Санкт-Петербургский государственный институт
точной механики и оптики (технический университет)

Кафедра «Компьютерные технологии»

А. А. Дистель, Д. А. Кобак, А. А. Шалыто

Система управления дорожным светофором

Программирование с явным выделением состояний

Проектная документация

Проект создан в рамках
«Движения за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург
2003

СОДЕРЖАНИЕ

1. Введение

Предлагаемая проектная документация системы управления дорожным светофором обеспечивает достижение нескольких целей.

Авторы не утверждают, что их проект должен быть немедленно взят на вооружение Городским Транспортным Управлением, и не ратуют за переделку всех светофоров в городе. Однако светофор представляет собой практически идеальный объект для автоматизации, при программировании которого могут быть использованы автоматы[1]. Для светофоров, как, впрочем, и для других объектов управления, документация на программное обеспечение является закрытой, в то время как авторы поддерживают движение «за открытую проектную документацию»[2].

Из-за простоты и наглядности объекта управления проектная документация на рассматриваемую систему может служить наглядным пособием по применению теории автоматов в программировании и, в особенности, SWITCH-технологии[2]. Поэтому проект создания программного обеспечения светофора, используя все основные идеи теории, значительно проще соответствующего проекта для танков из игры *Robocode*[3], а тем более – для дизель-генератора[4], и, следовательно, лучше подходит для первого знакомства с новой концепцией программирования.

Проект реализован на языке C++.

2. Постановка задачи

Рассматривается дорожный пешеходный светофор (его визуализатор приведен на рис. 1) с тремя огнями (красным - *red*, желтым - *yellow* и зеленым - *green*), который снабжен кнопкой включения зеленого света для пешеходов (*Go*) и дистанционным пультом управления (*Desk*). Визуализатор, кроме пульта управления, содержит имитатор внешних воздействий на светофор (*External Actions*).

---

[1] *Каган Б.М., Сташин В.В.* Основы проектирования микропроцессорных устройств автоматики. М.: Энергоатомиздат, 1987.

[2] http://is.ifmo.ru

[3] *Туккель Н.И., Шалыто А.А.* Система управления танком для игры Robocode. Объектно-ориентированное программирование с явным выделением состояний.
http://is.ifmo.ru/?i0=projects&i1=tanks

[4] *Туккель Н.И., Шалыто А.А.* Система управления дизель-генератором (фрагмент). Программирование с явным выделением состояний.
http://is.ifmo.ru/?i0=projects&i1=dg

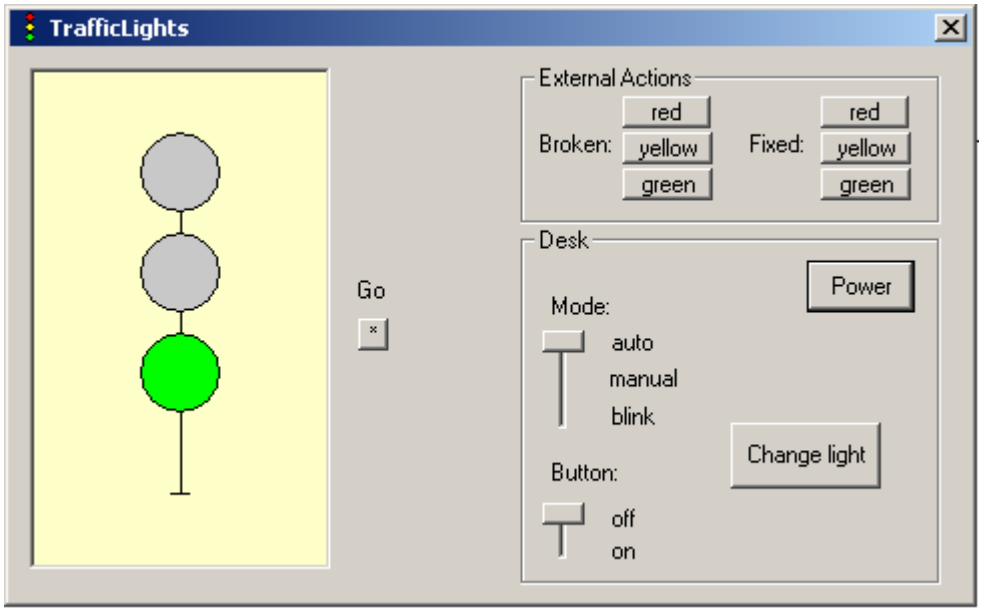

Рис. 1. Визуализатор работы светофора

Светофор может работать в нескольких режимах, выбираемых с помощью переключателя *Mode*: автоматическом (*auto*), ручном (*manual*) и режиме мигания (*blink*). Кроме того, имеется переключатель (*Button*), который обеспечивает возможность отключения (*off*) и включения (*on*) указанной выше кнопки.

В автоматическом режиме светофор по таймеру последовательно переключается следующим образом: друг за другом загораются зеленый, желтый и красный огни, причем зеленый перед выключением фиксированное время мигает, а красный – горит вместе с желтым светом.

Если при этом кнопка включения зеленого света (*Go*) разблокирована (переключатель *Button* находится в положении *on*), то светофор переключается с красного света на зеленый только при ее нажатии. Впрочем, если пешеходы будут постоянно нажимать на эту кнопку, красный свет для них должен гореть не менее предусмотренного автоматическим режимом времени.

Ручной режим предполагает переключение с красного на зеленый свет (и наоборот) только по сигналу с пульта управления (кнопка *Change light*).

Режим мигания означает, что светофор мигает желтым светом и должен интерпретироваться участниками дорожно-транспортного движения как неработающий[5].

Пульт управления содержит также кнопку включения/выключения светофора (*Power*).

[5] Правила дорожного движения. *http://www.rules.ru*

Кроме того, в системе предусмотрен контроль перегорания ламп, что обеспечивается с помощью имитатора внешних воздействий. В случае, если из строя выходит лампа красного или зеленого света (*Broken: red, green*), светофор автоматически переходит в режим мигания. Если же перегорает и лампа желтого света (*Broken: yellow*), то светофор выключается и может быть включен только после замены ламп (*Fixed: red, green, yellow*).

3. Класс «Светофор»

Система управления светофором реализована одним классом.

3.1. Словесное описание

Класс «Светофор» инкапсулирует одноименный автомат и все необходимые для его работы методы. Открытые методы класса вызываются извне, например, по нажатию соответствующих кнопок пользовательского интерфейса программы. Все эти методы перечислены и прокомментированы на структурной схеме класса.

3.2. Структурная схема класса

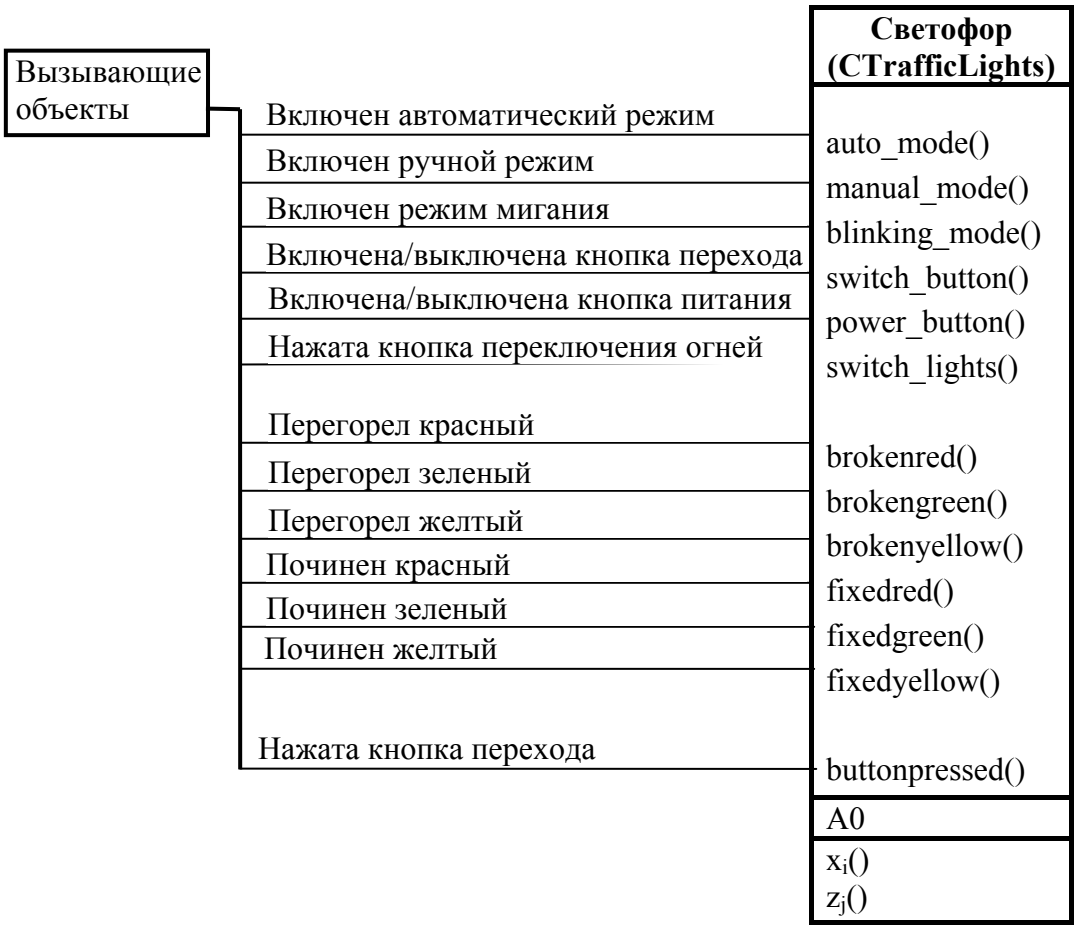

Рис. 2. Структурная схема класса

3.3. Автомат управления светофором

3.3.1. Словесное описание

Автомат управления светофором имеет восемь состояний, смысл которых понятен из названий.

Схема связей автомата и его граф переходов приведены на рис. 3 и рис. 4.

Обратим внимание, что те дуги, на которых не установлены приоритеты, ортогональны по физическому смыслу задачи.

3.3.2. Схема связей

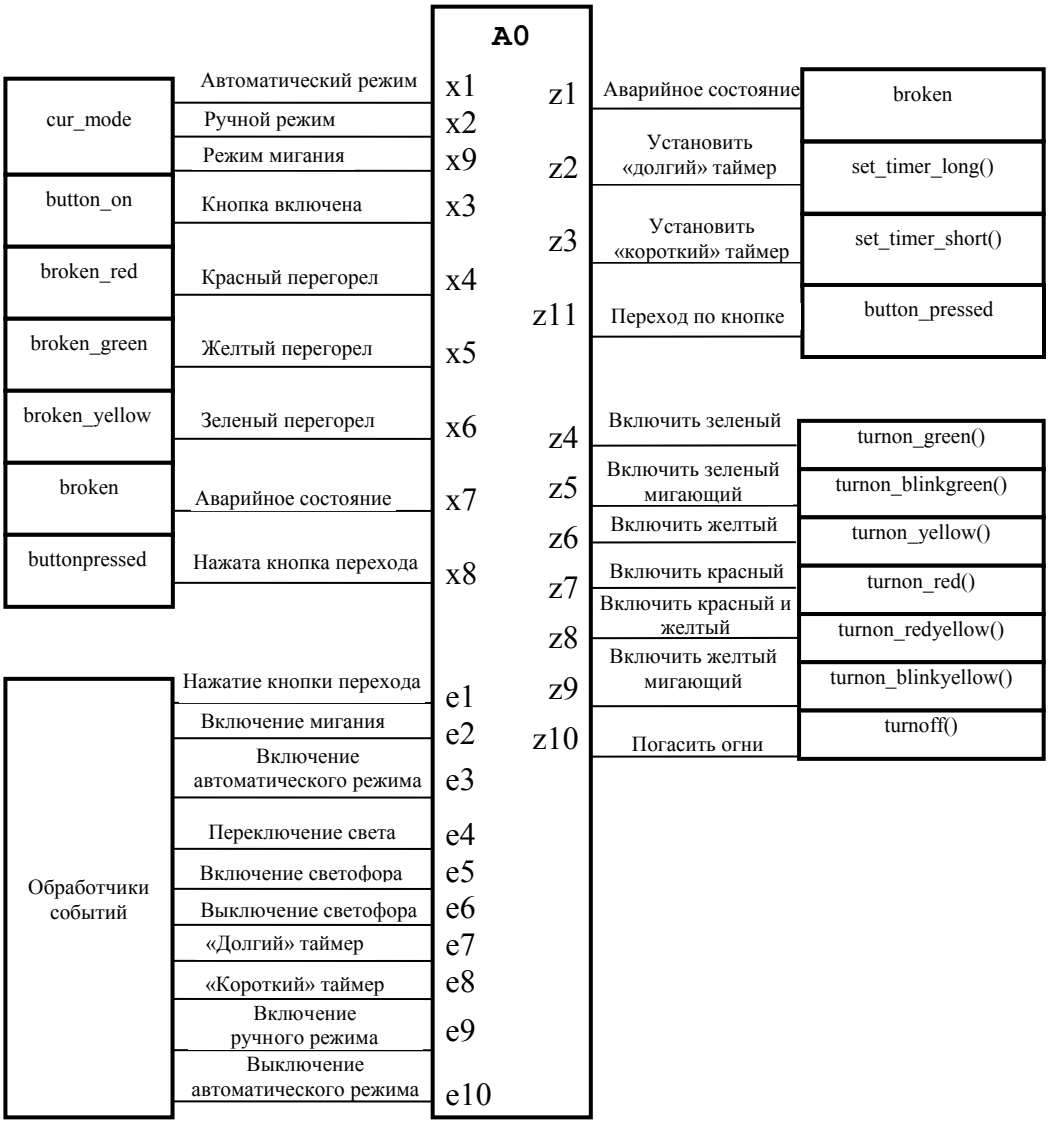

Рис. 3. Схема связей автомата управления светофором

3.3.3. Граф переходов

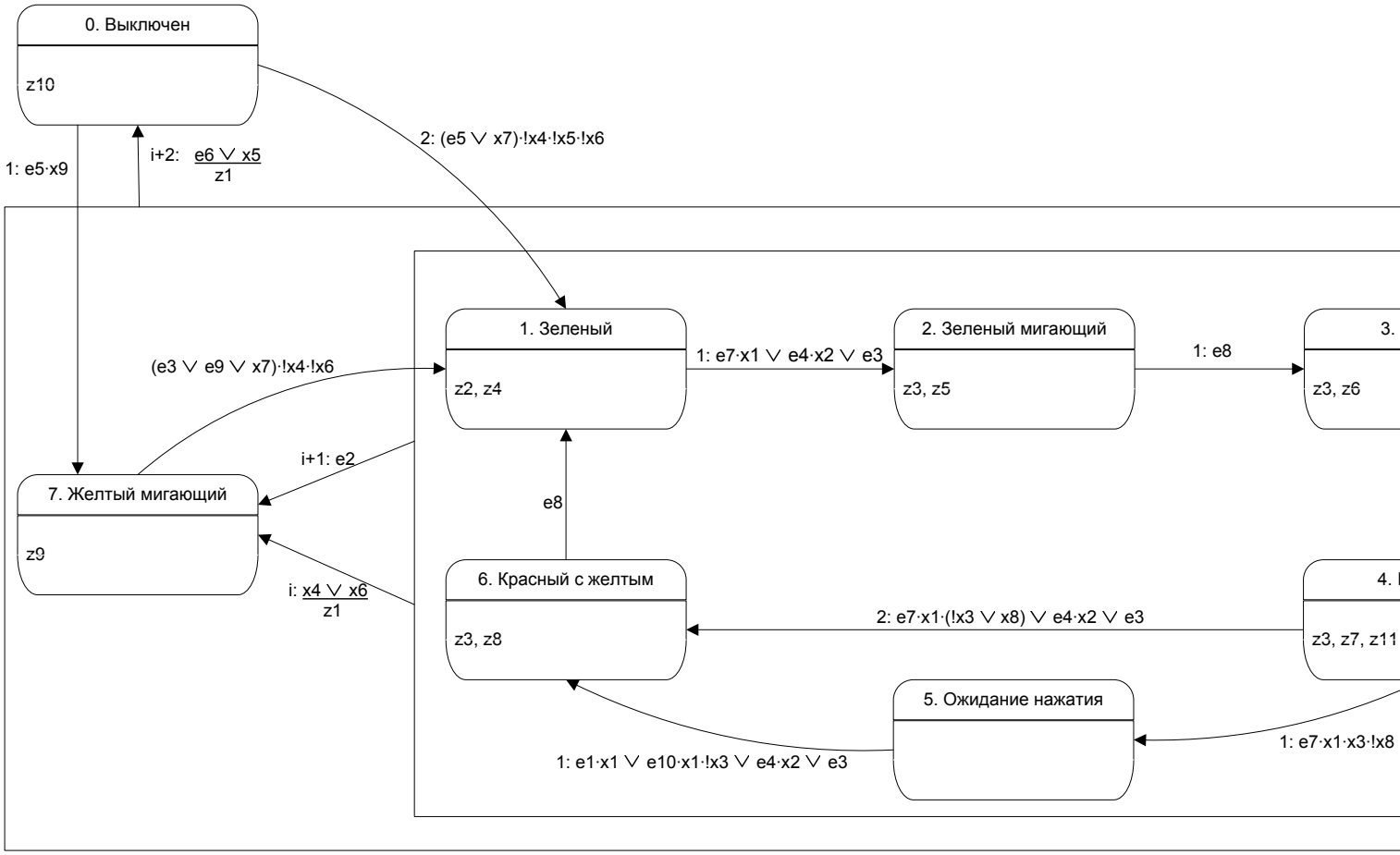

Рис. 4. Граф переходов автомата управления светофором

Обратим внимание, что в приведенном графе переходов у групповых дуг приоритеты обозначены как *i*, *i+1* и *i+2*. Это объясняется тем, что число исходящих дуг у вершин, входящих в одну группу, различно.

4. Текст программы

```
const char logname[] = "log.txt";

/*****************************************************************************
                                ИНТЕРФЕЙС КЛАССА
*****************************************************************************/
class CAutomat : public CWnd
{
public:
        void auto_mode();               // Включен автоматическмй режим
        void manual_mode();             // Включен ручной режим
        void blinking_mode();           // Включен режим мигания
        void switch_button();           // Включена/выключена кнопка перехода
        void power_button();            // Включена/выключена кнопка питания
        void switch_lights();           // Нажата кнопка переключения огней

        void brokenred();               // Перегорел красный
        void brokengreen();             // Перегорел зеленый
        void brokenyellow();            // Перегорел желтый
        void fixedred();                // Починен красный
        void fixedgreen();              // Починен зеленый
        void fixedyellow();             // Починен желтый

        void buttonpressed();           // Нажата кнопка перехода

        void long_time_elapsed();       // Истекла задержка долгого таймера
        void short_time_elapsed();      // Иистекла задержка короткого таймера

        CAutomat();
        ~CAutomat();

private:
// Автоматные переменные
        int     cur_mode;
        bool    blinking_mode;
        bool    button_on;
        bool    broken_red;
        bool    broken_yellow;
        bool    broken_green;
        bool    broken;
        bool    buttonpressed;
        int             y0;

// Вспомогательные переменные
        bool            LOGGING;
        ofstream        logfile;
        bool            on;

        // Возможные значения cur_mode
        enum modes {AUTO, MANUAL, BLINKING};

        // Идентификаторы таймеров
        enum timers {LTIMER=1, STIMER=2};

// Функция автомата
        void A0(int);

// Входные переменные
        bool x1();
        bool x2();
        bool x3();
        bool x4();
        bool x5();
        bool x6();
        bool x7();
        bool x8();
        bool x9();

// Выходные воздействия
        void z1();
        void z2();
        void z3();
        void z4();
        void z5();
        void z6();
        void z7();
        void z8();
        void z9();
        void z10();
        void z11();

// Методы для лога
```

```
        void log(char*, char);
        void log_begin(int, int);
        void log_end(int);
        void log_input(int, char*, bool);
        void log_output(int, char*);
        void log_err(int);
        void log_trans(int, int);


// Управление огнями светофора
        void light(states);
};

CAutomat::CAutomat()
{
        y0 = 0;
        on = false;

        LOGGING = true;
        cur_mode = AUTO;
        button_on = false;
        broken_red = false;
        broken_yellow = false;
        broken_green = false;
        broken = false;
        buttonpressed = false;

        if(LOGGING)
        {
                logfile.open(logname);
        }
}

CAutomat::~CAutomat()
{
        if(LOGGING)
        {
                logfile.close();
        }
}


//Реализация функции автомата

void CAutomat::A0(int e)
{
        int yold = y0;

        if(LOGGING)
                log_begin(y0, e);

        switch(y0)
        {
                case 0:
                        if(e == 5 && x9())                                      {       y0 = 7; }
                        else if((e == 5 || x7()) && !x4() && !x5() && !x6())    {       y0 = 1; }
                        break;
                case 1:
                        if(e == 7 && x1() || e == 4 && x2() || e == 3)          {       y0 = 2; }
                        if(x4() || x6())                                        { z1(); y0 = 7; }
                        if(e == 2)                                              {       y0 = 7; }
                        if(e == 6 || x5())                                      { z1(); y0 = 0; }
                        break;
                case 2:
                        if(e == 8)                                              {       y0 = 3; }
                        if(x4() || x6())                                        { z1(); y0 = 7; }
                        if(e == 2)                                              {       y0 = 7; }
                        if(e == 6 || x5())                                      { z1(); y0 = 0; }
                        break;
                case 3:
                        if(e == 8)                                              {       y0 = 4; }
                        if(x4() || x6())                                        { z1(); y0 = 7; }
                        if(e == 2)                                              {       y0 = 7; }
                        if(e == 6 || x5())                                      { z1(); y0 = 0; }
                        break;
                case 4:
                        if(e == 7 && x1() && x3() && !x8())                     {       y0 = 5; }
                        if(e == 7 && x1() && (!x3() || x8())
                                || e == 4 && x2() || e == 3)                    {       y0 = 6; }
                        if(x4() || x6())                                        { z1(); y0 = 7; }
                        if(e == 2)                                              {       y0 = 7; }
                        if(e == 6 || x5())                                      { z1(); y0 = 0; }
                        break;
```

```
		case 5:
			if(e == 1 && x1() || e == 10 && x1() && !x3()
				|| e == 4 && x2() || e == 3)		{	y0 = 6; }
			if(x4() || x6())			{ z1();	y0 = 7; }
			if(e == 2)				{	y0 = 7; }
			if(e == 6 || x5())			{ z1();	y0 = 0; }
			break;

		case 6:
			if(e == 8)				{	y0 = 1; }
			if(x4() || x6())			{ z1();	y0 = 7; }
			if(e == 2)				{	y0 = 7; }
			if(e == 6 || x5())			{ z1();	y0 = 0; }
			break;
		case 7:
			if((e == 3 || e == 9 || x7()) && !x4() && !x6())	{	y0 = 1; }
			if(e == 6 || x5())			{ z1();	y0 = 0; }
			break;
		default:
			if(LOGGING)
				log_err(y0);
	}

	if(y0 != yold)
	{
		if(LOGGING)
			log_trans(y0, yold);

		switch(y0)
		{
			case 0:
				{ z10(); }
				break;
			case 1:
				{ z2(); z4(); }
				break;
			case 2:
				{ z3();z5(); }
				break;
			case 3:
				{ z3(); z6(); }
				break;
			case 4:
				{ z2(); z7(); z11();}
				break;
			case 5:
				break;
			case 6:
				{ z3(); z8(); }
				break;
			case 7:
				{ z9(); }
				break;
			default:
				if(LOGGING)
					log_err(y0);
		}
	}

	if(LOGGING)
		log_end(y0);
}

/*****************************************************************************
			ВХОДНЫЕ ПЕРЕМЕННЫЕ
*****************************************************************************/

bool CAutomat::x1()
{
	bool res = cur_mode == AUTO;
	if(LOGGING)
		log_input(1, "Автоматический режим", res);
	return res;
}



bool CAutomat::x2()
{
	bool res = cur_mode == MANUAL;
	if(LOGGING)
		log_input(2, "Ручной режим", res);
	return res;
}
```

```
bool CAutomat::x3()
{
	bool res = button_on;
	if(LOGGING)
		log_input(3, "Кнопка включена", res);
	return res;
}

bool CAutomat::x4()
{
	bool res = broken_red;
	if(LOGGING)
		log_input(4, "Красный перегорел", res);
	return res;
}

bool CAutomat::x5()
{
	bool res = broken_yellow;
	if(LOGGING)
		log_input(5, "Желтый перегорел", res);
	return res;
}

bool CAutomat::x6()
{
	bool res = broken_green;
	if(LOGGING)
		log_input(6, "Зеленый перегорел", res);
	return res;
}

bool CAutomat::x7()
{
	bool res = broken;
	if(LOGGING)
		log_input(7, "Аварийное мигание", res);
	return res;
}

bool CAutomat::x8()
{
	bool res = buttonpressed;
	if(LOGGING)
		log_input(8, "Нажата кнопка 'Переход'", res);
	return res;
}

bool CAutomat::x9()
{
	bool res = cur_mode == BLINKING;
	if(LOGGING)
		log_input(9, "Режим мигания", res);
	return res;
}


/*****************************************************************************
                        ОБРАБОТЧИКИ СОБЫТИЙ
*****************************************************************************/


void CAutomat::buttonpressed()
{
	buttonpressed = true;
	A0(1);
}

void CAutomat::blinking_mode()
{
	cur_mode = BLINKING;
	A0(2);
}

void CAutomat::auto_mode()
{
	A0(3);
	cur_mode = AUTO;
}

void CAutomat::switch_lights()
{
	A0(4);
```

```
}

void CAutomat::switch_button()
{
	button_on = !button_on;
	if(!button_on)
		A0(10);
}

void CAutomat::power_button()
{
	if(on)
	{
		on = false;
		A0(6);
	}
	else
	{
		on = true;
		A0(5);
	}
}

void CAutomat::long_time_elapsed()
{
	A0(7);
}

void CAutomat::short_time_elapsed()
{
	A0(8);
}

void CAutomat::manual_mode()
{
	A0(9);
	cur_mode = MANUAL;
}

/*****************************************************************************************
                                ВЫХОДНЫЕ ВОЗДЕЙСТВИЯ
*****************************************************************************************/

void CAutomat::z1()
{
	broken = true;
	if(LOGGING)
		log_output(1, "Аварийное состояние");
}



void CAutomat::z2()
{
	SetTimer(LTIMER, 5000, NULL);
	if(LOGGING)
		log_output(2, "Установить 'долгий' таймер");
}

void CAutomat::z3()
{
	SetTimer(STIMER, 3000, NULL);
	if(LOGGING)
		log_output(3, "Установить 'короткий' таймер");
}

void CAutomat::z4()
{
	light(GREEN);
	if(LOGGING)
		log_output(4, "Включить зеленый");
}

void CAutomat::z5()
{
	light(BLINKGREEN);
	if(LOGGING)
		log_output(5, "Включить зеленый мигающий");
}

void CAutomat::z6()
{
	light(YELLOW);
	if(LOGGING)
```

```
		log_output(6, "Включить желтый");
}

void CAutomat::z7()
{
	light(RED);
	if(LOGGING)
		log_output(7, "Включить красный");
}

void CAutomat::z8()
{
	light(REDYELLOW);
	if(LOGGING)
		log_output(8, "Включить красный и желтый");
}

void CAutomat::z9()
{
	light(BLINKYELLOW);
	if(LOGGING)
		log_output(9, "Включить желтый мигающий");
}

void CAutomat::z10()
{
	light(OFF);
	if(LOGGING)
		log_output(10, "Погасить огни");
}

void CAutomat::z11()
{
	buttonpressed = false;
	if(LOGGING)
		log_output(11, "Переход по кнопке");
}




/*****************************************************************************************
                                ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ
*****************************************************************************************/

void CAutomat::brokenred()
{
	broken_red = true;
}

void CAutomat::brokenyellow()
{
	broken_yellow = true;
}

void CAutomat::brokengreen()
{
	broken_green = true;
}

void CAutomat::fixedred()
{
	broken_red = false;
}

void CAutomat::fixedyellow()
{
	broken_yellow = false;
}

void CAutomat::fixedgreen()
{
	broken_green = false;
}

void CAutomat::light(states what)
{
	(((CTrafficLightsDlg *)((CTrafficLightsApp *)AfxGetApp())->m_pMainWnd))-
>m_Canvas).SetMode(what);
}

void CAutomat::log_begin(int state, int event)
```

```
{
	logfile << "{ Автомат A0 начал свою работу в состоянии " << state;
	logfile << " с событием e" << event << endl;
}

void CAutomat::log_end(int state)
{
	logfile << "Автомат A0 закончил свою работу в состоянии " << state;
	logfile << "}" << endl;
}

void CAutomat::log_input(int id, char* comment, bool result)
{
	char* res;
	res = result ? "ДА" : "НЕТ";

	logfile << " I ";
	logfile << "x" << id << ": " << comment << "? - " << res << endl;
}

void CAutomat::log_output(int id, char* comment)
{
	logfile << " O ";
	logfile << "z" << id << ": " << comment << endl;
}

void CAutomat::log_err(int state)
{
	logfile << " E ";
	logfile << "Неопознанное состояние: " << state << endl;
}

void CAutomat::log_trans(int to, int from)
{
	logfile << "Автомат A0 перешел из состояния " << from;
	logfile << " в состояние " << to << "}{" << endl;
}

void CAutomat::OnTimer(UINT nIDEvent)
{
	KillTimer(nIDEvent);
	if(nIDEvent == LTIMER)
		long_time_elapsed();
	if(nIDEvent == STIMER)
		short_time_elapsed();
}
```

5. Протокол работы светофора

Ниже представлен протокол одного цикла работы программы «от зеленого до зеленого». В этом протоколе символ *I* соответствует входным переменным, символ *O* – выходным воздействиям, фигурные скобки *{}* выделяют обработку автоматом события.

```
{ Автомат A0 начал свою работу в состоянии 0 с событием e5
 I x9: Режим мигания? - НЕТ
 I x4: Красный перегорел? - НЕТ
 I x5: Желтый перегорел? - НЕТ
 I x6: Зеленый перегорел? - НЕТ
Автомат A0 перешел из состояния 0 в состояние 1}{
 O z2: Установить 'долгий' таймер
 O z4: Включить зеленый
Автомат A0 закончил свою работу в состоянии 1}
{ Автомат A0 начал свою работу в состоянии 1 с событием e7
 I x1: Автоматический режим? - ДА
 I x4: Красный перегорел? - НЕТ
 I x6: Зеленый перегорел? - НЕТ
 I x5: Желтый перегорел? - НЕТ
Автомат A0 перешел из состояния 1 в состояние 2}{
 O z3: Установить 'короткий' таймер
 O z5: Включить зеленый мигающий
Автомат A0 закончил свою работу в состоянии 2}
{ Автомат A0 начал свою работу в состоянии 2 с событием e8
 I x4: Красный перегорел? - НЕТ
 I x6: Зеленый перегорел? - НЕТ
 I x5: Желтый перегорел? - НЕТ
Автомат A0 перешел из состояния 2 в состояние 3}{
 O z3: Установить 'короткий' таймер
 O z6: Включить желтый
Автомат A0 закончил свою работу в состоянии 3}
{ Автомат A0 начал свою работу в состоянии 3 с событием e8
 I x4: Красный перегорел? - НЕТ
 I x6: Зеленый перегорел? - НЕТ
 I x5: Желтый перегорел? - НЕТ
Автомат A0 перешел из состояния 3 в состояние 4}{
 O z2: Установить 'долгий' таймер
 O z7: Включить красный
 O z11: Переход по кнопке
Автомат A0 закончил свою работу в состоянии 4}
{ Автомат A0 начал свою работу в состоянии 4 с событием e7
 I x1: Автоматический режим? - ДА
 I x3: Кнопка включена? - НЕТ
 I x1: Автоматический режим? - ДА
 I x3: Кнопка включена? - НЕТ
 I x1: Автоматический режим? - ДА
 I x3: Кнопка включена? - НЕТ
 I x4: Красный перегорел? - НЕТ
 I x6: Зеленый перегорел? - НЕТ
 I x5: Желтый перегорел? - НЕТ
Автомат A0 перешел из состояния 4 в состояние 6}{
 O z3: Установить 'короткий' таймер
 O z8: Включить красный и желтый
Автомат A0 закончил свою работу в состоянии 6}
{ Автомат A0 начал свою работу в состоянии 6 с событием e8
 I x4: Красный перегорел? - НЕТ
 I x6: Зеленый перегорел? - НЕТ
 I x5: Желтый перегорел? - НЕТ
Автомат A0 перешел из состояния 6 в состояние 1}{
 O z2: Установить 'долгий' таймер
 O z4: Включить зеленый
Автомат A0 закончил свою работу в состоянии 1}
```