – вспомогательная функция получения состояния автомата;
- `nested_automatas()` – метод, вызываемый из `wake()` и реализующий вызов вложенных автоматов. Этот метод класса `Automata` является виртуальным и должен быть переопределен в наследниках;
- `hande_state()` – метод, вызываемый из `wake()` и реализующий проверку условий на дугах и производящего переход по ним в другое состояние. В классе `Automata` он чисто виртуальный и должен быть переопределен в наследниках;
- `state_changed()` – метод, вызываемый из `wake()` и реализующий действия при изменении состояния автомата. Этот метод класса `Automata` является виртуальным и должен быть переопределен в наследниках.

6. Автоматный класс `A0`

6.1. Словесное описание

Класс `A0` является автоматным классом, порожденным от класса `Automata`. Описание автомата «Менеджер заданий» (`A0`), который реализует данный класс, приводится ниже.

6.2. Автомат «Менеджер заданий» (A0)

6.2.1. Словесное описание

Автомат реализует выполнение заданий микроволновой печью. Он обеспечивает начало работы, отсчет времени до конца задания и последовательное выполнение этапов. Схема связей автомата приведена на рис. 3, а граф переходов – на рис. 4.

6.2.2. Схема связей

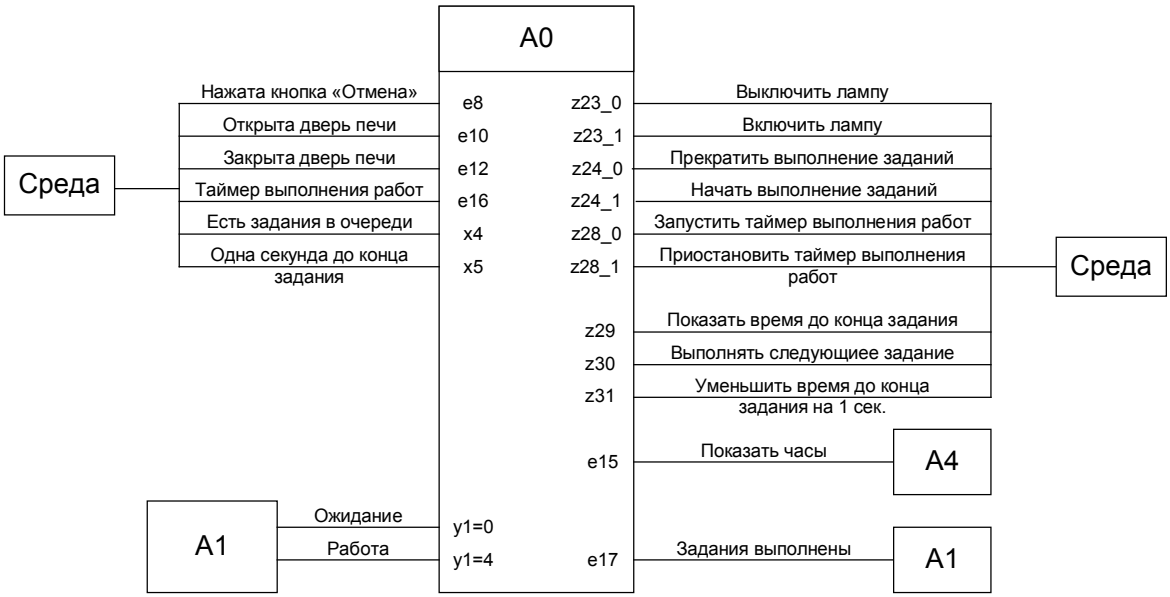

Рис. 3. Схема связей автомата A0

6.2.3. Граф переходов

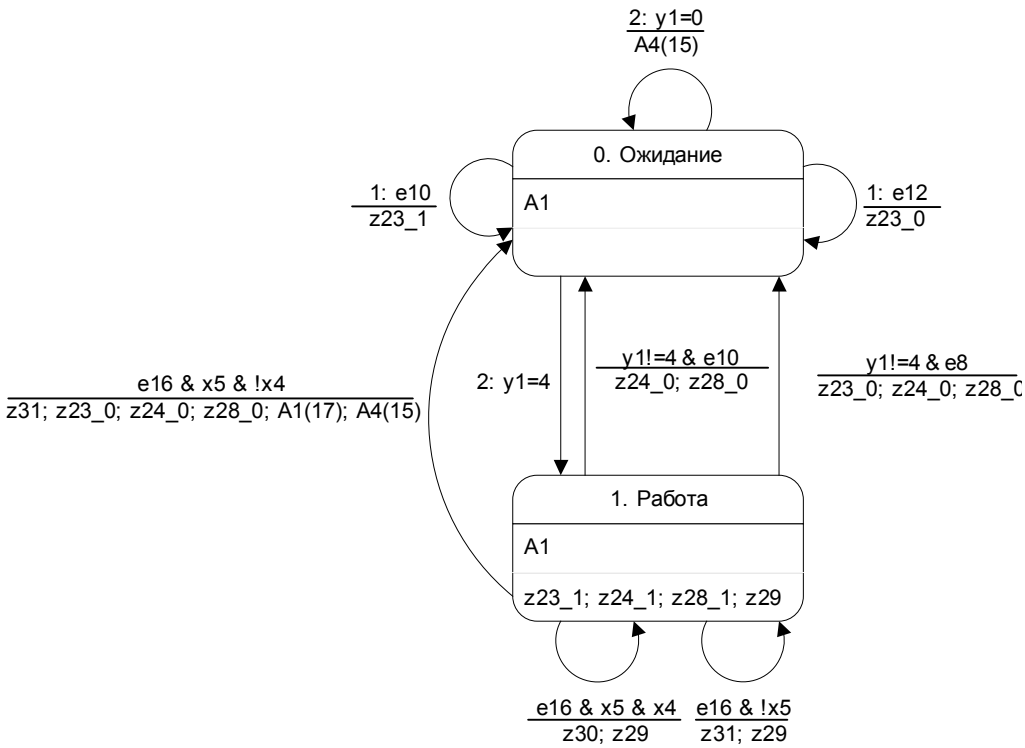

Рис. 4. Граф переходов автомата A0

7. Автоматный класс A1

7.1. Словесное описание

Класс A1 является автоматным классом, порожденным от класса Automata. Описание автомата «Обработка нажатий клавиш» (A1), который реализует данный класс, приводится ниже.

7.2. Автомат «Обработка нажатий клавиш» (A1)

7.2.1. Словесное описание

Данный автомат реализует обработку нажатий клавиш на приборной панели микроволновой печи. Схема связей автомата приведена на рис. 5, а граф переходов - на рис. 6.

7.2.2. Схема связей

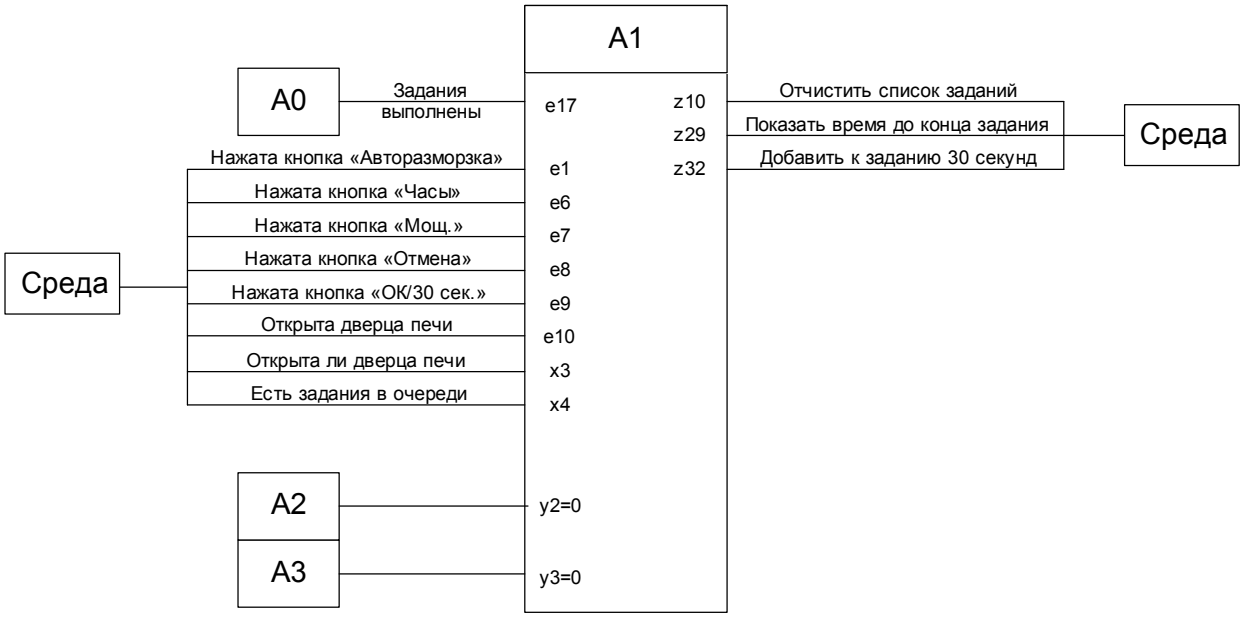

Рис. 5. Схема связей автомата A1

7.2.3. Граф переходов

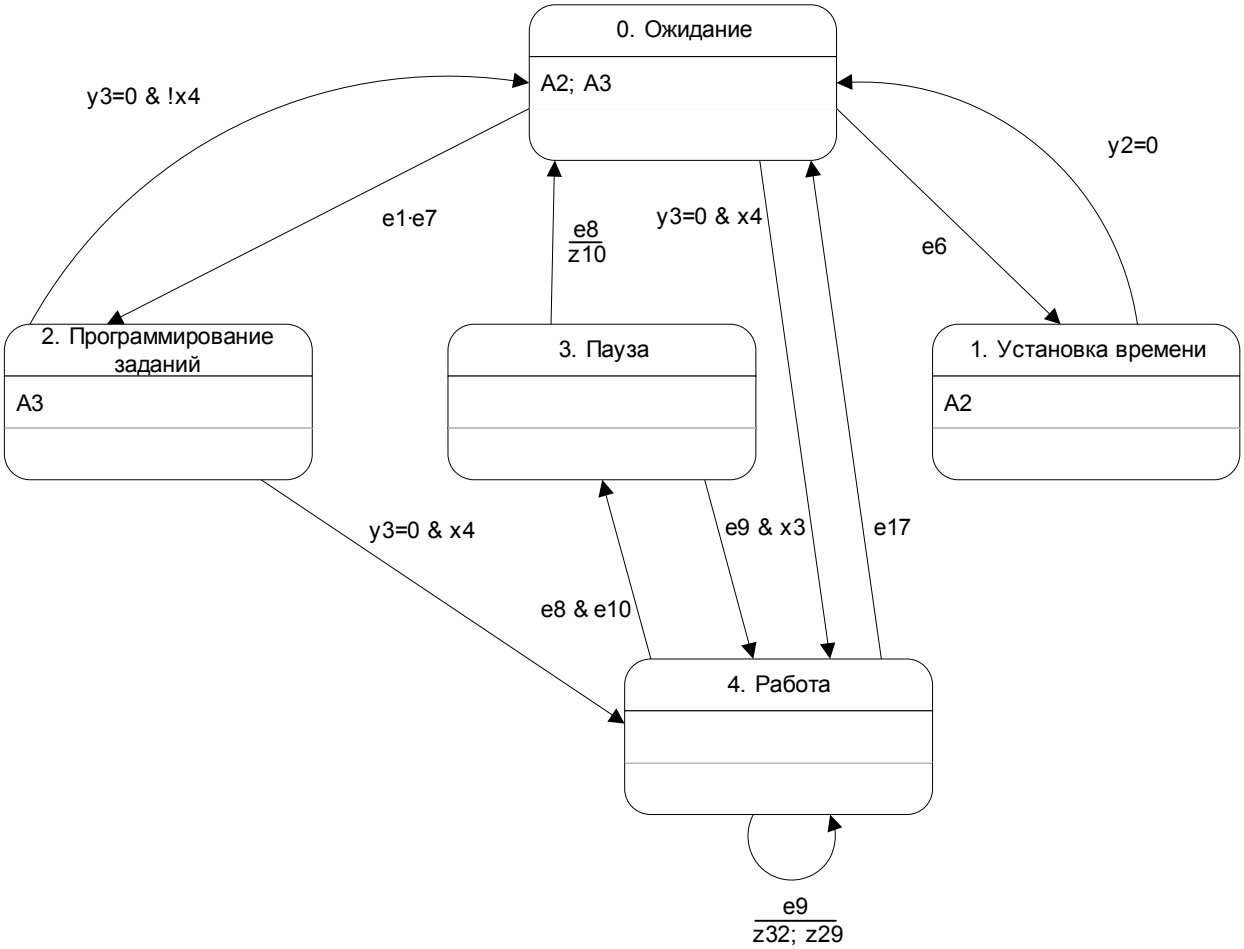

Рис. 6. Граф переходов автомата A1

8. Автоматный класс A2

8.1. Словесное описание

Класс A2 является автоматным классом, порожденным от класса Automata. Описание автомата «Установка часов» (A2), который реализует этот класс, приводится ниже.

8.2. Автомат «Установка часов» (A2)

8.2.1. Словесное описание

Данный автомат описывает обработку нажатий клавиш на приборной панели микроволновой печи во время установки часов. Схема связей автомата приведена на рис. 7, а граф переходов – на рис. 8.

8.2.2. Схема связей автомата

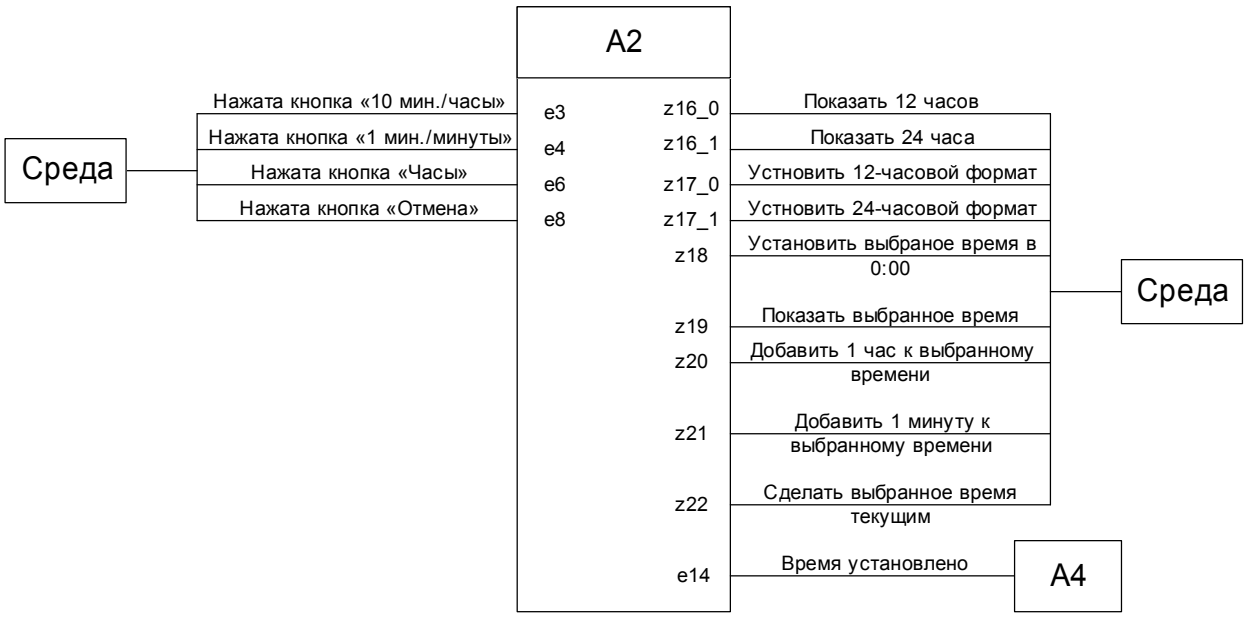

Рис. 7. Схема связей автомата A2

8.3.3. Граф переходов

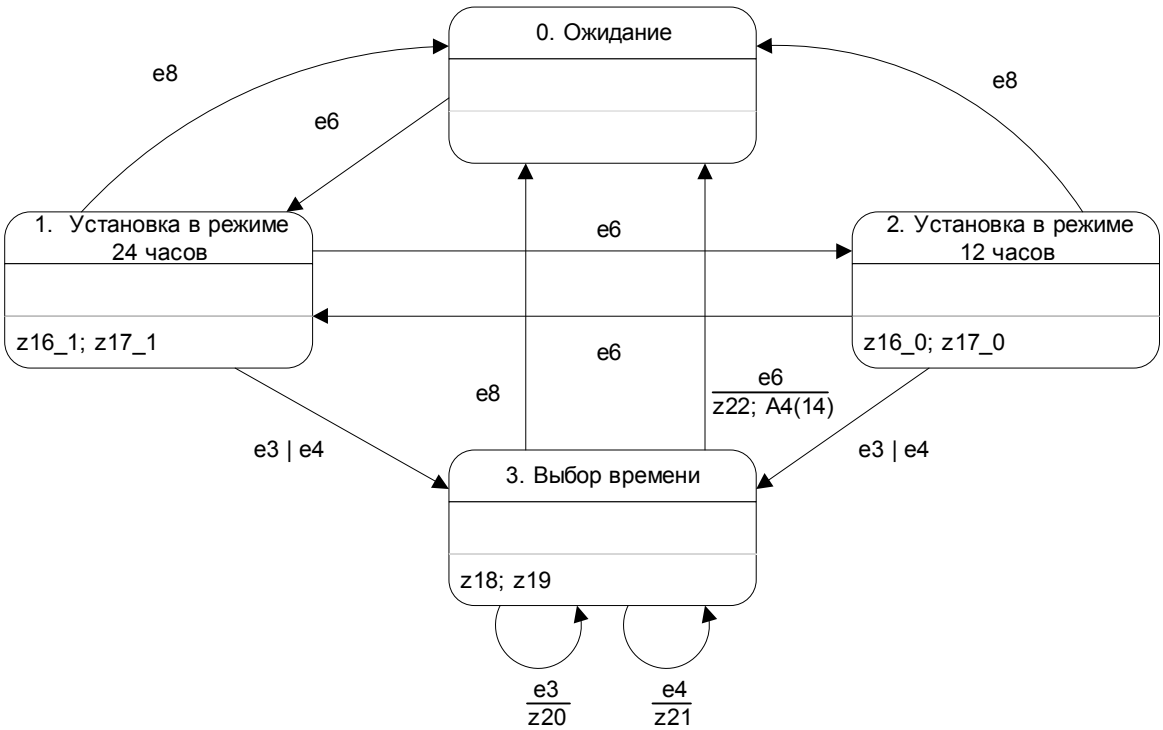

Рис. 8. Граф переходов автомата A2

9. Автоматный класс A3

9.1. Словесное описание

Класс A3 является автоматным классом, порожденным от класса Automata. Описание автомата «Программирование заданий» (A3), который реализует этот класс, приводится ниже.

9.2. Автомат «Программирование заданий» (A3)

9.2.1. Словесное описание

Данный автомат описывает обработку нажатий клавиш на приборной панели микроволновой печи во время программирования заданий. Он обеспечивает добавление заданий в очередь на выполнение и программирование нескольких этапов приготовления. Схема связей автомата приведена на рис. 9, а граф переходов – на рис. 10.

9.2.2. Схема связей

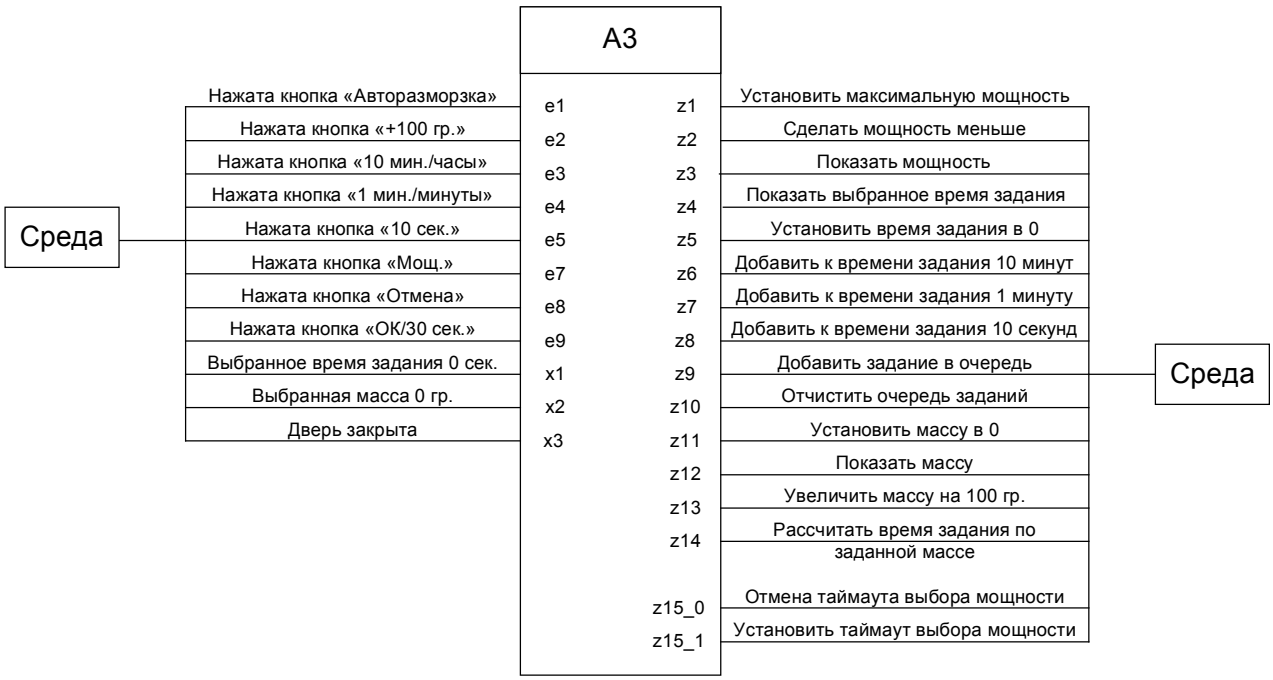

Рис. 9. Схема связей автомата A3

9.2.3. Граф переходов

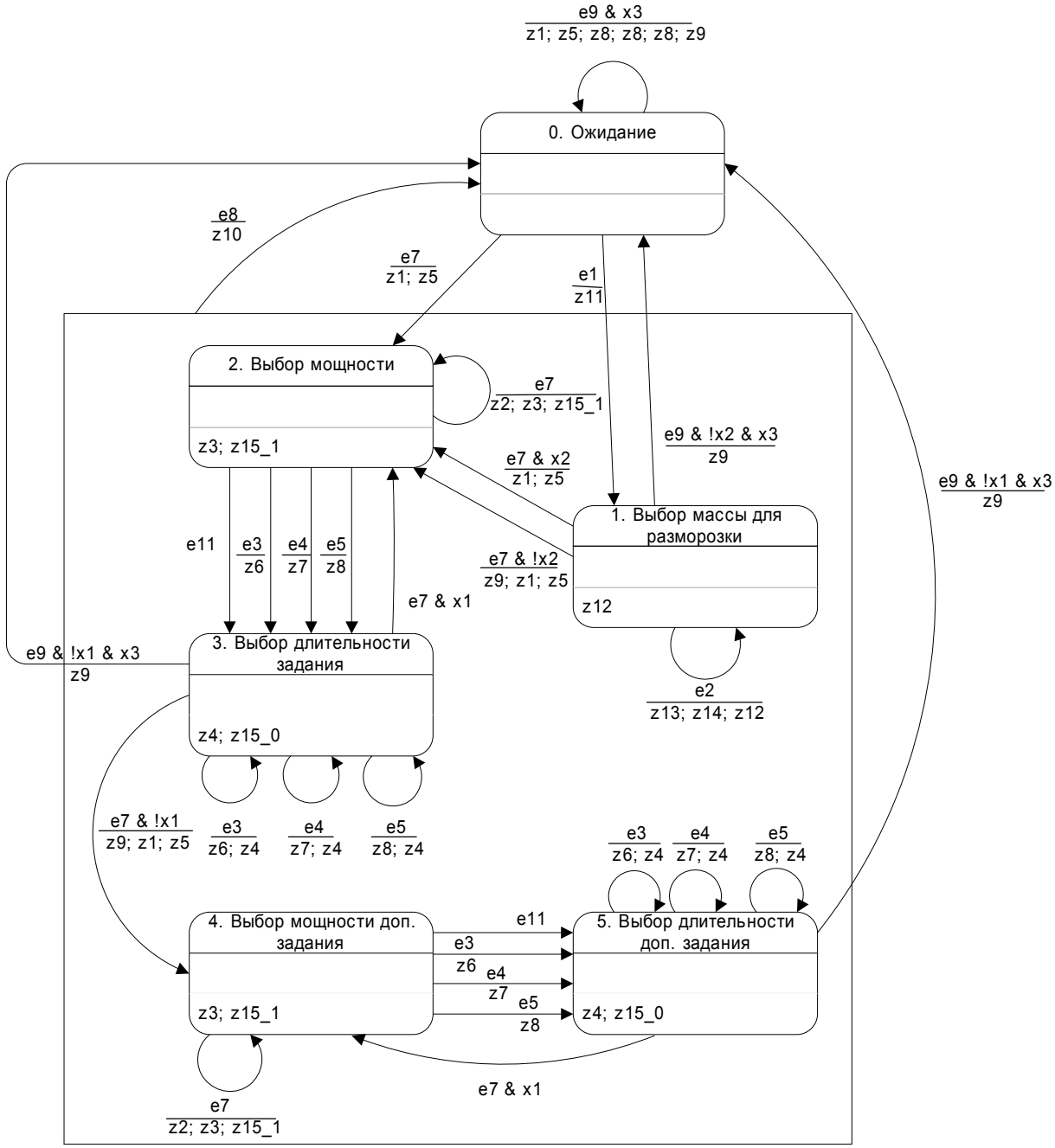

Рис. 10. Граф переходов автомата A3

10. Автоматный класс A4

10.1. Словесное описание

Класс A4 является автоматным классом, порожденным от класса Automata. Описание автомата «Часы» (A4), который реализует данный класс, приводится ниже.

10.2. Автомат «Часы» (A4)

10.2.1. Словесное описание

Данный автомат описывает обработку событий, связанных с изменением времени, и действия, связанные с отображением часов. Схема связей автомата приведена на рис. 11, а граф переходов – на рис. 12.

10.2.2. Схема связей

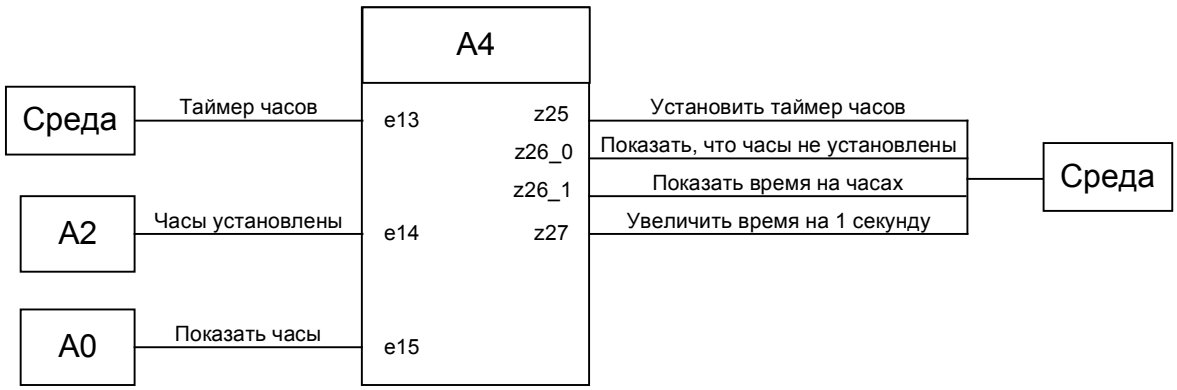

Рис. 11. Схема связей автомата A4

11.2.3. Граф переходов

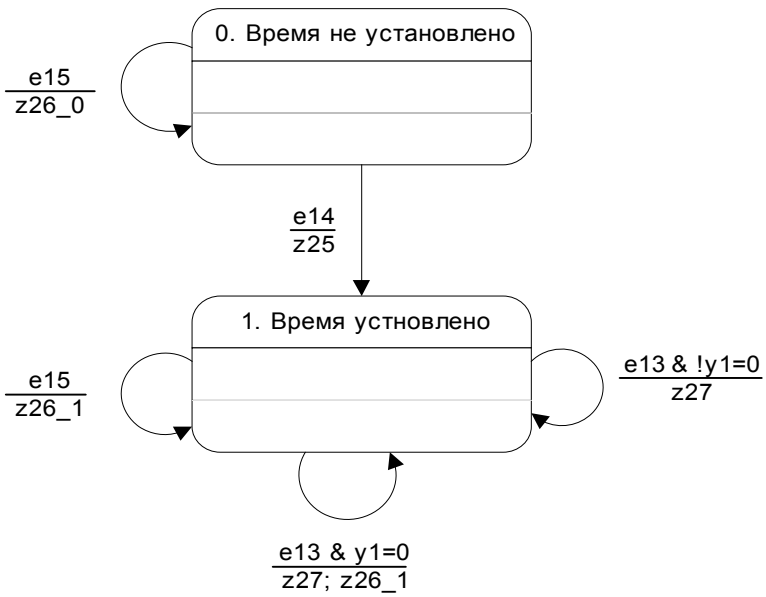

Рис. 12. Граф переходов автомата A4

Выводы

Выполненная работа показала высокую эффективность применения SWITCH-технологии для решения задач управления техническими объектами. Несомненным достоинством технологии является разделение этапа проектирования и программирования, а также возможность написания кода изоморфного графам переходов, что значительно упрощает процесс отладки.

Работа, по мнению авторов, применительно к рассматриваемому примеру опровергает утверждение Б. Страуструпа [3] о том, что «проектирование и программирование – итеративные процессы».

В приложении 1 приведен пример протокола в терминах автоматов, описывающего работу программы в одном из режимов, а приложение 2 содержит листинги программы имитации микроволновой печи.

Литература

1. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998.
2. *Шалыто А. А., Туккель Н. И.* SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем. //Программирование. 2001. №5. http://is.ifmo.ru раздел «Статьи».
3. *Страуструп Б.* Язык программирования C++. Специальное издание. СПб: Невский диалект, 2002.

Приложение 1. Пример протокола работы программы

Ниже приводится протокол работы микроволновой печи при разогреве пищи в течение 10 секунд на мощности 750 Вт.

```
18:27:44 {  A0: в состоянии 0 запущен с событием e0
18:27:44 {  A1: в состоянии 0 запущен с событием e0
18:27:44 {  A2: в состоянии 0 запущен с событием e0
18:27:44 }  A2: завершил обработку события e0 в состоянии 0
18:27:44 {  A3: в состоянии 0 запущен с событием e0
18:27:44 }  A3: завершил обработку события e0 в состоянии 0
18:27:44 >  x4: - есть еще задания в очереди? - вернул 0
18:27:44 }  A1: завершил обработку события e0 в состоянии 0
18:27:44 {  A4: в состоянии 0 запущен с событием e15
18:27:44 *  z26_0. Показать, что часы не установлены.
18:27:44 }  A4: завершил обработку события e15 в состоянии 0
18:27:44 }  A0: завершил обработку события e0 в состоянии 0
18:28:12 {  A0: в состоянии 0 запущен с событием e7
18:28:12 {  A1: в состоянии 0 запущен с событием e7
18:28:12 {  A2: в состоянии 0 запущен с событием e7
18:28:12 }  A2: завершил обработку события e7 в состоянии 0
18:28:12 {  A3: в состоянии 0 запущен с событием e7
18:28:12 *  z1. Установить максимальную мощность (750W).
18:28:12 *  z5. Установить время выполнения задания в 0.
18:28:12 T  A3: перешел из состояния 0 в состояние 2
18:28:12 *  z3. Показать текущую мощность.
18:28:12 *  z15_1. Установить таймаут выбора мощности.
18:28:12 }  A3: завершил обработку события e7 в состоянии 2
18:28:12 T  A1: перешел из состояния 0 в состояние 2
18:28:12 {  A3: в состоянии 2 запущен с событием e0
18:28:12 }  A3: завершил обработку события e0 в состоянии 2
18:28:12 }  A1: завершил обработку события e7 в состоянии 2
18:28:12 }  A0: завершил обработку события e7 в состоянии 0
18:28:14 {  A0: в состоянии 0 запущен с событием e11
18:28:14 {  A1: в состоянии 2 запущен с событием e11
18:28:14 {  A3: в состоянии 2 запущен с событием e11
18:28:14 T  A3: перешел из состояния 2 в состояние 3
18:28:14 *  z4. Показать выбранное время выполнения задания.
18:28:14 *  z15_0. Отменить таймаут выбора мощности.
18:28:14 }  A3: завершил обработку события e11 в состоянии 3
18:28:14 >  x4: - есть еще задания в очереди? - вернул 0
18:28:14 }  A1: завершил обработку события e11 в состоянии 2
18:28:14 }  A0: завершил обработку события e11 в состоянии 0
18:28:14 {  A0: в состоянии 0 запущен с событием e5
18:28:14 {  A1: в состоянии 2 запущен с событием e5
18:28:14 {  A3: в состоянии 3 запущен с событием e5
18:28:14 >  x1: - выбранное время выполнения задания равно 0? - вернул 1
18:28:14 *  z8. Добавить к выбранному времени выполнения задания 10 с
18:28:14 *  z4. Показать выбранное время выполнения задания.
18:28:14 }  A3: завершил обработку события e5 в состоянии 3
18:28:14 >  x4: - есть еще задания в очереди? - вернул 0
18:28:14 }  A1: завершил обработку события e5 в состоянии 2
18:28:14 }  A0: завершил обработку события e5 в состоянии 0
18:28:17 {  A0: в состоянии 0 запущен с событием e9
18:28:17 {  A1: в состоянии 2 запущен с событием e9
18:28:17 {  A3: в состоянии 3 запущен с событием e9
18:28:17 >  x1: - выбранное время выполнения задания равно 0? - вернул 0
18:28:17 >  x3: - дверь закрыта? - вернул 1
18:28:17 *  z9. Добавить задание в очередь.
18:28:17 T  A3: перешел из состояния 3 в состояние 0
18:28:17 }  A3: завершил обработку события e9 в состоянии 0
18:28:17 >  x4: - есть еще задания в очереди? - вернул 1
```

```
18:28:17 T  A1: перешел из состояния 2 в состояние 4
18:28:17 }  A1: завершил обработку события e9 в состоянии 4
18:28:17 T  A0: перешел из состояния 0 в состояние 1
18:28:17 {  A1: в состоянии 4 запущен с событием e0
18:28:17 }  A1: завершил обработку события e0 в состоянии 4
18:28:17 *  z23_1. Включить лампу.
18:28:18 *  z24_1. Начать выполнение заданий.
18:28:18 *  z28_1. Запустить ежесекундный таймер выполнения задания.
18:28:18 *  z29. Показать сколько времени до конца задания.
18:28:18 }  A0: завершил обработку события e9 в состоянии 1
18:28:19 {  A0: в состоянии 1 запущен с событием e16
18:28:19 {  A1: в состоянии 4 запущен с событием e16
18:28:19 }  A1: завершил обработку события e16 в состоянии 4
18:28:19 >  x4: - до конца задания одна секунда? - вернул 0
18:28:19 >  x4: - есть еще задания в очереди? - вернул 0
18:28:19 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:19 *  z29. Показать сколько времени до конца задания.
18:28:19 }  A0: завершил обработку события e16 в состоянии 1
18:28:20 {  A0: в состоянии 1 запущен с событием e16
18:28:20 {  A1: в состоянии 4 запущен с событием e16
18:28:20 }  A1: завершил обработку события e16 в состоянии 4
18:28:20 >  x4: - до конца задания одна секунда? - вернул 0
18:28:20 >  x4: - есть еще задания в очереди? - вернул 0
18:28:20 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:20 *  z29. Показать сколько времени до конца задания.
18:28:20 }  A0: завершил обработку события e16 в состоянии 1
18:28:21 {  A0: в состоянии 1 запущен с событием e16
18:28:21 {  A1: в состоянии 4 запущен с событием e16
18:28:21 }  A1: завершил обработку события e16 в состоянии 4
18:28:21 >  x4: - до конца задания одна секунда? - вернул 0
18:28:21 >  x4: - есть еще задания в очереди? - вернул 0
18:28:21 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:21 *  z29. Показать сколько времени до конца задания.
18:28:21 }  A0: завершил обработку события e16 в состоянии 1
18:28:22 {  A0: в состоянии 1 запущен с событием e16
18:28:22 {  A1: в состоянии 4 запущен с событием e16
18:28:22 }  A1: завершил обработку события e16 в состоянии 4
18:28:22 >  x4: - до конца задания одна секунда? - вернул 0
18:28:22 >  x4: - есть еще задания в очереди? - вернул 0
18:28:22 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:22 *  z29. Показать сколько времени до конца задания.
18:28:22 }  A0: завершил обработку события e16 в состоянии 1
18:28:23 {  A0: в состоянии 1 запущен с событием e16
18:28:23 {  A1: в состоянии 4 запущен с событием e16
18:28:23 }  A1: завершил обработку события e16 в состоянии 4
18:28:23 >  x4: - до конца задания одна секунда? - вернул 0
18:28:23 >  x4: - есть еще задания в очереди? - вернул 0
18:28:23 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:23 *  z29. Показать сколько времени до конца задания.
18:28:23 }  A0: завершил обработку события e16 в состоянии 1
18:28:24 {  A0: в состоянии 1 запущен с событием e16
18:28:24 {  A1: в состоянии 4 запущен с событием e16
18:28:24 }  A1: завершил обработку события e16 в состоянии 4
18:28:24 >  x4: - до конца задания одна секунда? - вернул 0
18:28:24 >  x4: - есть еще задания в очереди? - вернул 0
18:28:24 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:24 *  z29. Показать сколько времени до конца задания.
18:28:24 }  A0: завершил обработку события e16 в состоянии 1
18:28:25 {  A0: в состоянии 1 запущен с событием e16
18:28:25 {  A1: в состоянии 4 запущен с событием e16
18:28:25 }  A1: завершил обработку события e16 в состоянии 4
18:28:25 >  x4: - до конца задания одна секунда? - вернул 0
18:28:25 >  x4: - есть еще задания в очереди? - вернул 0
18:28:25 *  z31. Уменьшить время до конца задания на одну секунду.
```

```
18:28:25 *  z29. Показать сколько времени до конца задания.
18:28:25 }  A0: завершил обработку события e16 в состоянии 1
18:28:26 {  A0: в состоянии 1 запущен с событием e16
18:28:26 {  A1: в состоянии 4 запущен с событием e16
18:28:26 }  A1: завершил обработку события e16 в состоянии 4
18:28:26 >  x4: - до конца задания одна секунда? - вернул 0
18:28:26 >  x4: - есть еще задания в очереди? - вернул 0
18:28:26 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:26 *  z29. Показать сколько времени до конца задания.
18:28:26 }  A0: завершил обработку события e16 в состоянии 1
18:28:27 {  A0: в состоянии 1 запущен с событием e16
18:28:27 {  A1: в состоянии 4 запущен с событием e16
18:28:27 }  A1: завершил обработку события e16 в состоянии 4
18:28:27 >  x4: - до конца задания одна секунда? - вернул 0
18:28:27 >  x4: - есть еще задания в очереди? - вернул 0
18:28:27 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:27 *  z29. Показать сколько времени до конца задания.
18:28:27 }  A0: завершил обработку события e16 в состоянии 1
18:28:28 {  A0: в состоянии 1 запущен с событием e16
18:28:28 {  A1: в состоянии 4 запущен с событием e16
18:28:28 }  A1: завершил обработку события e16 в состоянии 4
18:28:28 >  x4: - до конца задания одна секунда? - вернул 1
18:28:28 >  x4: - есть еще задания в очереди? - вернул 0
18:28:28 *  z31. Уменьшить время до конца задания на одну секунду.
18:28:28 *  z23_0. Выключить лампу.
18:28:28 *  z24_0. Прекратить выполнение заданий.
18:28:28 *  z28_0. Приостановить ежесекундный таймер выполнения задания.
18:28:28 {  A1: в состоянии 4 запущен с событием e12
18:28:28 T  A1: перешел из состояния 4 в состояние 0
18:28:28 {  A2: в состоянии 0 запущен с событием e0
18:28:28 }  A2: завершил обработку события e0 в состоянии 0
18:28:28 {  A3: в состоянии 0 запущен с событием e0
18:28:28 }  A3: завершил обработку события e0 в состоянии 0
18:28:28 }  A1: завершил обработку события e12 в состоянии 0
18:28:28 {  A4: в состоянии 0 запущен с событием e15
18:28:28 *  z26_0. Показать, что часы не установлены.
18:28:28 }  A4: завершил обработку события e15 в состоянии 0
18:28:28 T  A0: перешел из состояния 1 в состояние 0
18:28:28 {  A1: в состоянии 0 запущен с событием e0
18:28:28 {  A2: в состоянии 0 запущен с событием e0
18:28:28 }  A2: завершил обработку события e0 в состоянии 0
18:28:28 {  A3: в состоянии 0 запущен с событием e0
18:28:28 }  A3: завершил обработку события e0 в состоянии 0
18:28:28 >  x4: - есть еще задания в очереди? - вернул 0
18:28:28 }  A1: завершил обработку события e0 в состоянии 0
18:28:28 }  A0: завершил обработку события e16 в состоянии 0
```

Приложение 2. Листинг программы

В листинг включены лишь файлы, имеющие отношение к автоматам, а файлы, сгенерированные автоматически средой *Microsoft Visual C++,* не приведены. Полные исходные тексты программы размещены на сайте http://is.ifmo.ru в разделе «Проекты».

Листинг файла «Dispatcher.h»

```
// Dispatcher.h: interface for the Dispatcher class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_DISPATCHER_H__9EE2FCAE_AC4C_4705_8653_F58E1B950E3E__INCLUDED_)
#define AFX_DISPATCHER_H__9EE2FCAE_AC4C_4705_8653_F58E1B950E3E__INCLUDED_

#include <windows.h>
#include <vector>
#include <deque>

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class Automata;
class CMicroWaveDlg;

struct Time {
  int hours;
  int minutes;
  int seconds;
};

struct Task_time {
  int minutes;
  int seconds;
};

struct Task {
  Task(Task_time t, int p) : time(t), power(p) {}
  Task_time time;
  int power;
};

struct State
{
  int current_power;
  int current_mass;
  bool clock_mode;
  Time current_time;
  Time choosed_time;

  Task_time current_task_time;
  Task_time choosed_task_time;

  static const bool CLOCK_MODE_24H;
  static const bool CLOCK_MODE_12H;

  std::deque<Task> task_queue;
};

class Dispatcher
{
private:
  State state;
  CMicroWaveDlg *parent;
  void put_log(const CString&);
public:
      Dispatcher(CMicroWaveDlg *);
      virtual ~Dispatcher();
```

```
  void log_enter(int automata, int state, int event);
  void log_trans(int automata, int old_state, int new_state);
  void log_exit(int automata, int state, int event);
  void log_input(int x, bool result, const char *descr);
  void log_output(int z, const char *descr, bool has_arg = false, bool arg = false);

  std::vector<Automata *> A;

  // input variables
  bool x1();
  bool x2();
  bool x3();
  bool x4();
  bool x5();

  // output actions
  void z1();
  void z2();
  void z3();
  void z4();
  void z5();
  void z6();
  void z7();
  void z8();
  void z9();
  void z10();
  void z11();
  void z12();
  void z13();
  void z14();
  void z15(bool);
  void z16(bool);
  void z17(bool);
  void z18();
  void z19();
  void z20();
  void z21();
  void z22();
  void z23(bool);
  void z24(bool);
  void z25();
  void z26(bool);
  void z27();
  void z28(bool);
  void z29();
  void z30();
  void z31();
  void z32();
};

#endif // !defined(AFX_DISPATCHER_H__9EE2FCAE_AC4C_4705_8653_F58E1B950E3E__INCLUDED_)
```

Листинг файла «Dispatcher.cpp»

```
// Dispatcher.cpp: implementation of the Dispatcher class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "MicroWaveDlg.h"
#include "Dispatcher.h"
#include "A0.h"
#include "A1.h"
#include "A2.h"
#include "A3.h"
#include "A4.h"

#ifdef _DEBUG
```

```
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

const bool State::CLOCK_MODE_12H = false;
const bool State::CLOCK_MODE_24H = true;

Dispatcher::Dispatcher(CMicroWaveDlg *p) : A(5), parent(p)
{
  A[0] = new A0(this);
  A[1] = new A1(this);
  A[2] = new A2(this);
  A[3] = new A3(this);
  A[4] = new A4(this);

  A[0]->wake(0);
}

Dispatcher::~Dispatcher()
{
  if(parent->second_delay_timer)
    parent->KillTimer(parent->second_delay_timer);
  if(parent->power_choose_timer)
    parent->KillTimer(parent->power_choose_timer);
  if(parent->task_timer)
    parent->KillTimer(parent->task_timer);

  for(std::vector<Automata*>::iterator it = A.begin(); it != A.end(); ++it) {
    delete (*it);
  }
}
```

Листинг файла «Input.cpp»

```
#include "stdafx.h"
#include "Dispatcher.h"
#include "MicroWaveDlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

bool Dispatcher::x1()
{
  bool result = (state.choosed_task_time.minutes == 0) &&
    (state.choosed_task_time.seconds == 0);

  log_input(1, result, "выбранное время выполнения задания равно 0?");

  return result;
}

bool Dispatcher::x2()
{
  bool result = state.current_mass == 0;
  log_input(2, result, "выбранная масса равна 0?");
  return result;
}

bool Dispatcher::x3()
{
```

```
  bool result = !parent->is_door_open;
  log_input(3, result, "дверь закрыта?");
  return result;
}

bool Dispatcher::x4()
{
  bool result = !state.task_queue.empty();
  log_input(4, result, "есть еще задания в очереди?");
  return result;
}

bool Dispatcher::x5()
{
  bool result = (state.current_task_time.minutes == 0)
    && (state.current_task_time.seconds == 1);
  log_input(4, result, "до конца задания одна секунда?");
  return result;
}
```

Листинг файла «Output.cpp»

```
#include "stdafx.h"
#include "MicroWaveDlg.h"
#include "Dispatcher.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

void Dispatcher::z1()
{
  log_output(1, "Установить максимальную мощность (750W).");
  state.current_power = 750;
}

void Dispatcher::z2()
{
  CString str;
  int& pow = state.current_power;
  switch(pow) {
  case 750:
    pow = 600;
    break;
  case 600:
    pow = 450;
    break;
  case 450:
    pow = 300;
    break;
  case 300:
    pow = 180;
    break;
  case 180:
    pow = 100;
    break;
  case 100:
    pow = 750;
    break;
  }

  str.Format("Установить следующую по списку мощность (%dW).", pow);
  log_output(2, (LPCTSTR)str);
}
```

```
void Dispatcher::z3()
{
  log_output(3, "Показать текущую мощность.");
  parent->m_Content.Format("%dW",state.current_power);
  parent->UpdateData(false);
}

void Dispatcher::z4()
{
  log_output(4, "Показать выбранное время выполнения задания.");
  parent->m_Content.Format("%.2d:%.2d",
    state.choosed_task_time.minutes,
    state.choosed_task_time.seconds);
  parent->UpdateData(false);
}

void Dispatcher::z5()
{
  log_output(5, "Установить время выполнения задания в 0.");
  state.choosed_task_time.minutes = 0;
  state.choosed_task_time.seconds = 0;
}

void Dispatcher::z6()
{
  log_output(6, "Добавить к выбранному времени выполнения задания 10 мин.");
  state.choosed_task_time.minutes += 10;
  state.choosed_task_time.minutes %= 100;
}

void Dispatcher::z7()
{
  log_output(7, "Добавить к выбранному времени выполнения задания 1 мин.");
  if(state.choosed_task_time.minutes % 10 == 9) {
    state.choosed_task_time.minutes -= 9;
  } else {
    state.choosed_task_time.minutes += 1;
  }
}

void Dispatcher::z8()
{
  log_output(8, "Добавить к выбранному времени выполнения задания 10 сек.");
  state.choosed_task_time.seconds += 10;
  state.choosed_task_time.seconds %= 60;
}

void Dispatcher::z9()
{
  log_output(9, "Добавить задание в очередь.");
  state.task_queue.push_back(Task(state.choosed_task_time, state.current_power));
}

void Dispatcher::z10()
{
  log_output(10, "Отчистить очередь заданий.");
  state.task_queue.clear();
}

void Dispatcher::z11()
{
  log_output(11, "Установить текущую массу в 0.");
  state.current_mass = 0;
}

void Dispatcher::z12()
{
  log_output(12, "Показать текущую массу.");
  parent->m_Content.Format("%d",state.current_mass);
  parent->UpdateData(false);
}
```

```
void Dispatcher::z13()
{
  log_output(13, "Добавить к текущей массе 100 гр.");
  state.current_mass = (state.current_mass + 100) % 1600;
}

void Dispatcher::z14()
{
  int seconds;
  state.current_power = 100;
  seconds = 75 * (state.current_mass / 100);
  state.choosed_task_time.minutes = seconds / 60;
  state.choosed_task_time.seconds = seconds % 60;
  log_output(14, "Рассчитать по выбранной массе время для задания.");
}

void Dispatcher::z15(bool restart)
{
  if(parent->power_choose_timer != 0) {
    parent->KillTimer(parent->power_choose_timer);
    parent->power_choose_timer = 0;
  }

  if(restart) {
    parent->power_choose_timer = parent->SetTimer(1, 2000, 0);
    log_output(15, "Установить таймаут выбора мощности.", true, true);
  } else {
    log_output(15, "Отменить таймаут выбора мощности.", true, false);
  }
}

void Dispatcher::z16(bool mode_24h)
{
  if(mode_24h) {
    parent->m_Content = "24 h";
    log_output(16, "Показачать 24 часа.", true, true);
  } else {
    log_output(16, "Показачать 12 часов.", true, false);
    parent->m_Content = "12 h";
  }
  parent->UpdateData(false);
}

void Dispatcher::z17(bool mode_24h)
{
  state.clock_mode = mode_24h;
  if(mode_24h == state.CLOCK_MODE_24H) {
    log_output(17, "Установить часы режим 24 часов.", true, true);
  } else {
    log_output(17, "Установить часы режим 24 часов.", true, false);
  }
}

void Dispatcher::z18()
{
  log_output(18, "Установить выбранное время 0:00.");
  state.choosed_time.hours = 0;
  state.choosed_time.minutes = 0;
  state.choosed_time.seconds = 0;
}

void Dispatcher::z19()
{
  log_output(19, "Показать выбранное время.");
  parent->m_Content.Format("%.2d:%.2d", state.choosed_time.hours,
                                        state.choosed_time.minutes);
  parent->UpdateData(false);
}
```

```
void Dispatcher::z20()
{
  log_output(20, "Добавить к выбранному времени 1 час.");
  if(state.clock_mode == state.CLOCK_MODE_24H) {
    state.choosed_time.hours = (state.choosed_time.hours + 1) % 24;
  } else {
    state.choosed_time.hours = (state.choosed_time.hours + 1) % 12;
  }
}

void Dispatcher::z21()
{
  log_output(21, "Добавить к выбранному времени 1 минуту.");
  state.choosed_time.minutes = (state.choosed_time.minutes + 1) % 60;
}

void Dispatcher::z22()
{
  log_output(22, "Сделать выбранное время текущим.");
  state.current_time = state.choosed_time;
}

void Dispatcher::z23(bool on)
{
  if(on) {
    log_output(23, "Включить лампу.", true, true);
  } else {
    log_output(23, "Выключить лампу.", true, false);
  }
  if(parent->is_door_open)
    return;
  if (on) {
    parent->m_Image.SetBitmap(parent->turned_on);
  } else {
    parent->m_Image.SetBitmap(parent->turned_off);
  }
}

void Dispatcher::z24(bool begin)
{
  if(begin) {
    log_output(24, "Начать выполнение заданий.", true, true);
    Task next = state.task_queue.front();
    state.task_queue.pop_front();
    state.current_task_time = next.time;
    state.current_power = next.power;
  } else {
    log_output(24, "Прекратить выполнение заданий.", true, false);
    if((state.current_task_time.minutes != 0) ||
       (state.current_task_time.seconds != 0)) {
      state.task_queue.push_front(Task(state.current_task_time, state.current_power));
    }
  }
}

void Dispatcher::z25()
{
  log_output(25, "Установить ежесекундный таймер часов.");
  parent->second_delay_timer = parent->SetTimer(3, 1000, 0);
}

void Dispatcher::z26(bool time_set)
{
  if(time_set) {
    log_output(26, "Показать текущеей время на часах.", true, true);
    if(state.current_time.seconds % 2) {
      parent->m_Content.Format("%.2d:%.2d", state.current_time.hours,
                                            state.current_time.minutes);
    } else {
      parent->m_Content.Format("%.2d %.2d", state.current_time.hours,
                                            state.current_time.minutes);
```

```
    }
  } else {
    log_output(26, "Показать, что часы не установлены.", true, false);
    parent->m_Content = "Установите часы";
  }
  parent->UpdateData(false);
}

void Dispatcher::z27()
{
  log_output(27, "Увеличить текущее время на одну секунду.");
  state.current_time.seconds += 1;
  state.current_time.minutes += state.current_time.seconds / 60;
  state.current_time.seconds %= 60;
  state.current_time.hours += state.current_time.minutes / 60;
  state.current_time.minutes %=60;
  if(state.clock_mode == state.CLOCK_MODE_24H) {
    state.current_time.hours %= 24;
  } else {
    state.current_time.hours %= 12;
  }
}

void Dispatcher::z28(bool on)
{
  if(on) {
    log_output(28, "Запустить ежесекундный таймер выполнения задания.", true, true);
    parent->task_timer = parent->SetTimer(2, 1000, 0);
  } else {
    log_output(28, "Приостановить ежесекундный таймер выполнения задания.",
               true, false);
    parent->KillTimer(parent->task_timer);
    parent->task_timer = 0;
  }
}

void Dispatcher::z29()
{
  log_output(29, "Показать сколько времени до конца задания.");
  parent->m_Content.Format("%.2d:%.2d", state.current_task_time.minutes,
                                        state.current_task_time.seconds);
  parent->UpdateData(false);
}

void Dispatcher::z30()
{
  log_output(30, "Начать выполнять следующее задание в очереди.");
  Task next = state.task_queue.front();
  state.task_queue.pop_front();
  state.current_task_time = next.time;
  state.current_power = next.power;
}

void Dispatcher::z31()
{
  log_output(31, "Уменьшить время до конца задания на одну секунду.");
  if(state.current_task_time.seconds == 0) {
    state.current_task_time.seconds = 59;
    state.current_task_time.minutes--;
  } else {
    state.current_task_time.seconds--;
  }
}

void Dispatcher::z32()
{
  log_output(31, "Добавить к времени до конца задания 30 секунд.");
  state.current_task_time.seconds += 30;
  state.current_task_time.minutes += state.current_task_time.seconds / 60;
  state.current_task_time.seconds %= 60;
  if(state.current_task_time.minutes >= 100) {
```

```
    state.current_task_time.minutes = 99;
    state.current_task_time.minutes = 59;
  }
}
```

Листинг файла «Logger.cpp»

```
#include "stdafx.h"
#include <string>
#include "MicroWaveDlg.h"
#include "Dispatcher.h"

void Dispatcher::put_log(const CString& str)
{
  parent->m_Log += str;
  parent->m_LogWnd.SetRedraw(false);
  parent->UpdateData(false);
  parent->m_LogWnd.LineScroll(parent->lines++);
  parent->m_LogWnd.SetRedraw(true);
  parent->m_LogWnd.Invalidate();
}

void Dispatcher::log_enter(int automata, int state, int event)
{
  CTime time = CTime::GetCurrentTime();
  CString t((LPCTSTR)time.Format("%H:%M:%S"));
  CString str;

  str.Format(" {  A%d: в состоянии %d запущен с событием e%d\r\n", automata, state,
event);

  put_log(t + str);
}

void Dispatcher::log_exit(int automata, int state, int event)
{
  CTime time = CTime::GetCurrentTime();
  CString t((LPCTSTR)time.Format("%H:%M:%S"));
  CString str;

  str.Format(" }  A%d: завершил обработку события e%d в состоянии %d\r\n", automata,
event, state);

  put_log(t + str);
}

void Dispatcher::log_trans(int automata, int old_state, int new_state)
{
  CTime time = CTime::GetCurrentTime();
  CString t((LPCTSTR)time.Format("%H:%M:%S"));
  CString str;

  str.Format(" T   A%d: перешел из состояния %d в состояние %d\r\n", automata,
old_state, new_state);

  put_log(t + str);
}

void Dispatcher::log_input(int x, bool result, const char* descr)
{
  CTime time = CTime::GetCurrentTime();
  CString t((LPCTSTR)time.Format("%H:%M:%S"));
  CString str;

  str.Format(" >  x%d: - %s - вернул %d\r\n", x, descr, result);

  put_log(t + str);
```

```
}

void Dispatcher::log_output(int z, const char *descr, bool has_arg, bool arg)
{
  CTime time = CTime::GetCurrentTime();
  CString t((LPCTSTR)time.Format("%H:%M:%S"));
  CString str;

  if(has_arg) {
    str.Format(" *  z%d_%d. %s\r\n", z, arg, descr);
  } else {
    str.Format(" *  z%d. %s\r\n", z, descr);
  }

  put_log(t + str);
}
```

Листинг файла «Automata.h»

```
// Automata.h: interface for the Automata class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_AUTOMATA_H__39EFE032_7B8A_48A5_810A_16C9F7E75F8B__INCLUDED_)
#define AFX_AUTOMATA_H__39EFE032_7B8A_48A5_810A_16C9F7E75F8B__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class Dispatcher;

class Automata
{
protected:
  int number;
  int state;
  Dispatcher *dispatcher;
protected:
  virtual void state_changed() = 0;
  virtual void handle_state(const int e) = 0;
  virtual void nested_automatas(const int e) = 0;
public:
  int get_state();

  void wake(const int event);

  Automata(Dispatcher * d);
  virtual ~Automata();
};

#endif // !defined(AFX_AUTOMATA_H__39EFE032_7B8A_48A5_810A_16C9F7E75F8B__INCLUDED_)
```

Листинг файла «Automata.cpp»

```
// Automata.cpp: implementation of the Automata class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "MicroWave.h"
#include "Dispatcher.h"
#include "Automata.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
```

```
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

Automata::Automata(Dispatcher *d) : dispatcher(d), state(0)
{
}

Automata::~Automata()
{

}

void Automata::wake(const int event)
{
  int old_state = state;
  dispatcher->log_enter(number, state, event);
  nested_automatas(event);
  handle_state(event);
  if(state != old_state) {
    dispatcher->log_trans(number, old_state, state);
    state_changed();
  }
  dispatcher->log_exit(number, state, event);
}

int Automata::get_state()
{
  return state;
}
```

Листинг файла «A0.h»

```
// A0.h: interface for the A0 class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_A0_H__7ABC4FC7_F4F8_4520_91C6_05ED15AD1051__INCLUDED_)
#define AFX_A0_H__7ABC4FC7_F4F8_4520_91C6_05ED15AD1051__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Automata.h"

class A0 : public Automata
{
public:
  A0(Dispatcher *d);
  virtual ~A0();
protected:
  void handle_state(const int e);
  void state_changed();
  void nested_automatas(const int e);
};

#endif // !defined(AFX_A0_H__7ABC4FC7_F4F8_4520_91C6_05ED15AD1051__INCLUDED_)
```

Листинг файла «A0.cpp»

```
// A0.cpp: implementation of the A0 class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "A0.h"
#include "Dispatcher.h"


#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

A0::A0(Dispatcher *d) : Automata(d)
{
  number = 0;
}

A0::~A0()
{

}

void A0::nested_automatas(const int e)
{
  switch(state) {
  case 0:
    dispatcher->A[1]->wake(e);
    break;
  case 1:
    dispatcher->A[1]->wake(e);
    break;
  }
}

void A0::handle_state(const int e)
{
  bool _x4, _x5;
  switch(state) {
  case 0:
    if(e == 10) {
      dispatcher->z23(1);
    } else if(e == 12) {
      dispatcher->z23(0);
    } else if (dispatcher->A[1]->get_state() == 4) {
      state = 1;
    } else if (dispatcher->A[1]->get_state() == 0) {
      dispatcher->A[4]->wake(15);
    }
    break;
  case 1:
    _x5 = dispatcher->x5();
    _x4 = dispatcher->x4();
    if((dispatcher->A[1]->get_state() != 4) && (e == 8)) {
      state = 0;
      dispatcher->z23(0);
      dispatcher->z24(0);
      dispatcher->z28(0);
```

```
    } else if ((dispatcher->A[1]->get_state() != 4) && (e == 10)) {
      state = 0;
      dispatcher->z24(0);
      dispatcher->z28(0);
    } else if (e == 16 && _x5 && !_x4) {
      state = 0;
      dispatcher->z31();
      dispatcher->z23(0);
      dispatcher->z24(0);
      dispatcher->z28(0);
      dispatcher->A[1]->wake(17);
      dispatcher->A[4]->wake(15);
    } else if ((e == 16) && _x5 && _x4) {
      dispatcher->z30();
      dispatcher->z29();
    } else if ((e == 16) && !_x5) {
      dispatcher->z31();
      dispatcher->z29();
    }
    break;
  }
}

void A0::state_changed()
{
  switch(state) {
  case 0:
    dispatcher->A[1]->wake(0);
    break;
  case 1:
    dispatcher->A[1]->wake(0);
    dispatcher->z23(1);
    dispatcher->z24(1);
    dispatcher->z28(1);
    dispatcher->z29();
    break;
  }
}
```

Листинг файла «A1.h»

```
// A1.h: interface for the A1 class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_A1_H__FA3EB6DD_B0A0_448F_A4F3_9BDB008BA584__INCLUDED_)
#define AFX_A1_H__FA3EB6DD_B0A0_448F_A4F3_9BDB008BA584__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Automata.h"


class A1 : public Automata
{
protected:
  void handle_state(const int e);
  void nested_automatas(const int e);
  void state_changed();
public:
  A1(Dispatcher *d);
  virtual ~A1();
};

#endif // !defined(AFX_A1_H__FA3EB6DD_B0A0_448F_A4F3_9BDB008BA584__INCLUDED_)
```

Листинг файла «A1.cpp»

```
// A1.cpp: implementation of the A1 class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Dispatcher.h"
#include "A1.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

A1::A1(Dispatcher *d) : Automata(d)
{
  number = 1;
}

A1::~A1()
{

}

void A1::nested_automatas(const int e) {
  switch(state) {
  case 0:
    dispatcher->A[2]->wake(e);
    dispatcher->A[3]->wake(e);
    break;
  case 1:
    dispatcher->A[2]->wake(e);
    break;
  case 2:
    dispatcher->A[3]->wake(e);
    break;
  case 3:
    break;
  case 4:
    break;
  }
}

void A1::handle_state(const int e){
  bool _x4;
  switch(state) {
  case 0:
    if(e == 1 || e == 7) {
      state = 2;
    } else if(e == 6) {
      state = 1;
    } else if ((dispatcher->A[3]->get_state() == 0) && dispatcher->x4()) {
      state = 4;
    }
    break;
  case 1:
    if(dispatcher->A[2]->get_state() == 0) {
      state = 0;
    }
    break;
  case 2:
    _x4 = dispatcher->x4();
    if((dispatcher->A[3]->get_state() == 0) && _x4) {
      state = 4;
    } else if((dispatcher->A[3]->get_state() == 0) && !_x4) {
```

```
      state = 0;
    }
    break;
  case 3:
    if(e == 8) {
      state = 0;
      dispatcher->z10();
    } else if ((e == 9) && dispatcher->x3()) {
      state = 4;
    }
    break;
  case 4:
    if(e == 9) {
      dispatcher->z32();
      dispatcher->z29();
    } else if (e == 17) {
      state = 0;
    } else if (e == 8|| e == 10) {
      state = 3;
    }
    break;
  }
}

void A1::state_changed()
{
  switch(state) {
  case 0:
    dispatcher->A[2]->wake(0);
    dispatcher->A[3]->wake(0);
    break;
  case 1:
    dispatcher->A[2]->wake(0);
    break;
  case 2:
    dispatcher->A[3]->wake(0);
    break;
  case 3:
    break;
  case 4:
    break;
  }
}
```

Листинг файла «A2.h»

```
// A2.h: interface for the A2 class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_A2_H__839CF12F_227E_4056_B6FE_08F55ED58188__INCLUDED_)
#define AFX_A2_H__839CF12F_227E_4056_B6FE_08F55ED58188__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Automata.h"

class A2 : public Automata
{
protected:
  void handle_state(const int e);
  void nested_automatas(const int e) {};
  void state_changed();
public:
  A2(Dispatcher *d);
  virtual ~A2();
```

```
};

#endif // !defined(AFX_A2_H__839CF12F_227E_4056_B6FE_08F55ED58188__INCLUDED_)
```

Листинг файла «A2.cpp»

```
// A2.cpp: implementation of the A2 class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Dispatcher.h"
#include "A2.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

A2::A2(Dispatcher *d) : Automata(d)
{
  number = 2;
}

A2::~A2()
{

}

void A2::handle_state(const int e)
{
  switch(state) {
  case 0:
    if(e == 6) {
      state = 1;
    }
    break;
  case 1:
    if(e == 6) {
      state = 2;
    } else if((e == 3) || (e == 4)) {
      state = 3;
    } else if(e == 8) {
      state = 0;
    }
    break;
  case 2:
    if(e == 6) {
      state = 1;
    } else if((e == 3) || (e == 4)) {
      state = 3;
    } else if(e == 8) {
      state = 0;
    }
    break;
  case 3:
    if(e == 6) {
      state = 0;
      dispatcher->z22();
      dispatcher->A[4]->wake(14);
    } else if(e == 3) {
      dispatcher->z20();
      dispatcher->z19();
    } else if(e == 4) {
```

```
      dispatcher->z21();
      dispatcher->z19();
    } else if(e == 8) {
      state = 0;
    }
    break;
  }
}

void A2::state_changed()
{
  switch(state) {
  case 0:
    break;
  case 1:
    dispatcher->z16(1);
    dispatcher->z17(1);
    break;
  case 2:
    dispatcher->z16(0);
    dispatcher->z17(0);
    break;
  case 3:
    dispatcher->z18();
    dispatcher->z19();
  }
}
```

Листинг файла «A3.h»

```
// A3.h: interface for the A3 class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_A3_H__8396887B_E8EE_4783_932C_E2B1D809223A__INCLUDED_)
#define AFX_A3_H__8396887B_E8EE_4783_932C_E2B1D809223A__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Automata.h"

class A3 : public Automata
{
protected:
  void handle_state(const int e);
  void nested_automatas(const int e) {};
  void state_changed();
public:
  A3(Dispatcher *d);
  virtual ~A3();

};

#endif // !defined(AFX_A3_H__8396887B_E8EE_4783_932C_E2B1D809223A__INCLUDED_)
```

Листинг файла «A3.cpp»

```
// A3.cpp: implementation of the A3 class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Dispatcher.h"
#include "A3.h"
```

```
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

A3::A3(Dispatcher *d) : Automata(d)
{
    number = 3;
}

A3::~A3()
{

}

void A3::handle_state(const int e)
{
  bool _x1, _x2;
  switch(state) {
  case 0:
    if(e == 1) {
      state = 1;
      dispatcher->z11();
    } else if(e == 7) {
      state = 2;
      dispatcher->z1();
      dispatcher->z5();
    } else if((e == 9) && dispatcher->x3()) {
      dispatcher->z1();
      dispatcher->z5();
      dispatcher->z8();
      dispatcher->z8();
      dispatcher->z8();
      dispatcher->z9();
    }
    break;
  case 1:
    _x2 = dispatcher->x2();
    if(e == 2) {
      dispatcher->z13();
      dispatcher->z14();
      dispatcher->z12();
    } else if ((e == 7) && _x2) {
      state = 2;
      dispatcher->z1();
      dispatcher->z5();
    } else if ((e == 7) && !_x2) {
      state = 2;
      dispatcher->z9();
      dispatcher->z1();
      dispatcher->z5();
    } else if (e == 8) {
      state = 0;
      dispatcher->z10();
    } else if ((e == 9) && !_x2 && dispatcher->x3()) {
      state = 0;
      dispatcher->z9();
    }
    break;
  case 2:
    if(e == 3) {
      state = 3;
      dispatcher->z6();
    } else if(e == 4) {
      state = 3;
```

```
    dispatcher->z7();
  } else if(e == 5) {
    state = 3;
    dispatcher->z8();
  } else if(e == 7) {
    dispatcher->z2();
    dispatcher->z3();
    dispatcher->z15(true);
  } else if(e == 8) {
    state = 0;
    dispatcher->z10();
  } else if(e == 11) {
    state = 3;
  }
  break;
case 3:
  _x1 = dispatcher->x1();
  if(e == 3) {
    dispatcher->z6();
    dispatcher->z4();
  } else if (e == 4) {
    dispatcher->z7();
    dispatcher->z4();
  } else if (e == 5) {
    dispatcher->z8();
    dispatcher->z4();
  } else if ((e == 7) && _x1) {
    state = 2;
  } else if ((e == 7) && !_x1) {
    state = 4;
    dispatcher->z9();
    dispatcher->z1();
    dispatcher->z5();
  } else if (e == 8) {
    state = 0;
    dispatcher->z10();
  } else if ((e == 9) && !_x1 && dispatcher->x3()) {
    state = 0;
    dispatcher->z9();
  }
  break;
case 4:
  if(e == 3) {
    state = 5;
    dispatcher->z6();
  } else if(e == 4) {
    state = 5;
    dispatcher->z7();
  } else if(e == 5) {
    state = 5;
    dispatcher->z8();
  } else if(e == 7) {
    dispatcher->z2();
    dispatcher->z3();
    dispatcher->z15(true);
  } else if(e == 8) {
    state = 0;
    dispatcher->z10();
  } else if(e == 11) {
    state = 5 ;
  }
  break;
case 5:
  _x1 = dispatcher->x1();
  if(e == 3) {
    dispatcher->z6();
    dispatcher->z4();
  } else if (e == 4) {
    dispatcher->z7();
    dispatcher->z4();
  } else if (e == 5) {
```

```
      dispatcher->z8();
      dispatcher->z4();
    } else if ((e == 7) && _x1) {
      state = 4;
    } else if (e == 8) {
      state = 0;
      dispatcher->z10();
    } else if ((e == 9) && !_x1 && dispatcher->x3()) {
      state = 0;
      dispatcher->z9();
    }
    break;
  }
}

void A3::state_changed()
{
  switch(state) {
  case 0:
    break;
  case 1:
    dispatcher->z12();
    break;
  case 2:
    dispatcher->z3();
    dispatcher->z15(true);
    break;
  case 3:
    dispatcher->z4();
      dispatcher->z15(false);
    break;
  case 4:
    dispatcher->z3();
    dispatcher->z15(true);
    break;
  case 5:
    dispatcher->z4();
    dispatcher->z15(false);
    break;
  }
}
```

Листинг файла «A4.h»

```
// A4.h: interface for the A4 class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_A4_H__76090D0F_169A_439E_BA0B_93BD7B6231F5__INCLUDED_)
#define AFX_A4_H__76090D0F_169A_439E_BA0B_93BD7B6231F5__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "Automata.h"

class A4 : public Automata
{
protected:
  void handle_state(const int e);
  void nested_automatas(const int e) {};
  void state_changed() {};
public:
  A4(Dispatcher *d);
  virtual ~A4();

};
```

```
#endif // !defined(AFX_A4_H__76090D0F_169A_439E_BA0B_93BD7B6231F5__INCLUDED_)
```

Листинг файла «A4.cpp»

```
// A4.cpp: implementation of the A4 class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Dispatcher.h"
#include "A4.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

A4::A4(Dispatcher *d) : Automata(d)
{
    number = 4;
}

A4::~A4()
{

}

void A4::handle_state (const int e) {
  switch(state) {
  case 0:
    if (e == 15) {
      dispatcher->z26(0);
    } else if (e == 14) {
      state = 1;
      dispatcher->z25();
    }
    break;
  case 1:
    if (e == 15) {
      dispatcher->z26(1);
    } else if ((e == 13) && !(dispatcher->A[1]->get_state() == 0)) {
      dispatcher->z27();
    } else if ((e == 13) && (dispatcher->A[1]->get_state() == 0)) {
      dispatcher->z27();
      dispatcher->z26(1);
    }
    break;
  }
}
```