Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра “Компьютерные технологии”

Ю.А. Альшевский, М.Г. Раер, А.А. Шалыто

Механизм обмена сообщениями для параллельно работающих автоматов (на примере системы управления турникетом)

Объектно-ориентированное программирование с явным выделением состояний

Проектная документация

Разработано в рамках
“Движения за открытую проектную документацию”
http://is.ifmo.ru

Работа победила в общегородском межвузовском конкурсе курсовых и дипломных работ на премии “Компьютер-центра КЕЙ” (Санкт-Петербург) в номинации ”Компьютерные технологии в электронике и автоматике”. //Кей Центр. 2003. №6
(http://www.key.ru/about.php?p=10&page=release2б)

Санкт-Петербург
2003

Содержание

Введение

Для алгоритмизации задач логического управления в работе [1] была предложена SWITCH-технология. В этой работе автоматы, работающие параллельно, не рассматривались.

В настоящей работе предложено использовать параллельно работающие автоматы (параллельные автоматы), которые взаимодействуют между собой за счет обмена сообщениями. Это напоминает такую область параллельных вычислений, как распределенное программирование, в которой процессы также взаимодействуют посредством обмена сообщениями [2].

Данная работа призвана рассмотреть вариант расширения идеологии объектно-ориентированного программирования с явным выделением состояний [3, 4] на случай наличия параллельно работающих автоматов. Это выполняется на примере создания системы управления турникетом, контролирующим проход пассажиров.

При проектировании систем параллельных автоматов возникают три задачи:

- взаимодействие автоматов;
- синхронизация автоматов;
- протоколирование их работы.

Эти задачи решаются в настоящей работе. Отметим, что предлагаемый подход наследует все достоинства SWITCH-технологии. К ним можно отнести централизацию логики управления и изоморфизм между кодом и графом переходов автомата. Это облегчает чтение, модификацию, документирование и отладку программ.

Для демонстрации работы системы управления был создан визуализатор, имитирующий работу турникета.

Программный код написан на языке *Visual C++* с использованием библиотеки классов *Microsoft Foundation Classes* [5] и функционирует в одном потоке под управлением операционной системы *Microsoft Windows*.

С целью облегчения процесса отладки программы и детального ознакомления с механизмом взаимодействия *параллельных* автоматов выполняется протоколирование работы *каждого* автомата.

Данная работа выполнена в рамках движения за *открытую проектную документацию* (http://is.ifmo.ru). Открытыми являются также и исходные файлы.

1. Описание системы

Целью работы является создание программы управления турникетом, обеспечивающим контроль прохода пассажиров. Турникет управляется с помощью системы взаимодействующих, параллельно работающих автоматов.

В системе используются следующие устройства:

- дверь;
- пять фотодатчиков;
- сигнализация (индикатор несанкционированного прохода);
- устройство считывания магнитных карт (МК) и/или любые другие устройства, проверяющие возможность прохода;
- сервер;
- центр управления;
  - кнопка “Ручной контроль” (РК);
  - кнопка “Открытие/закрытие двери” (О/З);
  - лампа “Ручной контроль”;
- панель турникета;
  - лампа “Проход разрешен”;
  - лампа “Проход запрещен”;
- таймеры;
  - таймер времени работы сигнализации;
  - таймер времени ожидания прохода тележки;
  - таймер времени горения лампы “Проход запрещен”.

Внешний вид визуализатора системы изображен на рис. 1.

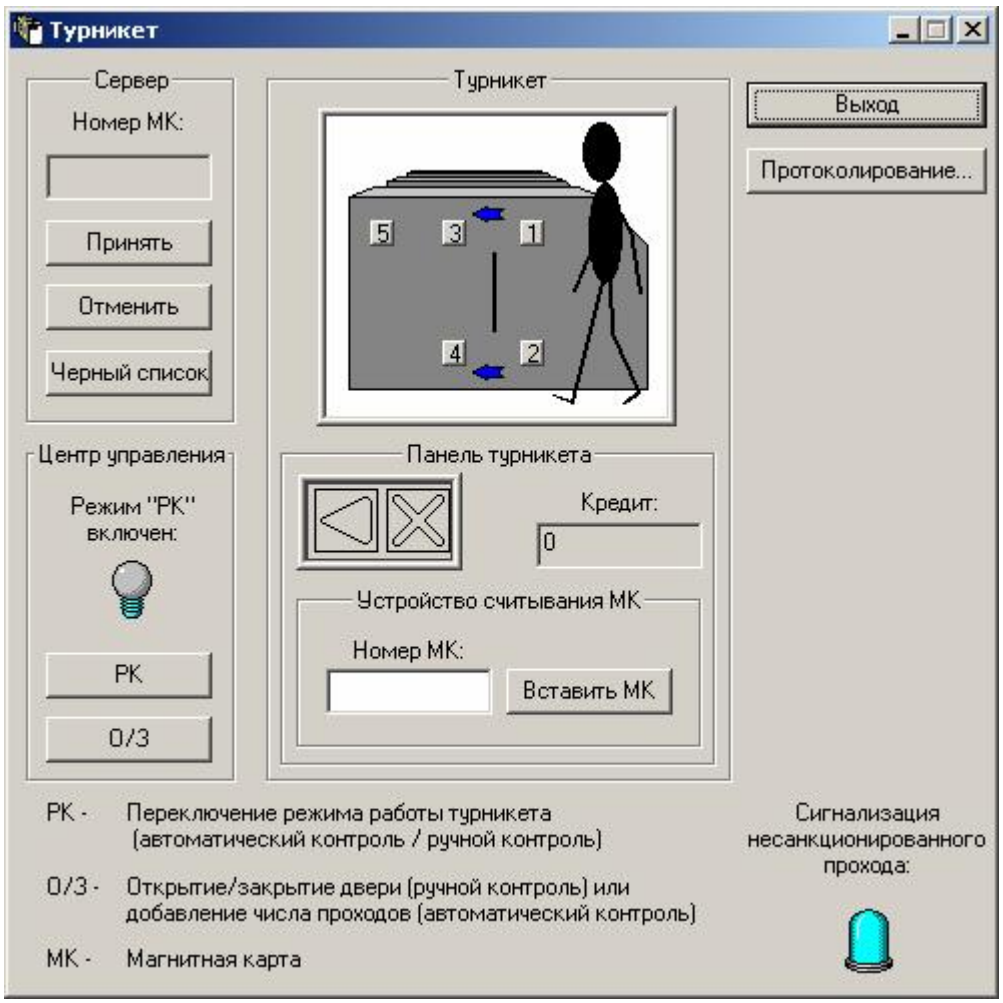

Рис. 1. Внешний вид визуализатора системы

Рабочая область окна разбита на три группы:

- сервер;
- центр управления;
- турникет.

1.1. Сервер

Нажатие на кнопки “Принять”, “Отменить” и “Черный список” соответствуют установке значений внешних переменных x12, x13 и x14, которые в дальнейшем используются в проекте. В текстовом поле “Номер МК” отображается номер магнитной карты, полученной сервером от турникета.

1.2. Центр управления

Нажатие на кнопки “РК” (“Ручной контроль”) и “О/З” (“Открытие/закрытие”) соответствует установке значений внешних переменных x8 и x10.

1.3. Турникет

Нажатие на кнопки “1”, “2”, “3”, “4”, “5” на изображении турникета соответствует срабатыванию фотодатчиков с теми же номерами и устанавливает значения внешних переменных x1, x2, x3, x4, x5. Воздействие на эти кнопки можно произвести как с помощью мыши, так и с помощью кнопок с соответствующими значениями на цифровой панели клавиатуры.

Под турникетом на рис. 1 расположена его панель. Она содержит лампы “Проход разрешен” и “Проход запрещен” , текстовое поле “Кредит” (число оплаченных поездок) и группу “Устройство считывания МК”, содержащую текстовое поле “Номер МК” и кнопку “Вставить МК”

Нажатие на кнопку “Вставить МК” посылает сообщение m11. При этом в текстовом поле “Номер МК” должен содержаться номер вставляемой карты. Отправка магнитной карты в бункер отображается стиранием содержимого текстового поля “Номер МК”

1.4. Протоколирование

При нажатии на кнопку “Протоколирование…” открывается диалоговое окно “Опции протоколирования”, внешний вид которого изображен на рис. 2.

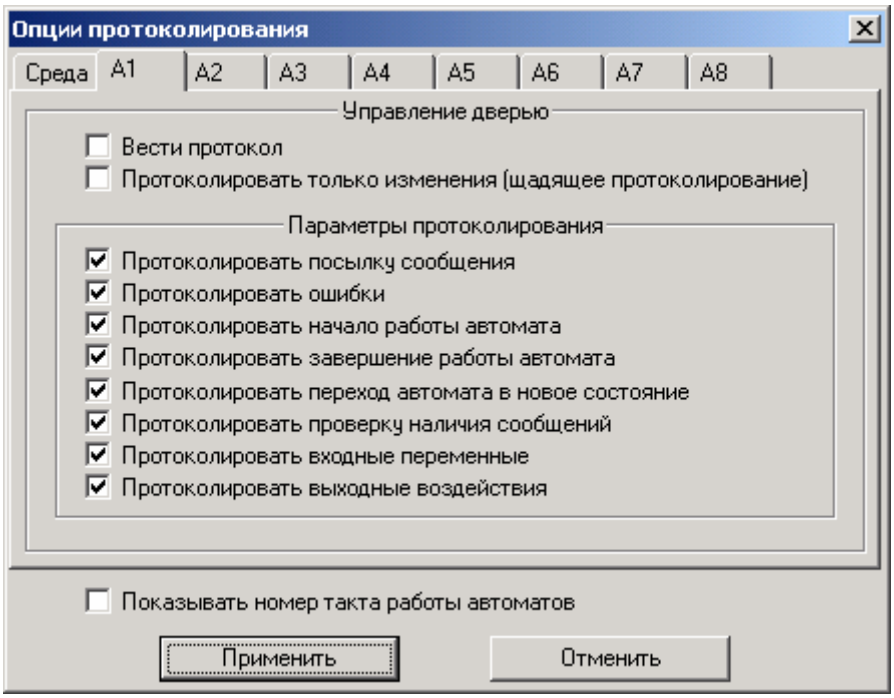

Рис. 2. Внешний вид диалогового окна "Опции протоколирования"

Выбирая закладки и устанавливая и сбрасывая флажки, можно управлять параметрами протоколирования соответствующего автомата или среды[1].

[1] Понятие *среда* будет определено далее.

2. Функциональные возможности системы

Турникет оснащен дверью, что делает возможным проход пассажиров с багажом (сумками, тележками и т.д.). Системой ведется контроль несанкционированного прохода. Для этого в турникет встроены фотодатчики. Если имеет место несанкционированный проход, то срабатывает сигнализация и происходит закрытие двери. Турникет накапливает число оплаченных поездок и разрешает проход сразу после выхода предыдущего пассажира.

Турникет поддерживает два режима работы:

- “Автоматический контроль” (проход контролируется устройствами, проверяющими возможность прохода);
- “Ручной контроль” (проход контролируется человеком).

В режиме автоматического контроля пропуск пассажиров может происходить либо по магнитной карте, либо по разрешению контролера (контролер нажимает кнопку “Открытие/закрытие двери”). Каждое устройство, проверяющее возможность прохода, управляется отдельным специализированным модулем. Поэтому список таких устройств можно расширять, добавляя, например, устройство считывания смарт-карт, приема жетонов и т.д.

При нажатии на кнопку “Ручной контроль” турникет дожидается прохода всех уже оплативших проезд пассажиров и только после этого переходит в режим “Ручной контроль”. При этом на панели центра управления загорается лампа “Ручной контроль”. В этом режиме дверь открывается/закрывается при нажатии на кнопку “Открытие/закрытие”.

Для более подробного ознакомления с функциональностью системы необходимо изучение схем связей и графов переходов автоматов, так как **словами сложное поведение системы описать практически невозможно.**

3. Описание подхода

Во введении отмечено, что при использовании параллельных автоматов необходимо решить три задачи. Опишем решение каждой их них.

3.1. Взаимодействие автоматов

Для решения этой задачи предлагается использовать *модель* их взаимодействия, в которой параллельные автоматы связываются друг с другом, используя *механизм обмена сообщениями*. Это дополняет методы взаимодействия автоматов, которые применялись в SWITCH-технологии и значительно упрощает процесс синхронизации параллельных автоматов.

В предлагаемой модели по сравнению со SWITCH-технологией не используются *события*, а вместо них применяются *сообщения*. За один такт работы, так как автоматы работают параллельно, автомату может быть послано более одного сообщения. Все полученные сообщения запоминаются в массивах и учитываются при определении нового состояния. Символы, соответствующие сообщениям, также как и входные переменные могут образовывать булевы формулы, помечающие дуги графов переходов.

Под *средой* будем понимать совокупность объектов управления.

В предлагаемой модели произведено изменение структуры автоматов с целью обеспечения приема и передачи сообщений. Это позволяет среде и автоматам посылать сообщения, как конкретному автомату, так и всем автоматам сразу. Последнее можно представить как подачу сигнала в общую шину сообщений или передачу сообщения в эфир (рис. 3).

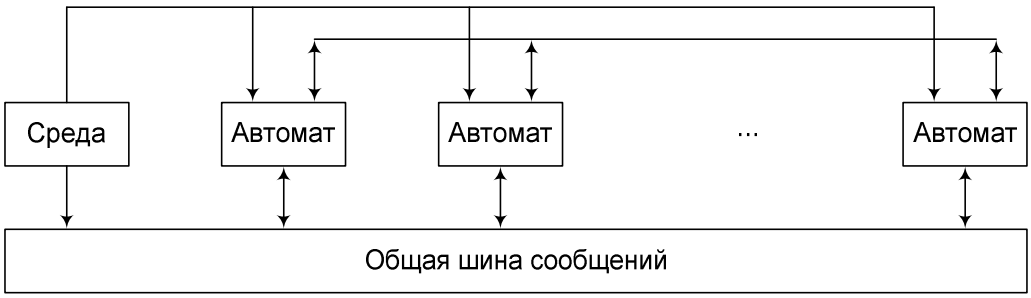

Рис. 3. Схема обмена сообщениями

Среда и каждый из автоматов могут посылать сообщения в общую шину сообщений (нижняя часть рисунка), из которой их читают автоматы. Среда и каждый автомат также могут посылать сообщения непосредственно конкретному автомату (верхняя часть рисунка).

На графах переходов действия, соответствующие посылке сообщения с номером i автомату Ak, будем обозначать zkmi, а в общую шину – zmi. Например, обозначение z1m3 соответствует посылке сообщения с номером 3 (m3) автомату с номером 1 (A1), а обозначение zm9 - посылке сообщения с номером 9 (m9) в общую шину.

Каждый автомат для хранения сообщений, получаемых от автоматов и среды, *содержит массив*, индексами которого являются номера сообщений, а значения элементов массива определяются наличием соответствующих сообщений.

Для хранения сообщений, посылаемых в общую шину, введен еще один массив, имеющий такую же организацию.

Отметим, что при такой реализации модели термины *вложенный автомат* и *вызываемый автомат* становятся неразличимыми, так как каждый автомат вызывается со своим набором сообщений, посланных ему на предыдущем такте. Авторы предлагают в данном случае использовать термин *вложенный автомат*.

3.2. Синхронизация автоматов

При проектировании систем параллельно работающих (даже в одном потоке) автоматов возникает задача их синхронизации, связанная с тем, что без синхронизации один из автоматов может использовать значения переменных, которые были изменены другими автоматами на этом же такте, а не значения, запомненные до изменения. Это может привести к неправильной работе системы.

Задачу синхронизации автоматов при условии их параллельности предлагается решать разделением функции, реализующей автомат, на две функции:

- переход-действие **TA** (Transition-Action);
- обновление **U** (Update).

Первая функция создается для каждого автомата. Она ***сначала*** осуществляет выходные воздействия, в текущем состоянии и на переходе, ***затем*** определяет номер нового состояния автомата и осуществляет выходные воздействия в новом состоянии.

Вторая функция создается одна для всех автоматов. Она обеспечивает обновление значения внутренней переменной, кодирующей состояния автомата, а также массив поступающих ему сообщений.

Таким образом, в общем цикле работы автоматов для их синхронизации сначала должны вызываться функции *переход-действие* для ***всех*** автоматов, ***работающих параллельно***, а затем функцию *обновление* для ***всех*** автоматов. После этого необходимо обновить массив, предназначенный для хранения сообщений, посылаемых в общую шину.

Идея разбиения автоматной функции для синхронизации параллельных автоматов на несколько была предложена в [6]. Однако из-за другой модели взаимодействия параллельных автоматов в этой работе автомат предлагалось реализовывать не двумя, а тремя функциями.

3.3. Протоколирование работы автоматов

В отличие от SWITCH-технологии, при использовании которой протоколирование работы автоматов обычно ведется в одном файле, при протоколировании параллельных автоматов целесообразно вести протоколы для каждого автомата в отдельности. Кроме указанных протоколов имеет смысл вести протоколирование сообщений, посылаемых средой. Таким образом, создается по одному файлу протокола для каждого автомата и один файл протокола, в который записываются сообщения, посланные средой.

Файл каждого протокола имеет расширение *log* и состоит из строк следующей структуры:

<номер шага работы><символ типа протоколирования><протоколируемый текст>

Используются следующие символы для задания типа протоколирования:

{ - начало работы автомата;
T – переход автомата в новое состояние;
E – ошибка (неизвестный номер состояния);
? – проверка наличия сообщения;
> - вызов функции входной переменной;
< - вызов функции выходной переменной;
! – посылка сообщения;
} – завершение работы автомата.

В программе предусмотрено включение/выключение каждого типа протоколирования для каждого автомата в отдельности, включение/выключение протоколирования автомата в целом и включение/выключение режима *щадящего протоколирования*.

Необходимость последнего режима связана с тем, что при протоколировании работы параллельных автоматов возникает проблема *лавинного роста* файла протокола. Дело в том,

что при данном подходе автоматы работают постоянно, пока запущено приложение. Если при этом включено протоколирование, то также постоянно происходит запись в файл протокола. Во избежание этого и был введен режим *щадящего протоколирования*.

В этом режиме запись в файл ведется только в случае, если текущий шаг работы автомата чем-то отличается от предыдущего. В противном случае в файл протокола записывается многоточие, а все последующие одинаковые шаги не протоколируются.

В целях ускорения сравнения текста протокола, полученного на текущем и предыдущих шагах, программой ведется *индексирование* текста протокола – параллельное протоколирование в памяти в сжатом виде.

Например, если на такте с номером 189 автомат “Управление дверью” (A1) перешел из состояния с номером “0” в состояние с номером “1”, то в протокол этого автомата добавится строка

“ 189 T A1(Управление дверью) перешел из состояния 0 (Дверь закрыта) в 1 (Дверь открыта). ”

При этом в индекс, соответствующий этому протоколу, добавится строка

“AT[0][1]”

Так как протоколирование каждого автомата ведется в отдельном файле, то фиксировать в индексе номер автомата бессмысленно.

Щадящий режим реализован автоматом, схема связей и граф переходов которого изображены на рис. 4, 5.

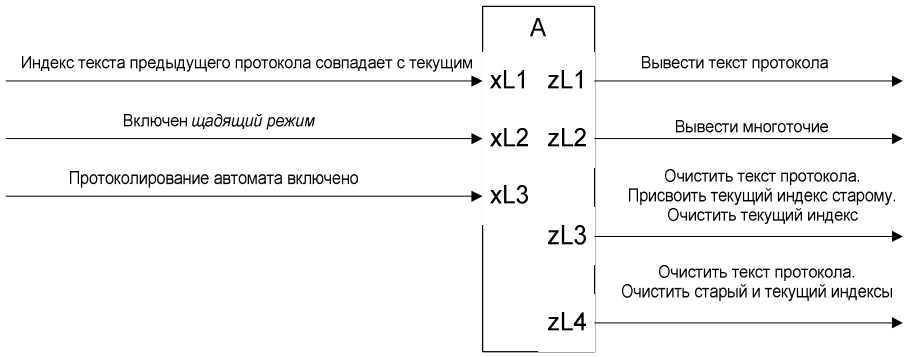

Рис. 4. Схема связей автомата протоколирования

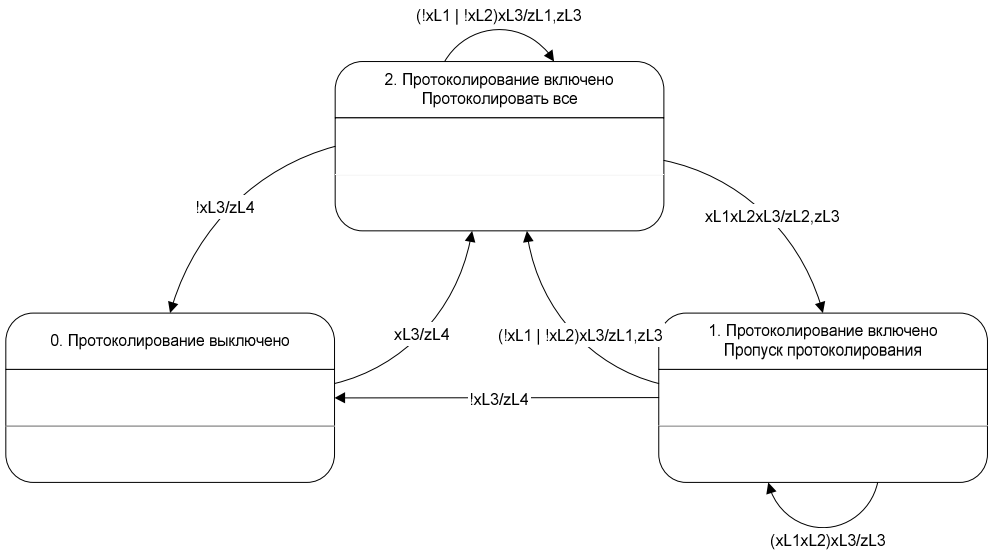

Рис. 5. Граф переходов автомата протоколирования

4. Структура программы

Каждый автомат представляет собой отдельный класс, унаследованный от класса *CAutomaton*, который реализует абстрактный автомат. Класс *ClogStr* реализует структуру с возможностью протоколирования. Поэтому класс *CAutomaton* унаследован от *CLogStr*. В виду того, что протоколирование ведется не только для автоматов, но и для среды, у класса *CLogStr* есть экземпляр, обеспечивающий протоколирование среды. Диаграмма классов системы управления турникетом изображена на рис. 6.

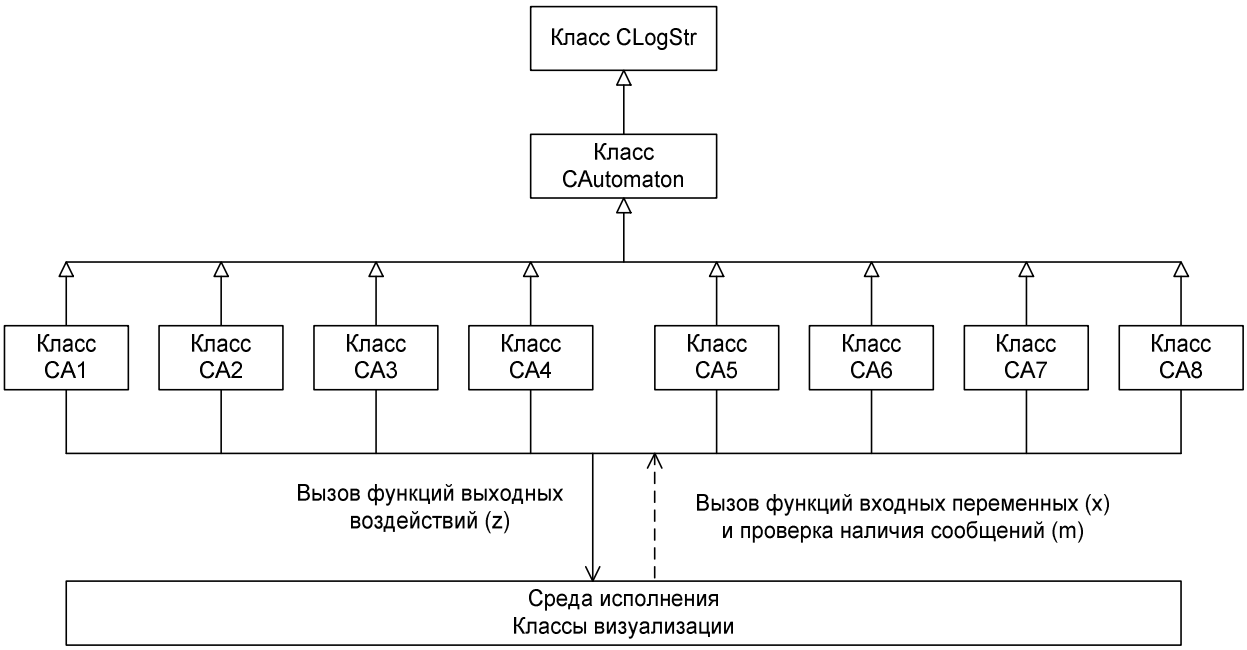

Рис. 6. Диаграмма классов

Ниже приводятся листинги файлов с описаниями классов *CLogStr* и *CAutomaton*. Листинги с реализацией методов этих классов приведены в приложении. В настоящем разделе также приведен шаблон функции *переход-действие*, функция *обновление* автомата и цикл работы автоматов.

4.1. Листинг описания класса “CLogStr”

```
/*================================[ REDEFINITION DEFENCE ]================*/
#ifndef __RA_CLogStr__
#define __RA_CLogStr__

/*================================[ IMPORT DECLARATIONS ]=================*/
#include <stdio.h>

// Число отработанных тактов автоматов
// Используется для протоколирования
extern long int Counter;

// Символ протоколирования...
enum
{
   M_LOGGING_C          = '?', // ...проверки наличия сообщений
   Z_LOGGING_C          = '<', // ...выходных воздействий
   X_LOGGING_C          = '>', // ...проверки входных переменных
   M_SEND_LOGGING_C     = '!', // ...посылки сообщения
   A_BEGINS_LOGGING_C   = '{', // ...начала работы автомата
   A_TRANS_LOGGING_C    = 'T', // ...перехода автомата в новое состояние
   A_ENDS_LOGGING_C     = '}', // ...завершения работы автомата
   A_ERRORS_LOGGING_C   = 'E'  // ...ошибок перехода в несуществующее состояние
};
```

```
/*****************************************************************************\
   CLASS.........: CLogStr
   DESCRIPTION...: Структура для протоколирования
\*****************************************************************************/
class CLogStr
{
protected:

   char number;       // Номер протоколируемой структуры
   char name[256];    // Имя протоколируемой структуры
   FILE *log_file;    // Указатель на дескриптор файла для протоколирования
   char LOGGING;      // Включено ли протоколирование?

   char buffer[5000]; // Буфер для записи протокола
   char *index_old;   // Буфер для старого индекса протокола
   char *index_cur;   // Буфер для текущего индекса протокола

   int y;             // Переменная, кодирующая состояние автомата протоколирования

   // Функции входных переменных автомата протоколирования
   char xL1(); // Индекс текущего протокола совпадает с текущим
   char xL2(); // Включен щадящий режим
   char xL3(); // Протоколирование автомата включено

   // Функции выходных воздействий автомата протоколирования
   void zL1(); // Вывести текст протокола
   void zL2(); // Вывести многоточие
   void zL3(); // Очистить текст протокола. Присвоить текущий индекс старому. Очистить текущий
               индекс
   void zL4(); // Очистить текст протокола. Очистить старый и текущий индексы

public:

   // Включено ли протоколирование...
   char M_LOGGING;         // ...проверки наличия сообщений
   char Z_LOGGING;         // ...выходных воздействий
   char X_LOGGING;         // ...проверки входных переменных
   char M_SEND_LOGGING;    // ...посылки сообщений
   char A_BEGINS_LOGGING;  // ...начала работы автомата
   char A_TRANS_LOGGING;   // ...перехода автомата в новое состояние
   char A_ENDS_LOGGING;    // ...завершения работы автомата
   char A_ERRORS_LOGGING;  // ...ошибок перехода в несуществующее состояние
   char NO_REPEAT_LOGGING; // Включен ли щадящий режим?


   CLogStr(const char *aname, char anumber);
   ~CLogStr();

   void SetLogging();      // Включить протоколирование
   void ReSetLogging();    // Выключить протоколирование
   char Is_Logging();      // Включено ли протоколирование?

   void TA_Log();          // Функция автомата протоколирования

   char *const GetName();  // Функция возвращает имя структуры для протоколирования
   char GetNumber();       // Функция возвращает номер структуры для протоколирования


   // Протоколирование проверки наличия сообщения
   void MLogging(int msgNum, char b);

   // Протоколирование проверки входных переменных
   void XLogging (int xNum, char b);

   // Протоколирование выходных воздействий
   void ZLogging (int zNum);

   // Протоколирование начала работы автомата
   void ABeginsLogging (int y, char *stName);

   // Протоколирование завершения работы автомата
   void AEndsLogging (int y, char *stName);

   // Протоколирование перехода автомата в новое состояние
   void ATransLogging(int y1, int y2, char *stName1, char *stName2);

   // Протоколирование ошибок перехода в несуществующее состояние
   void AErrorsLogging(int y);

   // Протоколирование посылки сообщения в общую шину
   static void MSendPubLogging (CLogStr *a, int msgNum);
```

```
   // Протоколирование посылки сообщения конкретному автомату
   void MSendPrvLogging (CLogStr *a, int msgNum);
};

/*================================[ END REDEFINITION DEFENCE ]==============*/
#endif  //__RA_CLogStr__

/** (END OF FILE : LogStr.h )*************************************************/
```

4.2. Листинг описания класса “CAutomaton”

```
/*================================[ REDEFINITION DEFENCE ]=================*/
#ifndef __RA_CAutomaton__
#define __RA_CAutomaton__

/*================================[ IMPORT  DECLARATIONS ]=================*/
#include <stdio.h>
#include "LogStr.h"

/*****************************************************************************\
   CLASS.........: CAutomaton
   DESCRIPTION...: Автомат
\*****************************************************************************/
class CAutomaton: public CLogStr
{
protected:

   int y_cur, y_new;         // Переменные, кодирующие текущее и новое состояния автомата
   char *mPrv_new;           // Массив новых сообщений, посланных автомату
   char *mPrv_cur;           // Массив текущих сообщений, посланных автомату
   static char *mPub_new;    // Массив новых сообщений, посланных в общую шину
   static char *mPub_cur;    // Массив текущих сообщений, посланных в общую шину
   char **stNames;           // Массив названий состояний автомата

   // Каждая из этих функций вызывает функцию протоколирования
   char M (int msgNum);      // Функция проверки наличия сообщения
   char X (int xNum);        // Функция проверки входной переменной
   void Z (int zNum);        // Функция вызова выходного воздействия


   void UpdatePrvMsg();      // Функция обновляет массив сообщений автомата

public:

   virtual void TA(){}       // Функция "Переход-действие" автомата

   void U(); // Функция "Обновление" автомата. Вызывает функцию UpdatePrvMsg и обновляет
             // переменную, кодирующую состояние автомата.

   static void UpdatePubMsg(); // Функция обновляет массив сообщений общей шины

   // Каждая из этих функций вызывает функцию протоколирования
   static void SendPubMsg(CLogStr *a, int msgNum); // Функция посылки сообщения в общую шину
   void SendPrvMsg(CLogStr *a, int msgNum);        // Функция посылки сообщения автомату

   CAutomaton(const char *aname, char anumber);
   ~CAutomaton();
};

/*================================[ END REDEFINITION DEFENCE ]==============*/
#endif  //__RA_CAutomaton__

/** (END OF FILE : Automaton.h )**********************************************/
```

4.3. Листинг шаблона функции “*Переход-действие*” автомата

```
/*============================[ IMPORT DECLARATIONS ]=================*/
#include "Automata.h"
void CAa::TA()
{
   // Протоколирование начала работы автомата
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   // Переключатель по значению текущего состояния
   switch (y_cur)
   {
      // Состояние с номером i
      case i:
         Z(k);                      // Действие в состоянии
         SendPubMsg(this, m);       // Посылка сообщения с номером m в общую шину
         An.SendPrvMsg(this, m);    // Посылка сообщения с номером m автомату An
         Ains.TA();                 // Вызов функций “Переход-действие” вложенных автоматов

         // Дуга
         if ( X(h) || M(g) )        // Проверка условий на дуге
         {
            Z(k);                   // Вызов функций выходных воздействий
            SendPubMsg(this, m);    // Посылка сообщения с номером m в общую шину
            An.SendPrvMsg(this, m); // Посылка сообщения с номером m автомату An
            y_new = j;              // Присваивание переменной нового состояния
         }
         else
         // Дуга
         if (…) {…}
         break;

      // Состояние с номером j
      case j:
         …
      // Неизвестный номер состояния
      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur); // Протоколирование ошибки
   }

   // Если был переход в новое состояние
   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING)
         // Протоколирование перехода в новое состояние
         ATransLogging(y_cur, y_new, stNames[y_cur], stNames[y_new]);

      // Переключатель по значению нового состояния
      switch (y_new)
      {
         // Состояние с номером i
         case i:

            /* Инициализация вложенных автоматов путем посылки им
            служебного сообщения с номером 0.*/
            Ains.SendPrvMsg(this, 0); Ains.TA();

            Z(k); // Действие по приходу в новое состояние
            // Посылка сообщений по приходу в новое состояние
            SendPubMsg(this, m);
            An.SendPrvMsg(this, m);
         break;

      // Неизвестный номер состояния
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new); // Протоколирование ошибки

      }
   }

   // Протоколирование завершения работы автомата
   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);

TA_Log(); // Вызов автомата протоколирования
}
/** (END OF FILE : Aa.cpp )*********************************************/
```

4.4. Листинг функции “Обновление” автомата

```
void CAutomaton::U()
{
   y_cur = y_new;    // обновление значения внутренней переменной, кодирующей состояния автомата
   UpdatePrvMsg();   // обновление массива поступающих автомату сообщений
}
```

4.5. Листинг цикла работы автоматов

Функция *RunAuta* реализует цикл работы автоматов. Один вызов этой функции является тактом работы системы.

```
void RunAuta()
{
   // Вызов функций “Переход-действие” параллельно работающих автоматов
   A1.TA();
   A2.TA();
   A4.TA();
   A5.TA();
   A7.TA();
   A8.TA();

   // Вызов функций “Обновление” всех автоматов
   A1.U();
   A2.U();
   A3.U();
   A4.U();
   A5.U();
   A6.U();
   A7.U();
   A8.U();

   // Вызов функции обновления массива, реализующего общую шину сообщений
   CAutomaton::UpdatePubMsg();

   // Увеличение номера такта
   Counter++;
}
```

5. Перечень сообщений (m)

Внешние сообщения от объектов управления

m3 - Дверь открылась

m4 - Дверь закрылась

m11 - Обнаружена магнитная карта в приемнике

Внутренние сообщения от автоматов

m1 - Открыть дверь

m2 - Закрыть дверь

m5 - Включить сигнализацию

m8 - Зажечь лампу “Проход разрешен”

m9 - Погасить лампу “Проход разрешен”

m10 - Зажечь лампу “Проход запрещен”

Сообщения от таймеров

m6 - Сработал таймер времени работы сигнализации

m7 - Сработал таймер времени горения лампы “Проход запрещен”

m12 - Сработал таймер времени ожидания прохода тележки

6. Перечень входных переменных (x)

От объекта управления

x1 - Пересечен фотодатчик № 1

x2 - Пересечен фотодатчик № 2

x3 - Пересечен фотодатчик № 3

x4 - Пересечен фотодатчик № 4

x5 - Пересечен фотодатчик № 5

На рис. 7 приведена схема расположения фотодатчиков.

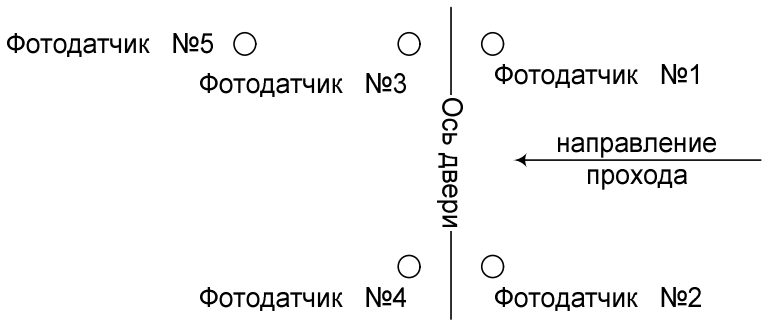

Рис. 7. Схема расположения фотодатчиков

x6 - Число оплаченных поездок больше нуля

x7 - Число оплаченных поездок равно нулю

От органов управления

x8 - Нажата кнопка “Ручной контроль”

x10 - Нажата кнопка “Открытие/закрытие дверей”

От смежных систем

x11 - Номер магнитной карты прочитан

x12 - Магнитная карта валидна (действительна)

x13 - Магнитная карта невалидна (недействительна)

x14 - Магнитная карта принадлежит черному списку

7. Перечень выходных воздействий (z)

На объект управления

z1 - Включить механизм открытия двери

z2 - Выключить механизм открытия двери

z3 - Включить механизм закрытия двери

z4 - Выключить механизм закрытия двери

z6 - Включить сигнализацию

z7 - Выключить сигнализацию

z9 - Зажечь лампу “Проход разрешен”

z10 - Погасить лампу “Проход разрешен”

z11 - Зажечь лампу “Проход запрещен”

z12 - Погасить лампу “Проход запрещен”

z13 - Уменьшить число оплаченных поездок на единицу

z18 - Увеличить число оплаченных поездок на единицу

На таймеры

z5 - Включить таймер времени работы сигнализации

z8 - Включить таймер времени горения лампы “Проход запрещен”

z24 - Включить таймер времени ожидания прохода тележки

На органы управления

z16 - Зажечь лампу “Ручной контроль”

z17 - Погасить лампу “Ручной контроль”

На смежные системы

z19 - Принять магнитную карту

z20 - Прочитать номер магнитной карты

z21 - Вернуть магнитную карту

z22 - Забрать магнитную карту в бункер

z23 - Отослать номер магнитной карты на сервер

8. Схема взаимодействия автоматов

На рис. 8 приведена схема взаимодействия автоматов.

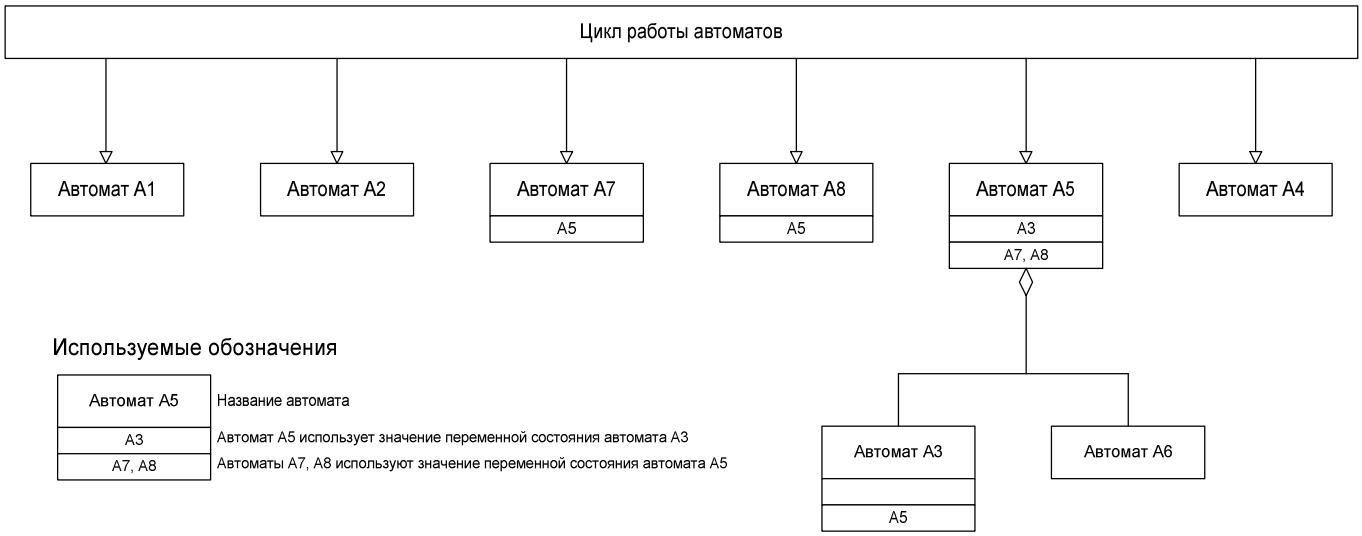

Рис. 8. Схема взаимодействия автоматов

В этой схеме используются следующие автоматы:

A1 - Управление дверью

A2 - Управление лампами контроля прохода

A3 - Управление проходом в автоматическом режиме

A4 - Управление сигнализацией

A5 - Управление режимами

A6 - Управление проходом в ручном режиме

A7 - Управление приемником магнитных карт

A8 - Управление устройством добавления контролером числа оплаченных поездок

9. Автомат “Управление дверью” (A1)

9.1. Словесное описание

Автомат управляет механизмом двери. Получая сообщения от автоматов, управляющих проходом пассажиров, и от датчиков на механизме, автомат воздействует на двигатель, обеспечивающий поворот двери.

9.2. Схема связей

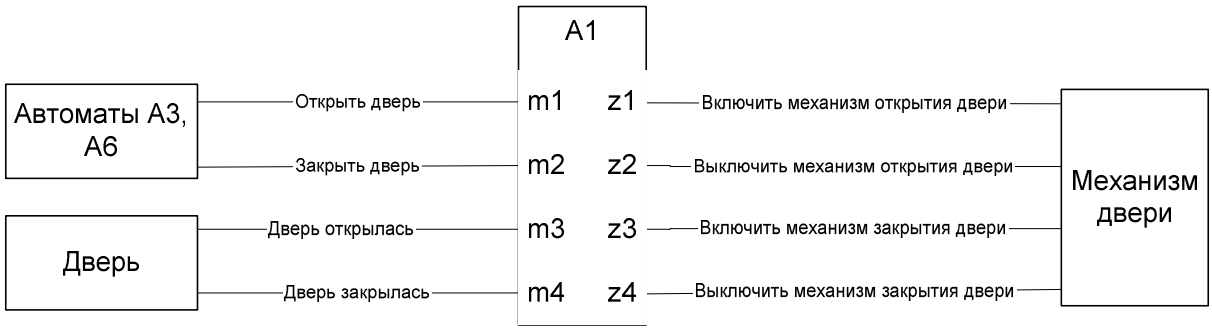

Рис. 9. Схема связей автомата A1

9.3. Граф переходов

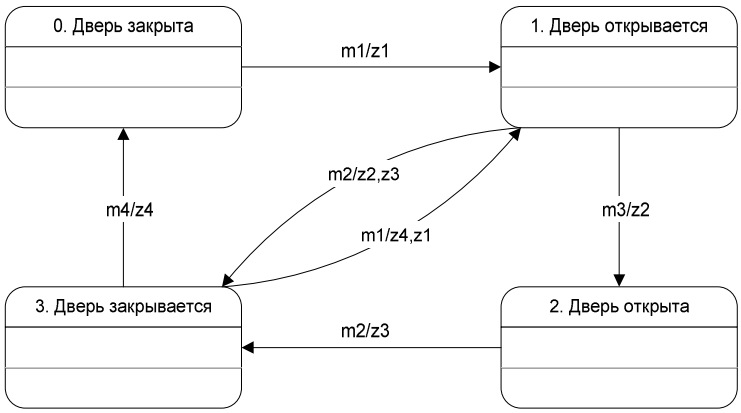

Рис. 10. Граф переходов автомата A1

10. Автомат “Управление лампами контроля прохода” (A2)

10.1. Словесное описание

Автомат управляет лампами “Проход разрешен” и “Проход запрещен”. Получая соответствующие сообщения, автомат зажигает и гасит лампу “Проход разрешен” и зажигает лампу “Проход запрещен”. Гашение последней происходит не по сообщению от автоматов, а по сообщению от таймера.

10.2. Схема связей

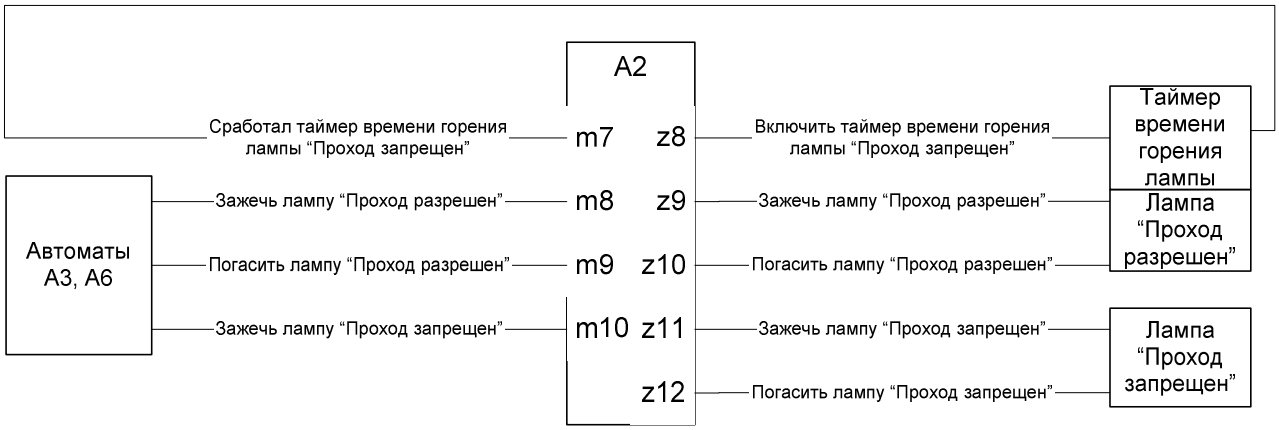

Рис. 11. Схема связей автомата A2

10.3. Граф переходов

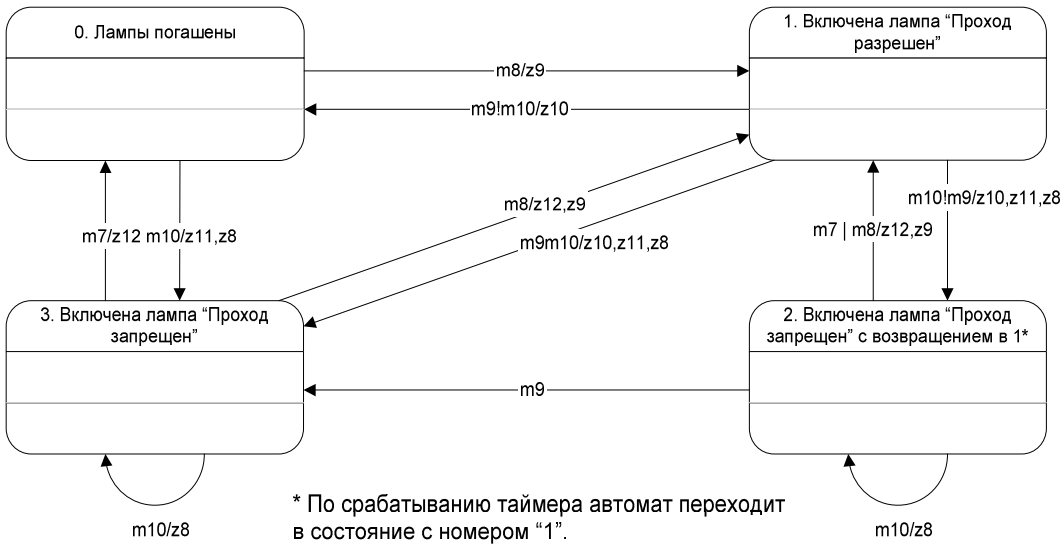

Рис. 12. Граф переходов автомата A2

11. Автомат “Управление проходом в автоматическом режиме” (А3)

11.1. Словесное описание

Автомат отслеживает проход пассажира через турникет. С помощью пяти фотодатчиков, расположенных на турникете, автомат “узнает”, идет ли человек с тележкой, без нее, или кто-то пытается пройти бесплатно, например, пристроившись сзади. Автомат воздействует на дверь, лампы контроля прохода, сигнализацию и на счетчик числа оплаченных поездок (кредит).

11.2. Схема связей

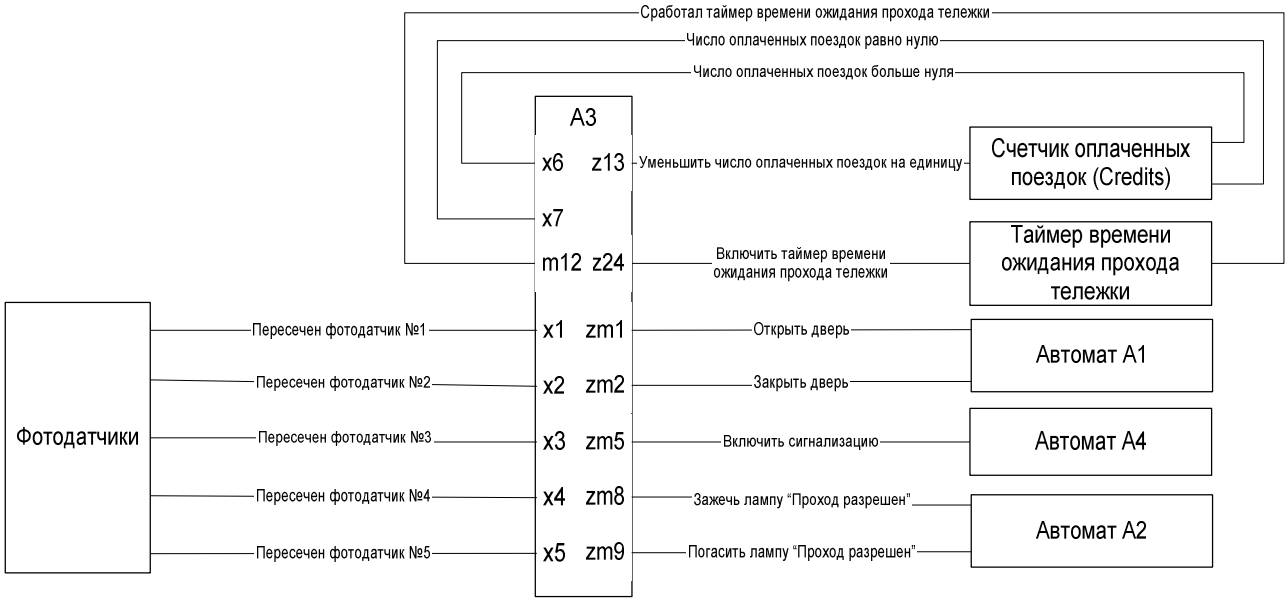

Рис. 13. Схема связей автомата А3

11.3. Граф переходов

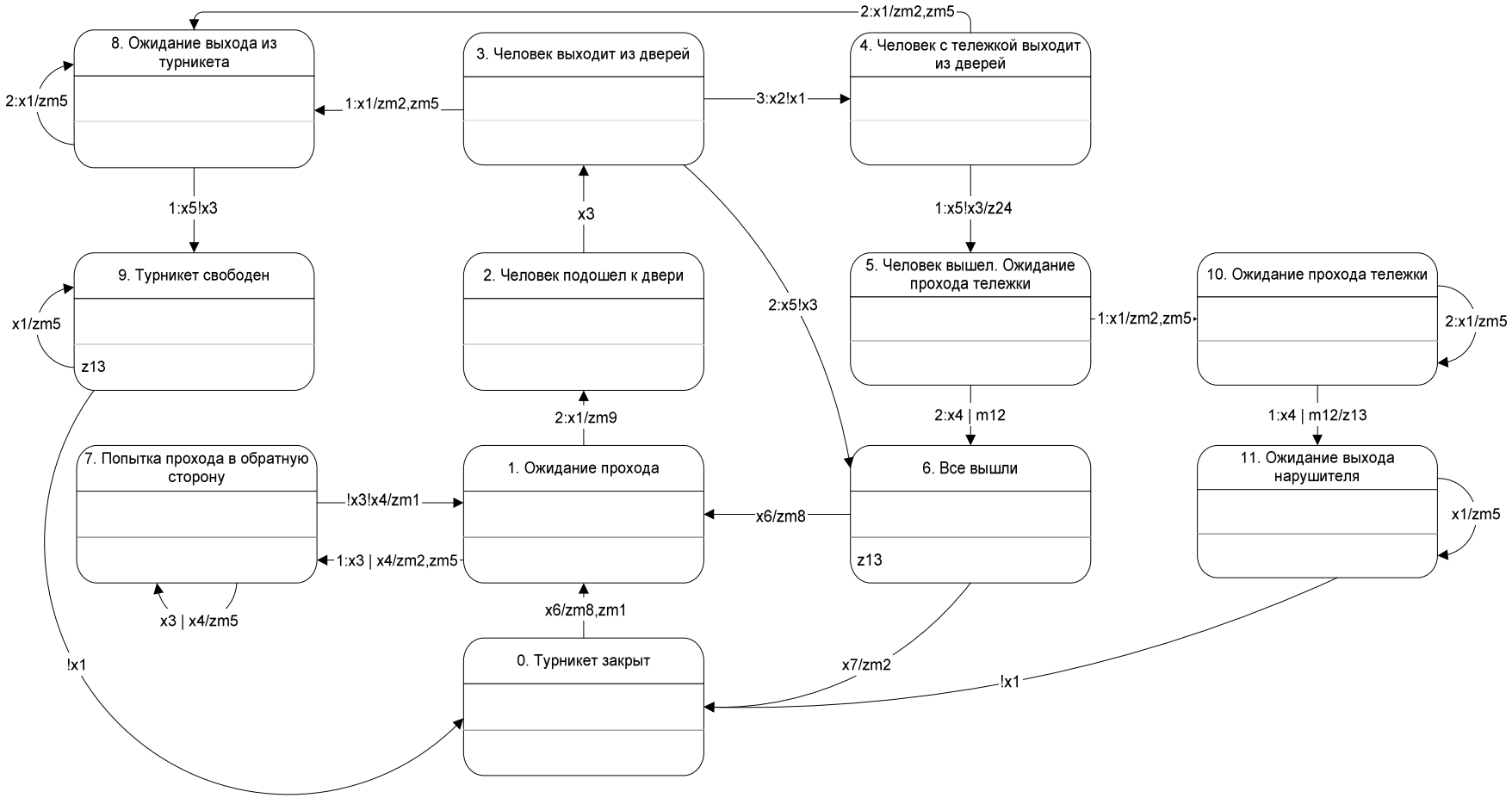

Рис. 14. Граф переходов автомата А3

12. Автомат “Управление сигнализацией” (А4)

12.1. Словесное описание

Автомат управляет устройством звуковой сигнализации. По сообщению от автомата “Управление проходом в автоматическом режиме” звуковая сигнализация включается. Она выключается по сообщению таймера сигнализации.

12.2. Схема связей

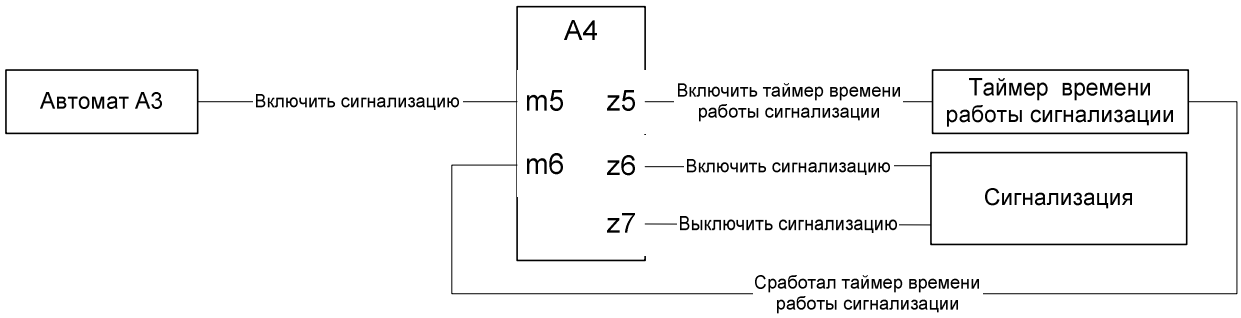

Рис. 15. Схема связей автомата А4

12.3. Граф переходов

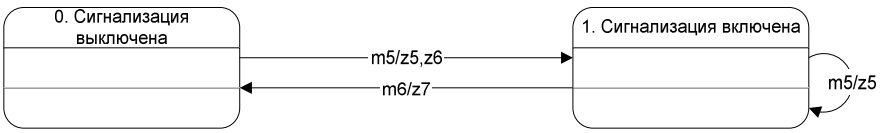

Рис. 16. Граф переходов автомата А4

13. Автомат “Управление режимами” (A5)

13.1. Словесное описание

Автомат управляет переключением двух режимов работы турникета. В зависимости от выбранного режима вызывается автомат “Управление проходом в автоматическом режиме” (A3) или “Управление проходом в ручном режиме” (A6). При нажатии кнопки “Ручной контроль” автомат дожидается (в состоянии с номером “1”) прохода всех уже оплативших проезд пассажиров и только после этого включает ручной контроль (переходит в состояние с номером “2”).

13.2. Схема связей

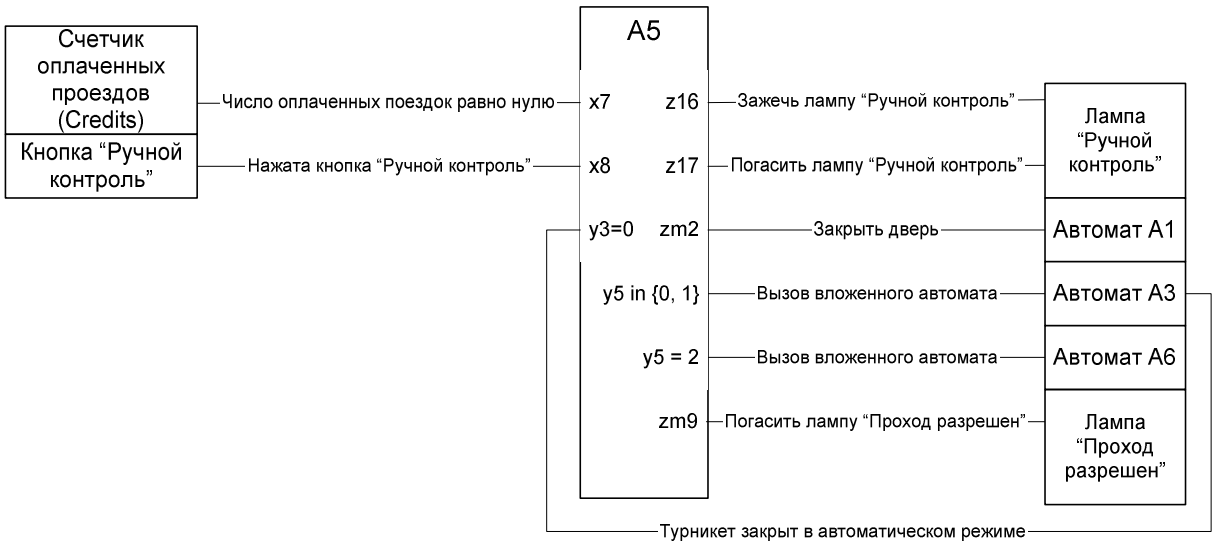

Рис. 17. Схема связей автомата A5

13.3. Граф переходов

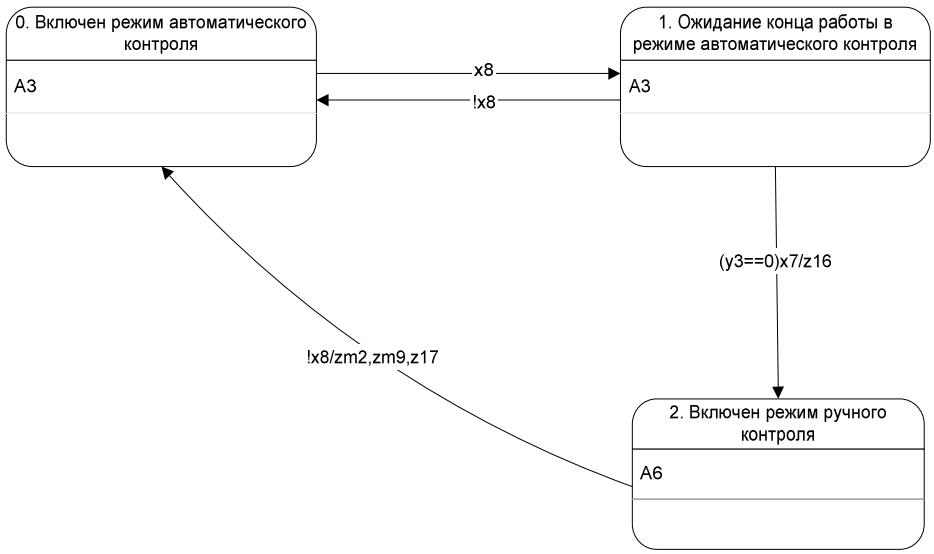

Рис. 18. Граф переходов автомата A5

14. Автомат “Управление проходом в ручном режиме” (А6)

14.1. Словесное описание

Автомат управляет проходом пассажиров в режиме ручного контроля. С помощью кнопки “Открытие/закрытие” контролер разрешает или запрещает проход пассажира.

14.2. Схема связей

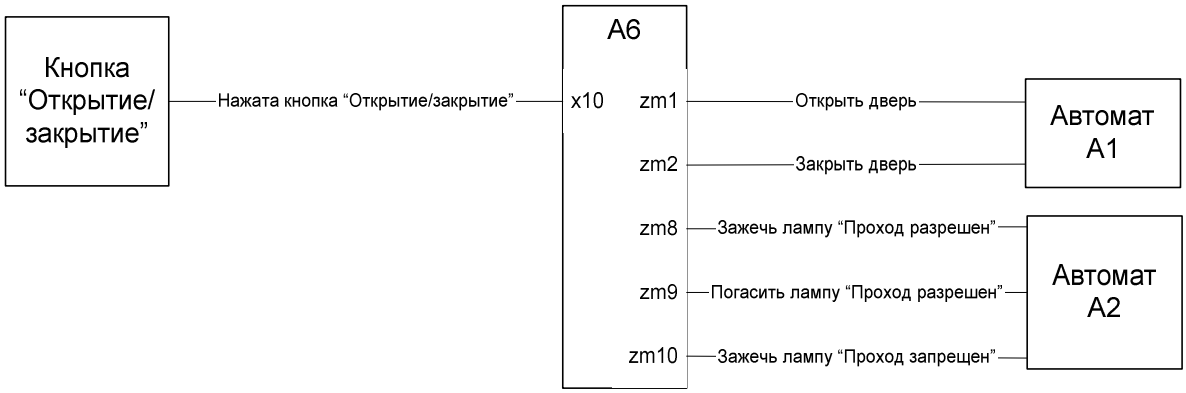

Рис. 19. Схема связей автомата А6

14.3. Граф переходов

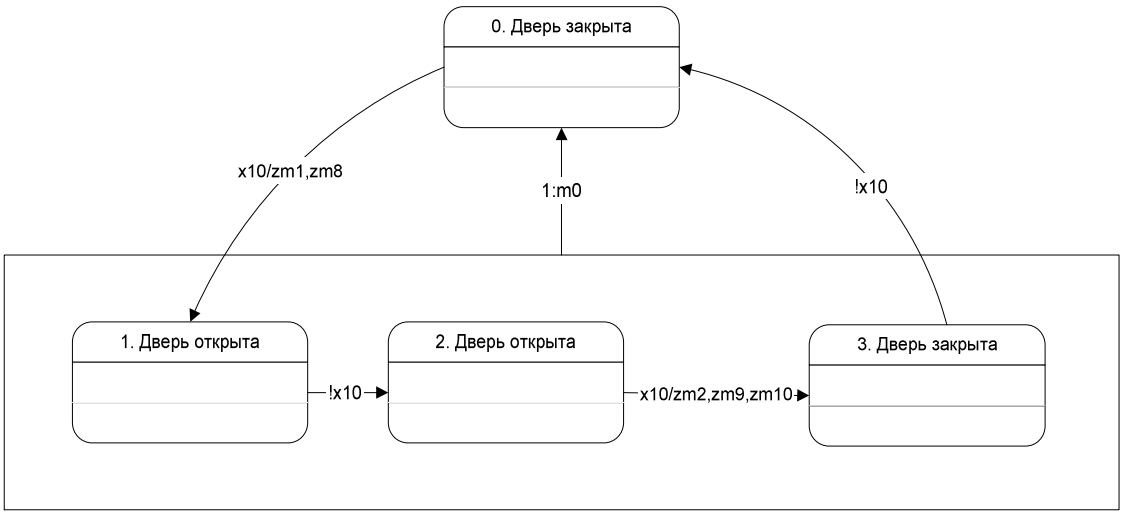

Рис. 20. Граф переходов автомата А6

15. Автомат “Управление приемником магнитных карт” (A7)

15.1. Словесное описание

Автомат управляет устройством, проверяющим возможность прохода. Этот автомат обеспечивает ввод магнитной карты, ее чтение, анализ и принятие решения о ее валидности (действительности).

15.2. Схема связей

Рис. 21. Схема связей автомата A7

15.3. Граф переходов

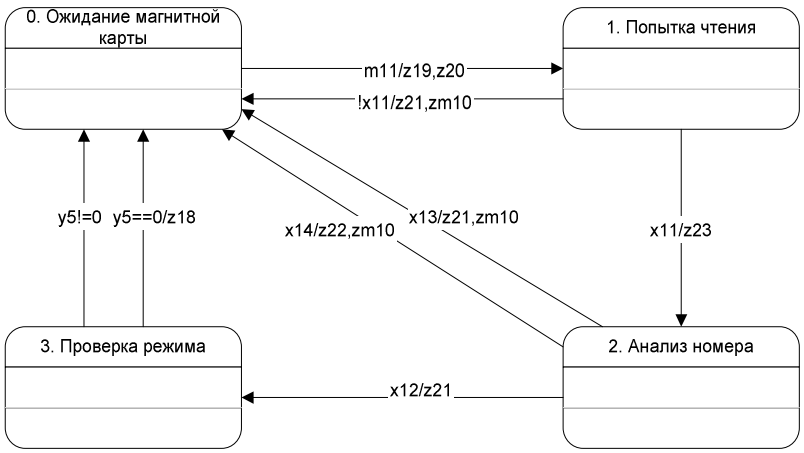

Рис. 22. Граф переходов автомата A7

16. Автомат “Управление устройством добавления контролером числа оплаченных поездок” (A8)

16.1. Словесное описание

Автомат управляет устройством, проверяющим возможность прохода. При нажатии на кнопку “Открытие/закрытие” автомат добавляет одну поездку к числу оплаченных.

16.2. Схема связей

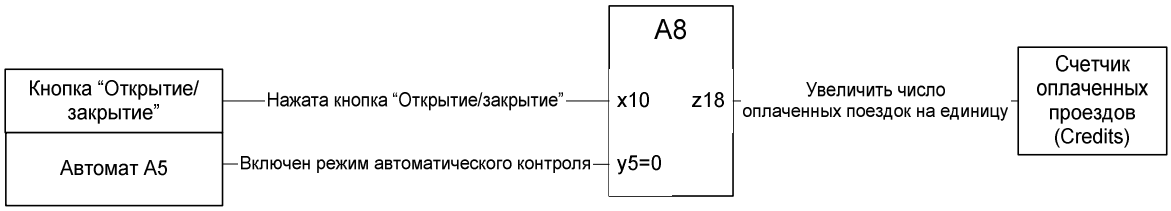

Рис. 23. Схема связей автомата A8

16.3. Граф переходов

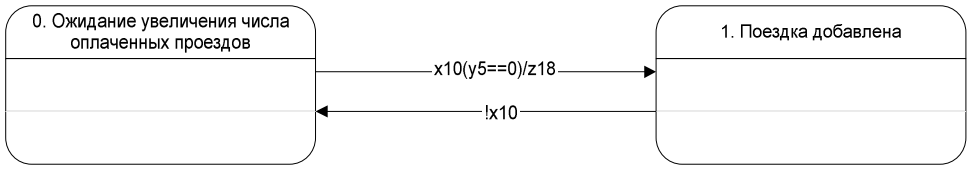

Рис. 24. Граф переходов автомата A8

17. Пример протокола

Ниже приведен протокол работы автомата “Управление проходом в автоматическом режиме” (А3) при “щадящем” протоколировании.

Протокол соответствует следующему сценарию.

Вставляется действительная магнитная карта с номером 239. Человек проходит через турникет с тележкой. При этом имеет место следующий порядок пересечения фотодатчиков:

- не один фотодатчик не пересечен;
- пересечены фотодатчики с номерами 1 и 2;
- пересечен фотодатчик с номером 1;
- не один фотодатчик не пересечен;
- пересечены фотодатчики с номерами 3, 4, 2;
- пересечен фотодатчик с номером 3;
- не один фотодатчик не пересечен;
- пересечен фотодатчик с номером 5;
- не один фотодатчик не пересечен;
- пересечен фотодатчик с номером 4;
- не один фотодатчик не пересечен.

Напомним, что в разд. 3.3 для задания типа протоколирования было предложено использовать следующие символы:

{ - начало работы автомата;
T – переход автомата в новое состояние;
E – ошибка (неизвестный номер состояния);
? – проверка наличия сообщения;
> - вызов функции входной переменной;
< - вызов функции выходной переменной;
! – посылка сообщения;
} – завершение работы автомата.

Протокол работы автомата A3(Управление проходом в автоматическом режиме)

```
64 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 0(Турникет
       закрыт).
64   > x6(Число оплаченных поездок больше нуля) - вернул 0.
64 }   A3(Управление проходом в автоматическом режиме) завершил работу в
       состоянии 0(Турникет закрыт).
       ...
180 {   A3(Управление проходом в автоматическом режиме) запущен в
       состоянии 0(Турникет закрыт).
180   > x6(Число оплаченных поездок больше нуля) - вернул 1.
180   ! A3(Управление проходом в автоматическом режиме) послал сообщение m8(Зажечь
       лампу "Проход разрешен") в общую шину сообщений.
180   ! A3(Управление проходом в автоматическом режиме) послал сообщение m1(Открыть
       дверь) в общую шину сообщений.
180   T A3(Управление проходом в автоматическом режиме) перешел из
       состояния 0(Турникет закрыт) в 1(Ожидание прохода).
180 }   A3(Управление проходом в автоматическом режиме) завершил работу в
       состоянии 1(Ожидание прохода).
181 {   A3(Управление проходом в автоматическом режиме) запущен в
       состоянии 1(Ожидание прохода).
181   > x3(Пересечен фотодатчик номер 3) - вернул 0.
181   > x4(Пересечен фотодатчик номер 4) - вернул 0.
181   > x1(Пересечен фотодатчик номер 1) - вернул 0.
```

```
181 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 1(Ожидание прохода).
        ...
261 {   A3(Управление проходом в автоматическом режиме) запущен в
        состоянии 1(Ожидание прохода).
261   > x3(Пересечен фотодатчик номер 3) - вернул 0.
261   > x4(Пересечен фотодатчик номер 4) - вернул 0.
261   > x1(Пересечен фотодатчик номер 1) - вернул 1.
261   ! A3(Управление проходом в автоматическом режиме) послал сообщение m9(Погасить
        лампу "Проход разрешен") в общую шину сообщений.
261   T A3(Управление проходом в автоматическом режиме) перешел из
        состояния 1(Ожидание прохода) в 2(Человек подошел к двери).
261 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 2(Человек подошел к двери).
262 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 2(Человек
        подошел к двери).
262   > x3(Пересечен фотодатчик номер 3) - вернул 0.
262 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 2(Человек подошел к двери).
        ...
456 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 2(Человек
        подошел к двери).
456   > x3(Пересечен фотодатчик номер 3) - вернул 1.
456   T A3(Управление проходом в автоматическом режиме) перешел из
        состояния 2(Человек подошел к двери) в 3(Человек выходит из дверей).
456 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 3(Человек выходит из дверей).
457 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 3(Человек
        выходит из дверей).
457   > x1(Пересечен фотодатчик номер 1) - вернул 0.
457   > x5(Пересечен фотодатчик номер 5) - вернул 0.
457   > x2(Пересечен фотодатчик номер 2) - вернул 1.
457   > x1(Пересечен фотодатчик номер 1) - вернул 0.
457   T A3(Управление проходом в автоматическом режиме) перешел из
        состояния 3(Человек выходит из дверей) в 4(Человек с тележкой выходит из
        дверей).
457 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 4(Человек с тележкой выходит из дверей).
458 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 4(Человек
        с тележкой выходит из дверей).
458   > x5(Пересечен фотодатчик номер 5) - вернул 0.
458   > x1(Пересечен фотодатчик номер 1) - вернул 0.
458 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 4(Человек с тележкой выходит из дверей).
        ...
540 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 4(Человек
        с тележкой выходит из дверей).
540   > x5(Пересечен фотодатчик номер 5) - вернул 1.
540   > x3(Пересечен фотодатчик номер 3) - вернул 0.
540   < z24(Включить таймер времени ожидания тележки).
540   T A3(Управление проходом в автоматическом режиме) перешел из
        состояния 4(Человек с тележкой выходит из дверей) в 5(Человек вышел. Ожидание
        прохода тележки).
540 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 5(Человек вышел. Ожидание прохода тележки).
541 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 5(Человек
        вышел. Ожидание прохода тележки).
541   > x1(Пересечен фотодатчик номер 1) - вернул 0.
541   > x4(Пересечен фотодатчик номер 4) - вернул 0.
541   ? m12(Сработал таймер времени ожидания прохода тележки) не получено.
541 }   A3(Управление проходом в автоматическом режиме) завершил работу в
        состоянии 5(Человек вышел. Ожидание прохода тележки).
```

```
      ...
591 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 5(Человек
      вышел. Ожидание прохода тележки).
591   > x1(Пересечен фотодатчик номер 1) - вернул 0.
591   > x4(Пересечен фотодатчик номер 4) - вернул 0.
591   ? m12(Сработал таймер времени ожидания прохода тележки) получено.
591   T A3(Управление проходом в автоматическом режиме) перешел из
      состояния 5(Человек вышел. Ожидание прохода тележки) в 6(Все вышли).
591   < z13(Уменьшить число оплаченных поездок на единицу).
591 }   A3(Управление проходом в автоматическом режиме) завершил работу в
      состоянии 6(Все вышли).
592 {   A3(Управление проходом в автоматическом режиме) запущен в состоянии 6(Все
      вышли).
592   > x6(Число оплаченных поездок больше нуля) - вернул 0.
592   > x7(Число оплаченных поездок равно нулю) - вернул 1.
592   ! A3(Управление проходом в автоматическом режиме) послал сообщение m2(Закрыть
      дверь) в общую шину сообщений.
592   T A3(Управление проходом в автоматическом режиме) перешел из состояния 6(Все
      вышли) в 0(Турникет закрыт).
592 }   A3(Управление проходом в автоматическом режиме) завершил работу в
      состоянии 0(Турникет закрыт).
593 {   A3(Управление проходом в автоматическом режиме) запущен в
      состоянии 0(Турникет закрыт).
593   > x6(Число оплаченных поездок больше нуля) - вернул 0.
593 }   A3(Управление проходом в автоматическом режиме) завершил работу в
      состоянии 0(Турникет закрыт).
      ...
```

Заключение

В работе предложено использовать автоматы, работающие параллельно и взаимодействующие между собой за счет обмена сообщениями. Это напоминает такую область параллельных вычислений, как распределенное программирование, в которой процессы также взаимодействуют посредством обмена сообщениями [2].

Предложенный подход позволил существенно упростить процесс взаимодействия автоматов, что позволило упростить процесс проектирования программного обеспечения системы.

Этот подход позволяет также расширить функциональность системы без изменения существующей структуры. При этом для реализации новой функции системы можно ввести еще один автомат, посылающий уже существующие сообщения имеющимся автоматам.

Известен метод проектирования объектно-ориентированных систем – *Рациональный Унифицированный Процесс (Rational Unified Process)*, разработанный фирмой **Rational Software Corporation**, который использует в качестве нотации язык UML [7]. Однако существуют и другие методы объектно-ориентированного проектирования, например, описанный в работе [8].

Подход, изложенный в настоящей работе, является еще одним методом проектирования подобных систем.

Источники

1. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. *Эндрюс Грегори Р.* Основы многопоточного, параллельного и распределенного программирования. М.: Вильямс, 2003.
3. *Шалыто А.А., Туккель Н.И.* SWITCH-технология – автоматный подход к созданию программного обеспечения событийных систем //Программирование. 2001. №5. http://is.ifmo.ru, раздел “Статьи”.
4. *Туккель Н.И., Шалыто А.А.* Система управления танком для игры *Robocode*. Вариант 1. Объектно-ориентированное программирование с явным выделением состояний. http://is.ifmo.ru, раздел “Проекты”.
5. http://msdn.microsoft.com
6. *Гуисов М.И., Кузнецов А.Б., Шалыто А.А.* Интеграция механизма обмена сообщениями в Switch-технологию. http://is.ifmo.ru, раздел “Проекты”.
7. *Якобсон А., Буч Г., Рамбо Дж.* Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002.
8. *Дейтел Х. М., Дейтел П. Дж.* Как программировать на C++. М.: Бином, 2003

Приложения

Листинг файла с описанием класса *CLogStr* приведен в **разд. 4.1**.

П 1. Листинг реализации методов класса “CLogStr”

```
/*================================[ IMPORT DECLARATIONS ]=================*/
#include "StdAfx.h"
#include "String.h"
#include "StdLib.h"
#include "LogStr.h"
#include "z.h"
#include "x.h"
#include "m.h"

CLogStr::CLogStr(const char *aname, char anumber)
{
   y = 0;
   index_old = new char[5000];
   index_cur = new char[5000];
   index_old[0] = 0;
   index_cur[0] = 0;
   buffer[0] = 0;
   LOGGING = 0;
   M_LOGGING = 1;
   Z_LOGGING = 1;
   X_LOGGING = 1;
   M_SEND_LOGGING = 1;
   A_BEGINS_LOGGING = 1;
   A_TRANS_LOGGING = 1;
   A_ENDS_LOGGING = 1;
   A_ERRORS_LOGGING = 1;
   NO_REPEAT_LOGGING = 0;
   number = anumber;
   log_file = 0;
   strcpy(name, aname);

}

CLogStr::~CLogStr()
{
   if (log_file != 0)
   {
      fprintf(log_file,"_____________________________________________________");
      fclose(log_file);
      log_file = 0;
   }
   delete [] index_old;
   delete [] index_cur;
}

void CLogStr::SetLogging()
{
   char fname[256];
   char numStr[256];

   if (LOGGING == 0)
   {
      LOGGING = 1;
      if (number != 0)
      {
         strcpy(fname,"A");
         itoa(number, numStr, 10);
         strcat(fname, numStr);
      } else
      {
         strcpy(fname,name);
      }

      strcat(fname, ".log");
      if (!log_file)
      {
         log_file = fopen(fname, "w");
         if (number != 0)
            fprintf(log_file,"\nПротокол работы автомата \"%s\"(A%d)\n",name,number);
         else
```

```
            fprintf(log_file,"\nПротокол работы объекта \"%s\"\n",name);
         fprintf(log_file,"_____________________________________________________\n\n");
      }
   }
}

void CLogStr::ReSetLogging()
{
   if (LOGGING == 1)
   {
      LOGGING = 0;
   }
}

char CLogStr::Is_Logging()
{
   return LOGGING;
}

// Входные переменные
char CLogStr::xL1()
{
   return (!strcmp(index_old,index_cur));
}

char CLogStr::xL2()
{
   return NO_REPEAT_LOGGING;
}

char CLogStr::xL3()
{
   return LOGGING;
}

// Выходные воздействия
void CLogStr::zL1()
{
   fprintf(log_file, "%s", buffer);
}

void CLogStr::zL2()
{
   fprintf(log_file, "       ...\n");
}

void CLogStr::zL3()
{
   char *temp;
   buffer[0] = 0;
   temp = index_old;
   index_old = index_cur;
   index_cur = temp;
   index_cur[0] = 0;
}

void CLogStr::zL4()
{
   buffer[0] = 0;
   index_old [0] = 0;
   index_cur [0] = 0;
}

// Автомат протоколирования
void CLogStr::TA_Log()
{
   switch (y)
   {
      case 0:
         if (xL3())                  { zL4();            y = 2; }
         break;
      case 1:
         if ((xL1()&&xL2())&&xL3())  { zL3();            y = 1; }
         else
         if ((!xL1()||!xL2())&&xL3()) { zL1(); zL3();    y = 2; }
         else
         if (!xL3())                 { zL4();            y = 0; }
         break;
```

```
      case 2:
         if ((xL1()&&xL2())&&xL3())   { zL2(); zL3();      y = 1; }
         else
         if ((!xL1()||!xL2())&&xL3()) { zL1(); zL3();      y = 2; }
         else
         if (!xL3())                  { zL4();             y = 0; }
         break;
   }
}

char *const CLogStr::GetName()
{
   return (char *const) name;
}

char CLogStr::GetNumber()
{
   return number;
}

void CLogStr::MLogging(int msgNum, char b)
{
   if (b)
   {
      sprintf(buffer+strlen(buffer), "%d   %c m%d(%s) получено. \n", Counter, M_LOGGING_C,
msgNum, MNames[msgNum]);
      sprintf(index_cur+strlen(index_cur),"M[%d][1]",msgNum);
   }
   else
   {
      sprintf(buffer+strlen(buffer), "%d   %c m%d(%s) не получено. \n", Counter, M_LOGGING_C,
msgNum, MNames[msgNum]);
      sprintf(index_cur+strlen(index_cur),"M[%d][0]",msgNum);
   }
}

void CLogStr::ABeginsLogging(int y, char *stName)
{
   sprintf(buffer+strlen(buffer), "%d %c   A%d(%s) запущен в состоянии %d(%s). \n", Counter,
A_BEGINS_LOGGING_C, number, name, y, stName);
   sprintf(index_cur+strlen(index_cur),"AB[%d]",y);
}

void CLogStr::AEndsLogging(int y, char *stName)
{
   sprintf(buffer+strlen(buffer), "%d %c   A%d(%s) завершил работу в состоянии %d(%s). \n",
Counter, A_ENDS_LOGGING_C, number, name, y, stName);
   sprintf(index_cur+strlen(index_cur), "AE[%d]",y);
}

void CLogStr::ATransLogging(int y1, int y2, char *stName1, char *stName2)
{
   sprintf(buffer+strlen(buffer), "%d   %c A%d(%s) перешел из состояния %d(%s) в %d(%s). \n",
Counter, A_TRANS_LOGGING_C, number, name, y1, stName1, y2, stName2);
   sprintf(index_cur+strlen(index_cur), "AT[%d][%d]",y1,y2);
}

void CLogStr::AErrorsLogging(int y)
{
   sprintf(buffer+strlen(buffer), "%d   %c Ошибка в автомате A%d(%s). Неизвестный номер состояния
- %d. \n", Counter, A_ERRORS_LOGGING_C, number, name, y);
   sprintf(index_cur+strlen(index_cur), "ER[%d]",y);
}

void CLogStr::ZLogging (int zNum)
{
   sprintf(buffer+strlen(buffer), "%d   %c z%d(%s). \n", Counter, Z_LOGGING_C, zNum,
ZNames[zNum]);
   sprintf(index_cur+strlen(index_cur), "Z[%d]",zNum);
}

void CLogStr::XLogging (int xNum, char b)
{
   if (b)
      sprintf(buffer+strlen(buffer), "%d   %c x%d(%s) - вернул %d. \n", Counter, X_LOGGING_C,
xNum, XNames[xNum], b);
   else
```

```
      sprintf(buffer+strlen(buffer), "%d   %c x%d(%s) - вернул %d. \n", Counter, X_LOGGING_C,
xNum, XNames[xNum], b);
   sprintf(index_cur+strlen(index_cur), "X[%d][%d]",xNum,b);

}

void CLogStr::MSendPubLogging (CLogStr *a, int msgNum)
{
   if (a->number != 0)
   {
      sprintf(a->buffer+strlen(a->buffer), "%d   %c A%d(%s) послал сообщение m%d(%s) в общую шину
сообщений. \n", Counter, M_SEND_LOGGING_C, a->number, a->name, msgNum, MNames[msgNum]);
      sprintf(a->index_cur+strlen(a->index_cur),"MSB[%d]",msgNum);
   }
   else
      fprintf(a->log_file, "%d   %c %s послала сообщение m%d(%s) в общую шину сообщений. \n",
Counter, M_SEND_LOGGING_C, a->name, msgNum, MNames[msgNum]);
}

void CLogStr::MSendPrvLogging (CLogStr *a, int msgNum)
{
   if (a->number != 0)
   {
      sprintf(a->buffer+strlen(a->buffer), "%d   %c A%d(%s) послал сообщение m%d(%s) автомату
A%d(%s). \n", Counter, M_SEND_LOGGING_C, a->number, a->name, msgNum, MNames[msgNum], number,
name);
      sprintf(a->index_cur+strlen(a->index_cur),"MSV[%d][%d]",msgNum, number);
   }
   else
      fprintf(a->log_file, "%d   %c %s послала сообщение m%d(%s) автомату A%d(%s). \n", Counter,
M_SEND_LOGGING_C, a->name, msgNum, MNames[msgNum], number, name);
}
/** (END OF FILE : LogStr.cpp )*********************************************/
```

Листинг файла с описанием класса *CAutomaton* приведен в **разд. 4.1**.

П 2. Листинг реализации методов класса “CAutomaton”

```
/*================================[ IMPORT  DECLARATIONS ]=================*/
#include "StdAfx.h"
#include "String.h"
#include "StdLib.h"
#include "Automaton.h"
#include "x.h"
#include "z.h"

const gMsgNum = 100;

CAutomaton::CAutomaton(const char *aname, char anumber):CLogStr(aname, anumber)
{
   mPrv_cur = new char[gMsgNum];
   mPrv_new = new char[gMsgNum];
   memset((void *)mPrv_new, 0, gMsgNum);
   memset((void *)mPrv_cur, 0, gMsgNum);
   y_new = 0;
   y_cur = 0;
   tNames = 0;
}

CAutomaton::~CAutomaton()
{
   delete [] mPrv_new;
   delete [] mPrv_cur;
   if(stNames) delete []stNames;
}


char CAutomaton::M(int msgNum)
{
   char b;

   b = (mPrv_cur[msgNum] || mPub_cur[msgNum]);
   if (LOGGING && M_LOGGING) MLogging(msgNum, b);
   return b;
}

char CAutomaton::X(int xNum)
{
   char b;
   b = x_arr[xNum]();
   if (LOGGING && X_LOGGING) XLogging(xNum, b);
   return b;
}

void CAutomaton::Z(int zNum)
{
   z_arr[zNum]();
   if (LOGGING && Z_LOGGING) ZLogging(zNum);
}

void CAutomaton::SendPubMsg(CLogStr *a, int msgNum)
{
   if (a->Is_Logging() && a->M_SEND_LOGGING) MSendPubLogging (a, msgNum);
   mPub_new[msgNum] = 1;
}

void CAutomaton::SendPrvMsg(CLogStr *a, int msgNum)
{
   if (a->Is_Logging() && a->M_SEND_LOGGING) MSendPrvLogging (a, msgNum);
   mPrv_new[msgNum] = 1;
}

void CAutomaton::UpdatePubMsg()
{
   memset((void *)mPub_cur, 0, gMsgNum);
   char *tmp;
   tmp = mPub_new;
   mPub_new = mPub_cur;
   mPub_cur = tmp;
}
```

```
void CAutomaton::UpdatePrvMsg()
{
   memset((void *)mPrv_cur, 0, gMsgNum);
   char *tmp;
   tmp = mPrv_new;
   mPrv_new = mPrv_cur;
   mPrv_cur = tmp;
}

void CAutomaton::U()
{
   y_cur = y_new;
   UpdatePrvMsg();
}

char tempM1[gMsgNum];
char tempM2[gMsgNum];
char *CAutomaton::mPub_new = tempM1;
char *CAutomaton::mPub_cur = tempM2;

/** (END OF FILE : Automaton.cpp )*********************************************/
```

П 3. Листинг функции “*Переход-действие*” автомата “Управление дверью” (A1)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

void CA1::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         if (M(1)) { Z(1);                  y_new = 1; }
         break;

      case 1:
         if (M(3)) { Z(2);                  y_new = 2; }
         else
         if (M(2)) { Z(2); Z(3);            y_new = 3; }
         break;

      case 2:
         if (M(2)) { Z(3);                  y_new = 3; }
         break;

      case 3:
         if (M(4)) { Z(4);                  y_new = 0; }
         else
         if (M(1)) { Z(4); Z(1);            y_new = 1; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         case 2:
            break;
         case 3:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A1.cpp )**************************************************/
```

П 4. Листинг функции “*Переход-действие*” автомата “Управление лампами контроля прохода” (A2)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

void CA2::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         if (M(8))           { Z(9);                    y_new = 1; }
         else
         if (M(10))          { Z(11); Z(8);             y_new = 3; }
         break;

      case 1:
         if (M(9) && !M(10)) { Z(10);                   y_new = 0; }
         else
         if (M(9) && M(10))  { Z(10); Z(11); Z(8);      y_new = 3; }
         else
         if (M(10) && !M(9)) { Z(10); Z(11); Z(8);      y_new = 2; }
         break;

      case 2:
         if (M(7) || M(8))   { Z(12); Z(9);             y_new = 1; }
         else
         if (M(9))           {                          y_new = 3; }
         else
         if(M(10))           { Z(8;                     y_new = 2; }
         break;

      case 3:
         if (M(7))           { Z(12);                   y_new = 0; }
         else
         if (M(8))           { Z(12); Z(9);             y_new = 1; }
         else
         if (M(10))          { Z(8);                    y_new = 3; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         case 2:
            break;
         case 3:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A2.cpp )**************************************************/
```

П 5. Листинг функции “*Переход-действие*” автомата “Управление проходом в автоматическом режиме” (А3)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

void CA3::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         if (X(6))             { SendPubMsg(this, 8); SendPubMsg(this, 1);  y_new = 1;  }
         break;

      case 1:
         if (X(3) || X(4))     { SendPubMsg(this, 2); SendPubMsg(this, 5);  y_new = 7;  }
         else
         if (X(1))             { SendPubMsg(this, 9);                       y_new = 2;  }
         break;

      case 2:
         if (X(3))             {                                            y_new = 3;  }
         break;

      case 3:
         if (X(1))             { SendPubMsg(this, 2); SendPubMsg(this, 5);  y_new = 8;  }
         else
         if (X(5) && !X(3))    {                                            y_new = 6;  }
         else
         if (X(2) && !X(1))    {                                            y_new = 4;  }
         break;

      case 4:
         if (X(5) && !X(3))    { Z(24);                                     y_new = 5;  }
         else
         if (X(1))             { SendPubMsg(this, 2); SendPubMsg(this, 5);  y_new = 8;  }
         break;

      case 5:
         if (X(1))             { SendPubMsg(this, 2); SendPubMsg(this, 5);  y_new = 10; }
         else
         if (X(4) || M(12))    {                                            y_new = 6;  }
         break;

      case 6:
         if (X(6))             { SendPubMsg(this, 8);                       y_new = 1;  }
         else
         if (X(7))             { SendPubMsg(this, 2);                       y_new = 0;  }
         break;

      case 7:
         if (!X(3) && !X(4))   { SendPubMsg(this, 1);                       y_new = 1;  }
         else
         if (X(3) || X(4))     { SendPubMsg(this, 5);                       y_new = 7;  }
         break;

      case 8:
         if (X(5) && !X(3))    {                                            y_new = 9;  }
         else
         if (X(1))             { SendPubMsg(this, 5);                       y_new = 8;  }
         break;

      case 9:
         if (!X(1))            {                                            y_new = 0;  }
         else
         if (X(1))             { SendPubMsg(this, 5);                       y_new = 9;  }
         break;

      case 10:
         if (X(4) || M(12))    { Z(13);                                     y_new = 11; }
         else
         if (X(1))             { SendPubMsg(this, 5);                       y_new = 10; }
```

```
         break;

      case 11:
         if (!X(1))            {                                        y_new = 0;  }
         else
         if (X(1))             { SendPubMsg(this, 5);                   y_new = 11; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         case 2:
            break;
         case 3:
            break;
         case 4:
            break;
         case 5:
            break;
         case 6:
            Z(13);
            break;
         case 7:
            break;
         case 8:
            break;
         case 9:
            Z(13);
            break;
         case 10:
            break;
         case 11:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A3.cpp )**************************************************/
```

П 6. Листинг функции “*Переход-действие*” автомата “Управление сигнализацией” (А4)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

void CA4::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         if (M(5)) { Z(5); Z(6);    y_new = 1; }
         break;

      case 1:
         if (M(6)) { Z(7);          y_new = 0; }
         else
         if (M(5)) { Z(5);          y_new = 1; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A4.cpp )**************************************************/
```

П 7. Листинг функции “*Переход-действие*” автомата “Управление режимами” (А5)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

extern CA3 A3;
extern CA6 A6;

void CA5::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         A3.TA();
         if (X(8))                  {                              y_new = 1; }
         break;

      case 1:
         A3.TA();
         if (!X(8))                 {                              y_new = 0; }
         else
         // A3.Is_Closed() ó y3 == 0
         if (A3.Is_Closed() && X(7)) { Z(16);                      y_new = 2; }
         break;

      case 2:
         A6.TA();
         if (!X(8){ SendPubMsg(this, 2); SendPubMsg(this, 9); Z(17); y_new = 0; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            A3.SendPrvMsg(this, 0);  A3.TA();
            break;

         case 1:
            A3.SendPrvMsg(this, 0);  A3.TA();
            break;

         case 2:
            A6.SendPrvMsg(this, 0);  A6.TA();
            break;

         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}

/** (END OF FILE : A5.cpp )*************************************************/
```

П 8. Листинг функции “*Переход-действие*” автомата “Управление проходом в ручном режиме” (A6)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

void CA6::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         if (X(10))  { SendPubMsg(this, 1); SendPubMsg(this, 8);                  y_new = 1; }
         break;

      case 1:
         if (M(0))   {                                                            y_new = 0; }
         else
         if (!X(10)) {                                                            y_new = 2; }
         break;

      case 2:
         if (M(0))   {                                                            y_new = 0; }
         else
         if (X(10))  { SendPubMsg(this, 2); SendPubMsg(this, 9); SendPubMsg(this, 10);
                                                                                  y_new = 3; }
         break;

      case 3:
         if (M(0))   {                                                            y_new = 0; }
         else
         if (!X(10)) {                                                            y_new = 0; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         case 2:
            break;
         case 3:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A6.cpp )*************************************************/
```

П 9. Листинг функции “*Переход-действие*” автомата “Управление приемником магнитных карт” (A7)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

extern CA5 A5;

void CA7::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         if (M(11))         { Z(19); Z(20);                    y_new = 1; }
         break;

      case 1:
         if (X(11))         { Z(23);                           y_new = 2; }
         else
         if (!X(11))        { Z(21); SendPubMsg(this, 10);     y_new = 0; }
         break;

      case 2:
         if (X(13))         { Z(21); SendPubMsg(this, 10);     y_new = 0; }
         else
         if (X(14))         { Z(22); SendPubMsg(this, 10);     y_new = 0; }
         else
         if (X(12))         { Z(21);                           y_new = 3; }
         break;

      case 3:
         if (!A5.Is_Auto()) {                                  y_new = 0; }
         else
         if (A5.Is_Auto())  { Z(18);                           y_new = 0; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         case 2:
            break;
         case 3:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A7.cpp )*************************************************/
```

П 10. Листинг функции “*Переход-действие*” автомата “Управленеи устройством добавления контролером числа оплаченных поездок” (A8)

```
/*================================[ IMPORT DECLARATIONS ]==================*/
#include "StdAfx.h"
#include "Automata.h"

extern CA5 A5;

void CA8::TA()
{
   if (LOGGING && A_BEGINS_LOGGING) ABeginsLogging(y_cur, stNames[y_cur]);

   switch (y_cur)
   {
      case 0:
         // A5.Is_Auto() ó y5 == 0
         if (X(10) && A5.Is_Auto()) { Z(18);   y_new = 1; }
         break;

      case 1:
         if (!X(10))               {          y_new = 0; }
         break;

      default:
         if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_cur);
   }

   if (y_new != y_cur)
   {
      if (LOGGING && A_TRANS_LOGGING) ATransLogging(y_cur, y_new, stNames[y_cur],
stNames[y_new]);

      switch (y_new)
      {
         case 0:
            break;
         case 1:
            break;
         default:
            if (LOGGING && A_ERRORS_LOGGING) AErrorsLogging(y_new);
      }
   }

   if (LOGGING && A_ENDS_LOGGING ) AEndsLogging(y_new, stNames[y_new]);
   TA_Log();
}
/** (END OF FILE : A8.cpp )*************************************************/
```