Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра "Компьютерные технологии"

С. В. Сытник, Е. Г. Князев, А. А. Шалыто

Моделирование процесса управления ядерным реактором

Объектно-ориентированное проектирование
с явным выделением состояний

Проектная документация

Проект создан в рамках
"Движения за открытую проектную документацию"

http://is.ifmo.ru

Санкт-Петербург
2004

Оглавление

Введение....3
1. Постановка задачи....3
2. Проектирование....4
3. Моделирование....5
4. Структура программного комплекса....7
5. Описание классов, интерфейсов....8
6. Описание логики управления....10
7. Протоколирование....21
8. Использование Switch-технологии....21
9. Литература....24
Приложение 1....25
П 1.1. Пример протоколирования....25
П 1.2. Контрольная панель с протоколом....28
Приложение 2....29
Текст программы....29

Введение

Предлагаемая проектная документация описывает учебный пример использования SWITCH-технологии при разработке программы управления ядерным реактором.

Для программирования задач логического управления была предложена SWITCH-технология, которая в дальнейшем была развита применительно к событийным и объектно-ориентированным программам. Подробно ознакомиться с этой технологией и с конкретными примерами ее использования можно на сайте http://is.ifmo.ru.

Эта технология удобна для задач управления объектами со сложным поведением, поскольку при использовании автоматного подхода, в частности, удается повысить централизацию логики управления. Другое достоинство этого подхода состоит в том, что код автоматных функций является изоморфным графу переходов автоматов, по которому этот код строился (подробнее это описано в разделе Visio2Switch). Поэтому появляется возможность не обращаться к текстам программ для того, чтобы понять, как они работают. Для этой цели достаточно рассмотреть соответствующие графы переходов.

Целью данного проекта является построение модели ядерного реактора и его системы управления. Модель «реактор – система управления» предназначена для проведения испытаний работы в различных режимах, в том числе аварийных.

Предлагаемая модель является учебной и не претендует на полноту описания процесса управления ядерным реактором. Авторы предполагают, что описанный подход может применяться при проектировании, реализации и тестировании реальных систем управления.

При разработке модели применен метод объектно-ориентированного проектирования и программирования с явным выделением состояний [1]. Этот подход позволяет формализовать написание программного кода и вывод трассировочной информации в терминах автоматов и состояний.

Выполненный проект содержит полную документацию по всем элементам проекта, включая спецификацию программы (описание входов и выходов, графов переходов), исходный код (за исключением графических файлов), а также полный отчет о работе программы.

Достоинствами использованного подхода применительно к задаче проектирования системы управления ядерным реактором являются:

- предсказуемость работы программы (благодаря точному соответствию логики работы программы графу переходов);
- полные формальные отчеты в терминах автоматов о поведении программы, в том числе и в критических ситуациях;
- возможность внесения изменений с сохранением гарантии работоспособности.

Описанный подход, являющийся развитием SWITCH-технологии, подробно описан в [1].

1. Постановка задачи

Требуется разработать систему моделирования работы ядерного реактора. Система моделирования должна включать в себя модель ядерного реактора с автоматическим управлением и контролем параметров реакции.

2. Проектирование

Требуется разработать программу, позволяющую в наглядной форме продемонстрировать процесс работы ядерного реактора с системой управления.

Разработка программы выполнена при помощи следующих инструментов:

- язык *C++* в среде разработки *Microsoft Visual C++ 6.0*;
- библиотека *MFC* (программа требует наличия соответствующих dll-файлов);
- *Visio2Switch* и *Microsoft Visio*.

В программе требуется обеспечить возможность подачи команд оператором и моделирования нештатных ситуаций. Программа создается для работы под управлением операционной системы *Windows*.

Интерфейс пользователя представлен на рис. 1.

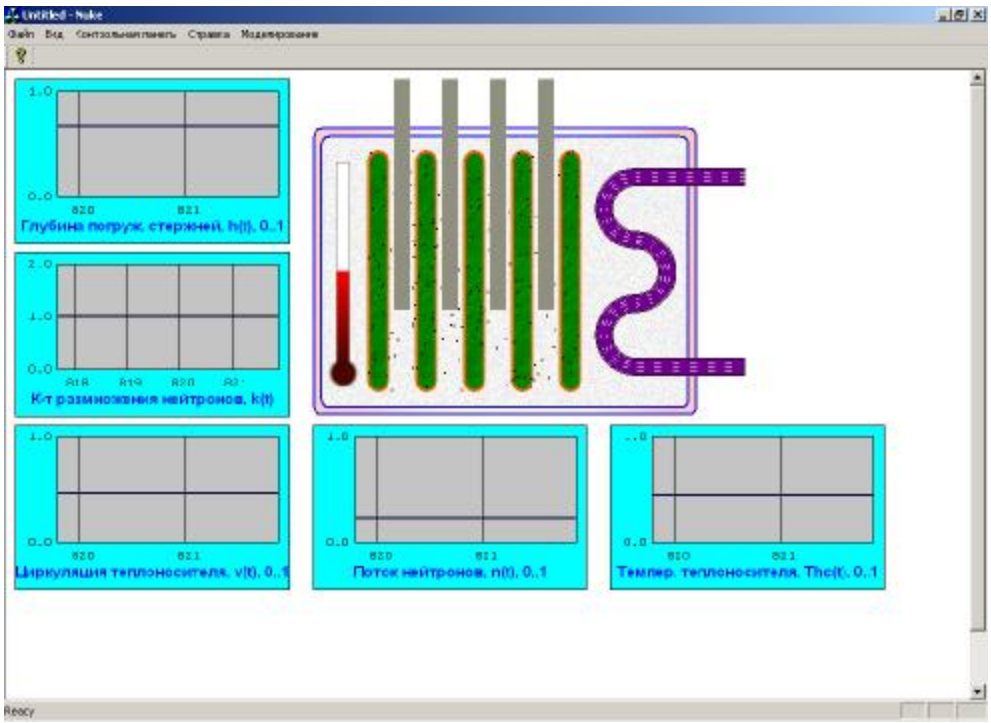

Рис 1. Интерфейс пользователя

2.1. Функциональная спецификация системы

Система управления должна выполнять следующие функции:

- оперативно реагировать на изменения условий протекания реакции путем выработки необходимых корректирующих воздействий;
- гарантировать поддержание всех важных параметров в допустимых пределах;
- при выходе параметров за пределы нормального диапазона попытаться восстановить нормальный режим работы;
- если параметры работы реактора таковы, что ситуация не может быть исправлена, производить экстренный останов реактора.

В критических ситуациях управления на себя должен брать блок защиты системы.

2.2. Структура системы управления

Состав системы
- блок контроля за ядерным реактором;
- блок, запускающий реактор;
- блок, останавливающий реактор;
- блок защиты.

Взаимодействие блоков системы
- блок контроля через интерфейс реактора получает от него (или от его модели) данные о состоянии;
- если произошла поломка (реактор находится в одном из критических состояний), необходимо передать управление блоку защиты системы;
- блок контроля на основе данных о реакторе (путем сравнения величин с пороговыми) формирует выходное воздействие – «поднять» или «опустить» стержни.

3. Моделирование

В поставленной задаче моделируется ядерный реактор как физическая система с непрерывными параметрами и система управления стержнями и теплоносителем. В определенные моменты времени система управления вырабатывает сигналы на поднятие или опускание стержней в активную зону, ускорение или замедление циркуляции теплоносителя.

Математическая модель ядерного реактора построена на основе известных из открытых источников базовых представлений о работе реактора и не претендует на полноту описания процесса [2-4].

В качестве входных данных модели используется множество констант, задающих параметры реакции, пороговые значения контролируемых параметров и т.д. Полное описание констант находится в исходном коде программы.

Ниже приведена спецификация математической модели ядерного реактора. В качестве параметров реакции используются:

входные параметры:
- time – текущее время реактора (модели);
- h – глубина погружения стержней, в активную зону;
- v – скорость подачи теплоносителя;

выходные величины:
- k – коэффициент размножения нейтронов;
- n – поток нейтронов активной зоны;
- Thc – температура теплоносителя;
- Twa – температура рабочей зоны;
- N – полная тепловая мощность реактора, выделяющаяся в процессе распада ядер;
- P – полезная мощность реактора (электрическая).

Система управления имеет возможность изменять глубину погружения стержней в активную зону, а также скорость циркуляции теплоносителя. В зависимости от положения стержней и скорости подачи теплоносителя зависят выходные параметры реактора, что и должна продемонстрировать модель.

Ниже приведен список параметров модели с указанием их взаимозависимостей:

$k = f_1(h,...)$ – функция для расчета коэффициента размножения нейтронов, которая зависит не только от глубины погружения стержней в активную зону, физических и геометрических характеристик активной зоны, ядерного топлива и т.д;

$n = f_2(k, n_{t-dt},...)$ – функция для расчета потока нейтронов, которая зависит от множества дополнительных параметров, кроме коэффициента размножения и количества нейтронов в предыдущем поколении;

$N = f_3(n, \mu, \varphi,...)$ – тепловая мощность ядерной реакции, которая рассчитывается исходя из общего числа нейтронов в данном поколении и множества специфических констант;

В виду того, что КПД реактора непостоянен и зависит от множества условий, то рассчитывать полезную мощность будем следующим образом:

$P = \eta_{hc} N_{hc}$, где η_{hc}, N_{hc} – коэффициент полезного действия использования энергии теплоносителя и тепловая мощность, переносимая теплоносителем;

$Thc = f_4(Twa, N,...)$ – температура теплоносителя, которая определяется на основе решения уравнения теплообмена и зависит от тепловыделения в активной зоне и некоторых других параметров;

$Twa = f_5(N, N_{hc},...)$ – температура активной зоны, которая зависит от притока тепловой энергии N в процессе реакции и оттока тепла, уносимого теплоносителем.

Конкретные реализации этих функций находятся в программном коде, приведенном в приложении 2.

Модель «реактор-система управления» является гибридной, так как она сочетает в себе особенности как континуальных, так и дискретных систем [5].

4. Структура программного комплекса

Структура разработанного программного комплекса, реализующего модель, приведена на рис. 2.

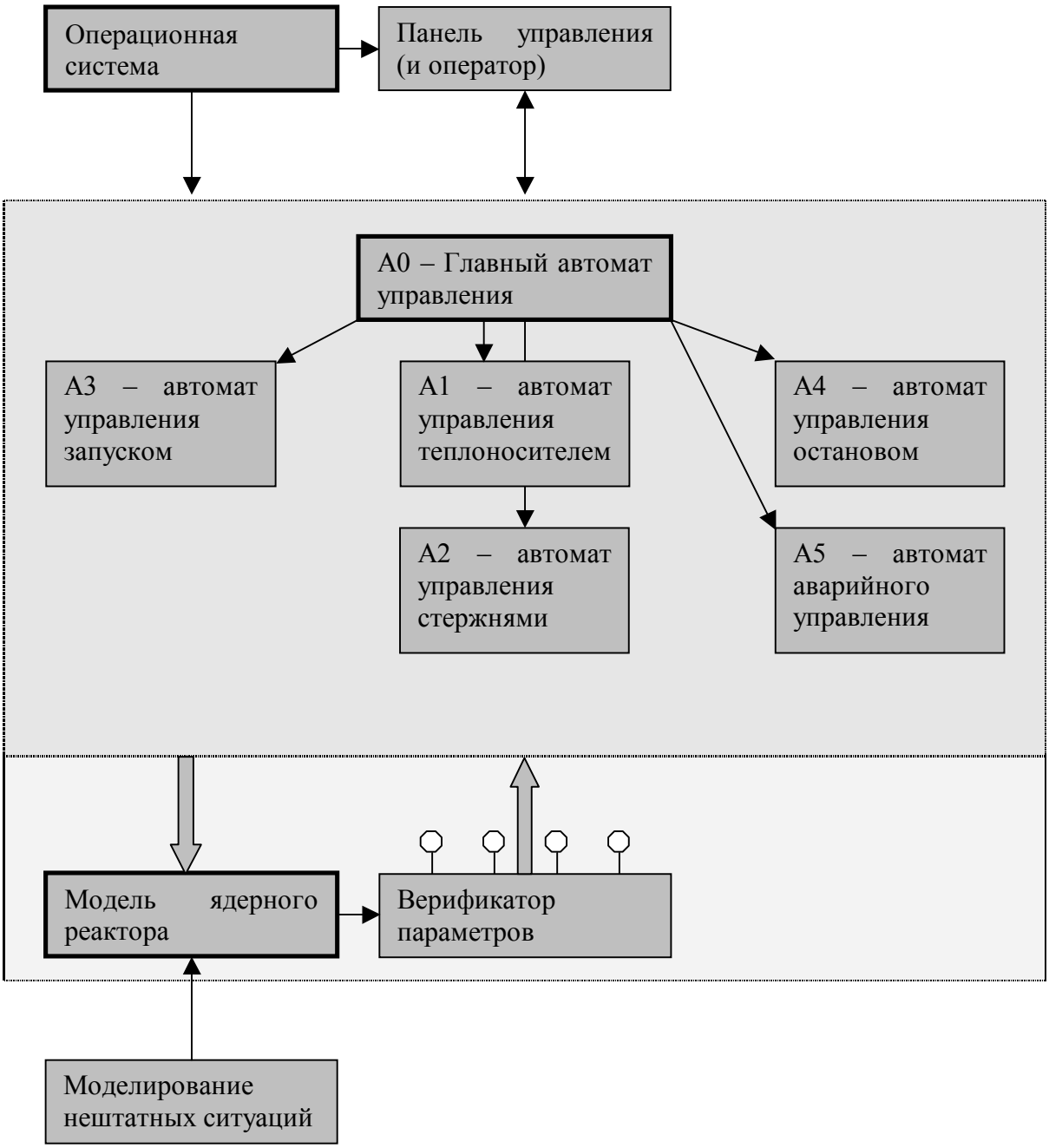

Рис. 2. Структура программного комплекса

5. Описание классов, интерфейсов

Приложение написано с использованием библиотеки *MFC*. Применена архитектура *document-view*. Это означает, что вводятся специальные классы для работы с документом и его представлением. В рассматриваемом примере это класс *CNukeView* (файл *NukeView.h*), который хранит необходимые данные и реализует обработчики событий для функционирования интерфейса программы.

Классы *CMainFrame* (файл *MainFrm.h*), *CControlPanelMain* (файл *ControlPanelMain.h*) реализуют формы для отображения информации и являются наследниками библиотечных классов *CFrameWnd*, *CDialog*.

Класс *NukeGraphs* (файл *nukeGraphs.h*), являющийся наследником класса *GraphBase* (файл *graphBase.h*), используется для визуализации графиков изменения параметров.

Схема взаимодействия классов представлена на рис. 3.

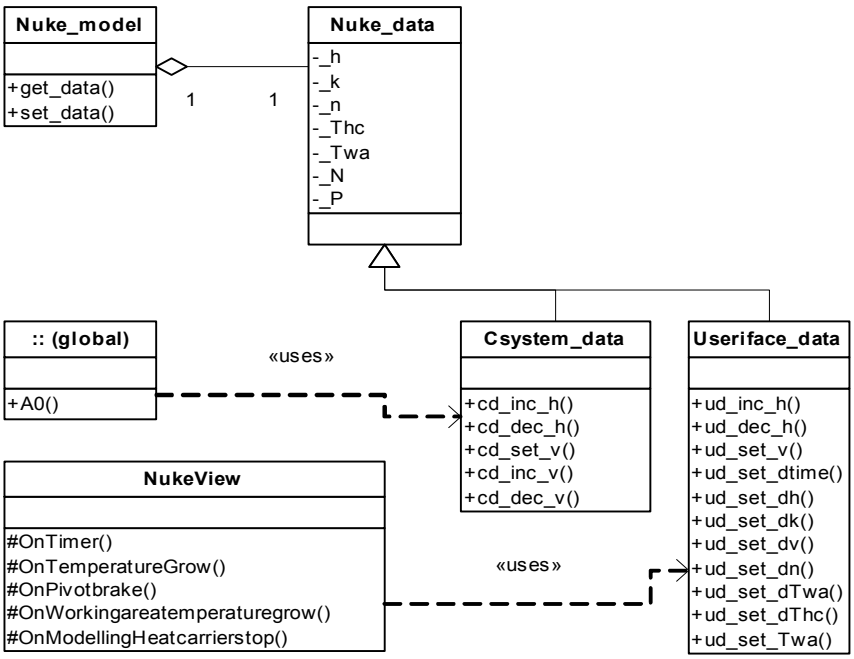

Рис. 3. Схема взаимодействия классов программы

Структура *Nuke_data* (файл *Nuke_data.h*) представляет собой набор данных, достаточных для отслеживания всех параметров системы. Структура *Nuke_data* организована таким образом, что доступ к полям класса можно получить только через специальные методы. К примеру, доступ к параметру *_h* (глубина погружения стержней) можно получить при помощи встраиваемого (*inline*) константного метода `inline double h() const`. Методы доступа на чтение параметров сделаны общедоступными (*public*), тогда как методы модификации параметров – защищенными (*protected*).

Структуры *Csystem_data*, *Useriface_data* (файл *Nuke_data.h*) унаследованы от структуры *Nuke_data*. В этих классах на полный доступ открыты только необходимые методы модификации параметров. Так, например, структура *Csystem_data* имеет методы модификации параметров `cd_inc_h()`, `cd_dec_h()`, увеличивающие и уменьшающие значение параметра *h* (глубина погружения стержней) на заданную величину. Аналогично устроена и структуры *Useriface_data*, в которой реализованы методы модификации параметров, необходимые для пользователя.

Назначение у структур следующее:

- *Csystem_data* предназначена для управления реактором (выдвижением стержней, установкой скорости циркуляции теплоносителя);
- *Useriface_data* предназначена для того, чтобы пользователь мог вмешаться в процесс моделирования, создавая внештатные ситуации (прекратилась циркуляция теплоносителя, сломался стержень замедлителя и т.п.).

Классы в программе взаимодействуют следующим образом:

- при получении события от таймера класс *CNukeView* вызывает метод *GetModelData*, который в свою очередь вызывает метод расчетов `execute_for()` и запрашивает данные у модели – класса *Nuke_model* (файл *Nuke_model.h*), а затем производится визуализация этих данных;
- при внесении пользователем возмущения в систему (поломка стержня, остановка циркуляции теплоносителя и т.п.) вызывается соответствующий метод (например, `OnPivotBrake()` для поломки стержня), который при помощи «оборачивания» структуры данных *Nuke_data* в структуру *Useriface_data* производит все необходимые изменения, а затем вызывается метод `SetModelData()` для обновления данных модели;
- получение данных модели автоматами системы управления происходит в подпрограммах опроса входных переменных от класса *Nuke_model*;
- изменение данных модели производится при помощи структуры *Csystem_data* в выходных воздействиях автоматов.

Заметим, что класс *Nuke_model* реализован паттерном *singleton*, что позволило избежать множества проблем с передачей ссылок. Такая реализация делает класс *Nuke_model* объектом.

6. Описание логики управления

Логика управления сосредоточена в системонезависимой автоматной части. Автоматы реализованы не как классы и не как методы классов, а как отдельные функции, так как для автоматической генерации кода автоматов была использована программа *Visio2Switch*.

6.1. События, входные и выходные воздействия, их нумерация

E		X		Z	
e0	Инициализация автомата	x10	Т-ра[1] критически низкая	z100	Сделать кнопку «СТАРТ» недоступной
e10	Системный таймер	x11	Т-ра ниже нормы	z101	Сделать кнопку «СТОП» недоступной
e100	Нажатие кнопки «Пуск»	x12	Т-ра в норме	z102	Включить аварийный звуковой сигнал
e101	Нажатие кнопки «Стоп»	x13	Т-ра выше нормы	z200	Увеличить скорость т-ля (понизить т-ру)
		x14	Т-ра критически превышена	z201	Уменьшить скорость т-ля (повысить т-ру)
		x20	Число н-нов[2] критически низкое	z210	Увеличить глубину погружения стержней (понизить число н-нов)
		x21	Число н-нов ниже нормы	z211	Уменьшить глубину погружения стержней (повысить число н-нов)
		x22	Число н-нов в норме	z220	Обнуление счетчика выхода из неустойчивого состояния для A1
		x23	Число н-нов выше нормы	z221	Обнуление счетчика выхода из неустойчивого состояния для A2
		x24	Число н-нов критически превышено	z230	Обнуление счетчика готовности третьих систем
		x30	Требуется выход из неустойчивого состояния для A1	z310	Инициализация A1: вызов A1(e0)
		x31	Требуется выход из неустойчивого состояния для A2	z311	Работа A1: вызов A1(e10)
		x40	Готовность третьих систем	z320	Инициализация A2: вызов A2(e0)
		x50	Скорость т-ля[3] >= начальной скорости теплоносителя	z321	Работа A2: вызов A2(e10)
		x60	Скорость т-ля < требуемой скорости т-я для останова	z330	Инициализация A3: вызов A3(e0)
		x70	Т-ра <= Т-ры останова	z331	Работа A3: вызов A3(e10)
		x80	Скорость т-ля максимальна	z340	Инициализация A4: вызов A4(e0)
				z341	Работа A4: вызов A4(e10)
				z350	Инициализация A5: вызов A5(e0)
				z351	Работа A5: вызов A5(e10)

[1] Здесь и далее – температура
[2] Здесь и далее – нейтронов
[3] Здесь и далее – теплоносителя

6.2. Главный автомат управления реактором (A0)

Словесное описание автомата

Главный автомат управления реактором A0 получает управление с различными событиями (таймер, нажатия клавиш) от обработчиков системных событий программы. После этого он вызывает соответствующие автоматы. Затем вырабатываются выходные воздействия.

Автомат является посредником между системозависимой и системонезависимой частями программы. Его состояния соответствуют состояниям реактора. Анализируя входные события и состояния вызываемых им автоматов, автомат A0 принимает решение о запуске реактора, его переходе в режим работы или остановки (штатной или аварийной).

Схема связей

Схема связей автомата A0 приведена на рис.4.

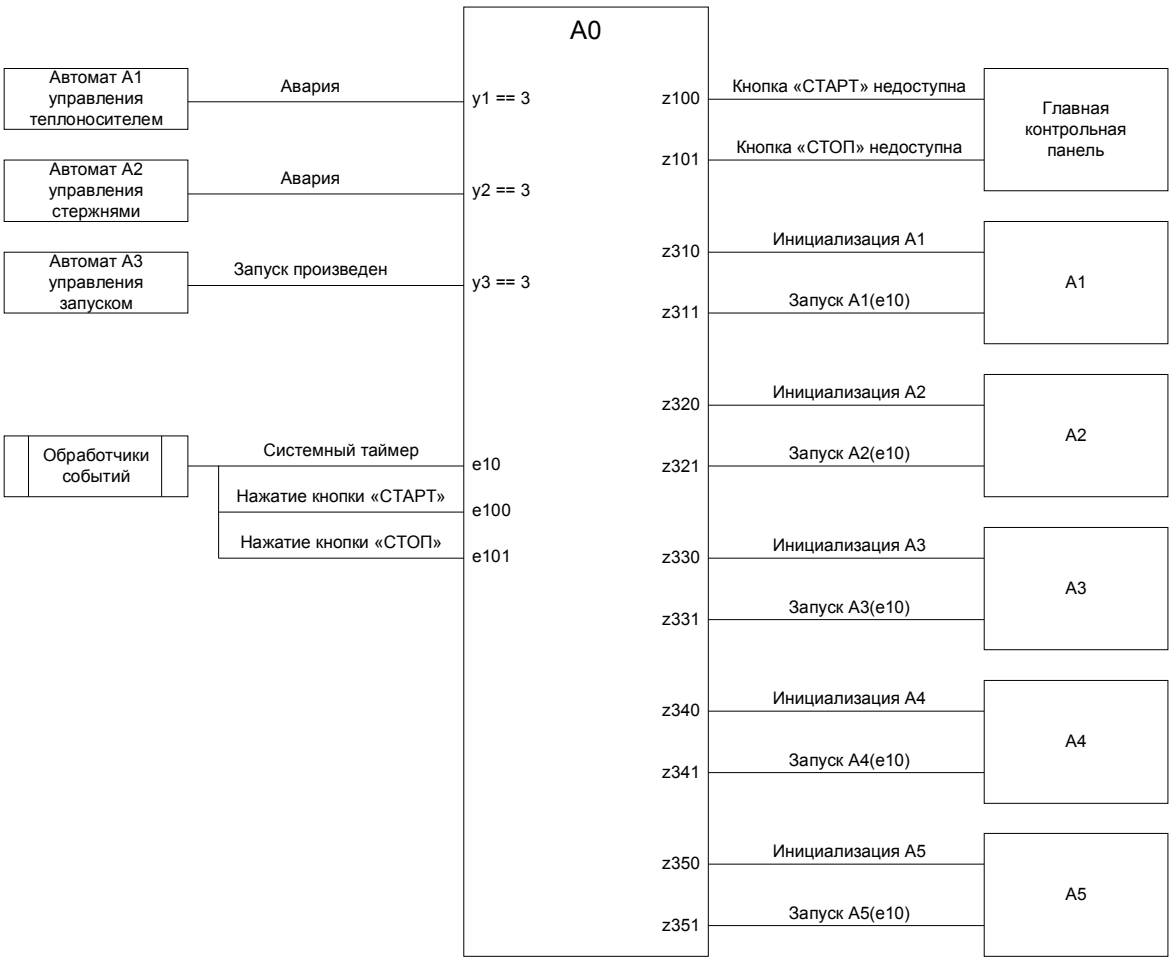

Рис.4. Схема связей автомата A0

Граф переходов

Граф переходов автомата A0 приведен на рис.5.

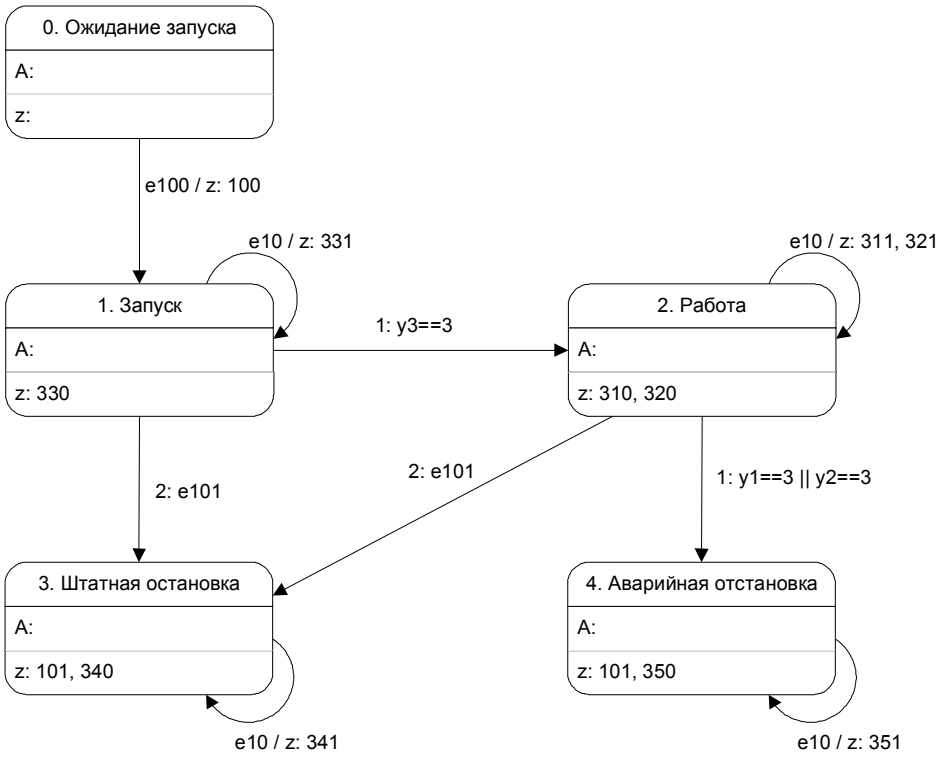

Рис. 5. Граф переходов автомата A0

6.3. Автомат управления теплоносителем (A1)

Словесное описание автомата

Как следует из названия, этот автомат управляет теплоносителем. Для этого он использует информацию о температуре и, частично, о количестве нейтронов. Логика этого автомата имеет схожую структуру с логикой автомата A2, управляющего стержнями. В качестве выходных воздействий в автомате используются функции «увеличить скорость теплоносителя» и «уменьшить скорость теплоносителя».

Схема связей

Схема связей автомата A1 приведена на рис.6.

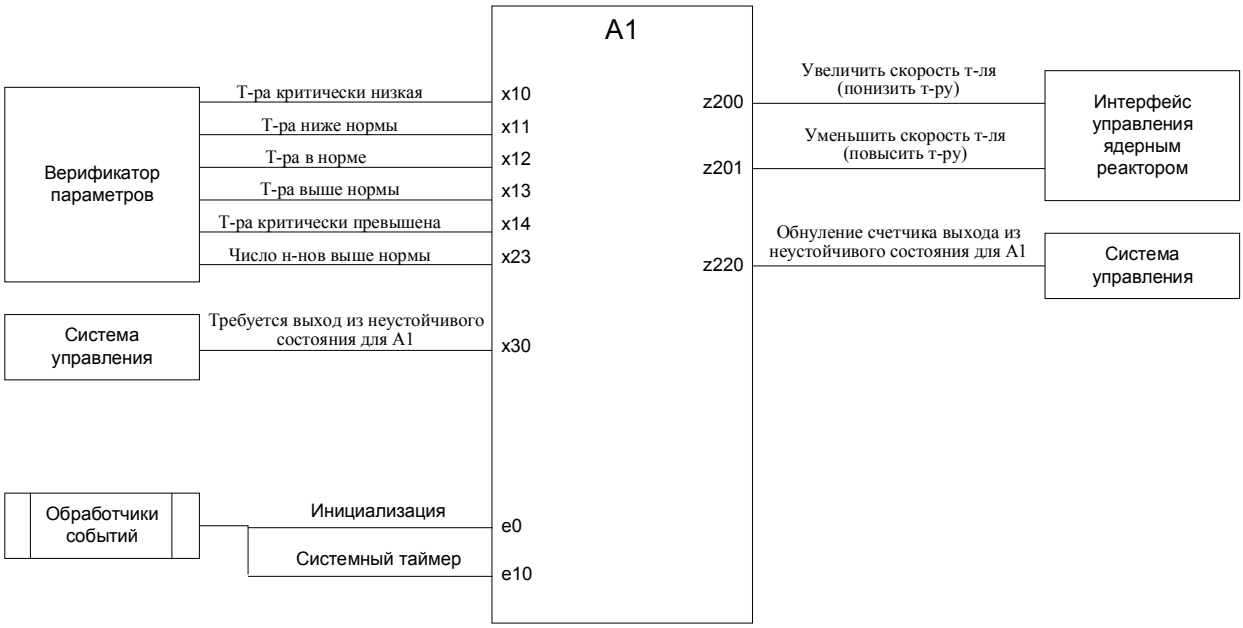

Рис. 6. Схема связей автомата A1

Граф переходов

Граф переходов автомата A1 приведен на рис. 7.

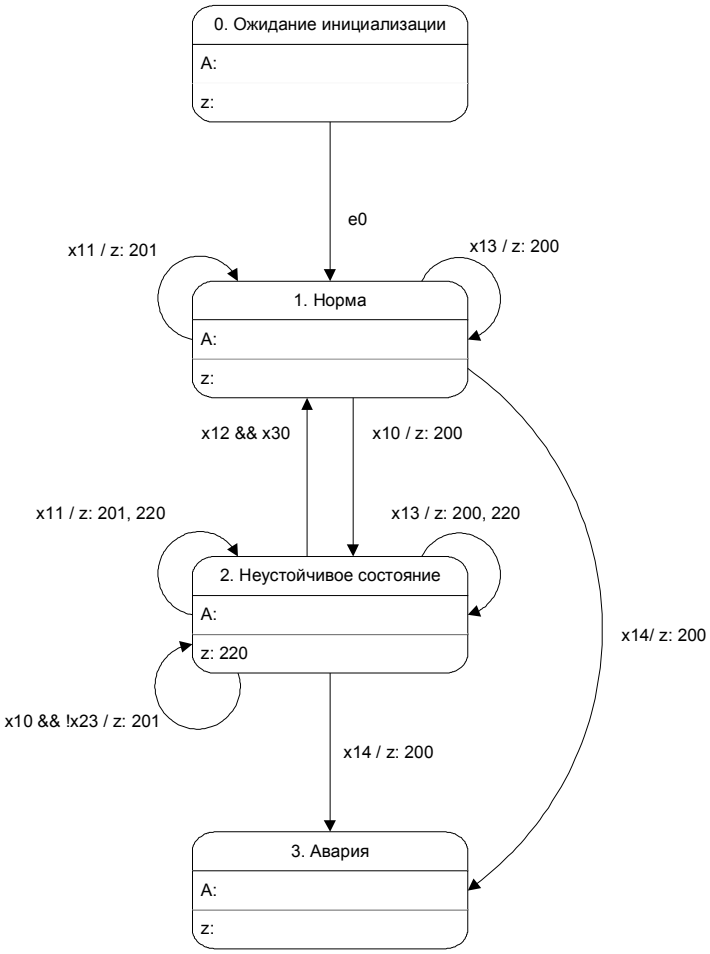

Рис. 7. Граф переходов автомата A1

6.4. Автомат управления стержнями (A2)

Словесное описание автомата

Этот автомат управляет стержнями. Для этого он использует информацию о количестве нейтронов и о температуре. Логика этого автомата имеет схожую структуру с логикой автомата A1. В качестве выходных воздействий в автомате используются функции «увеличить глубину погружения стержней» и «уменьшить глубину погружения стержней».

Схема связей

Схема связей автомата A2 приведена на рис. 8.

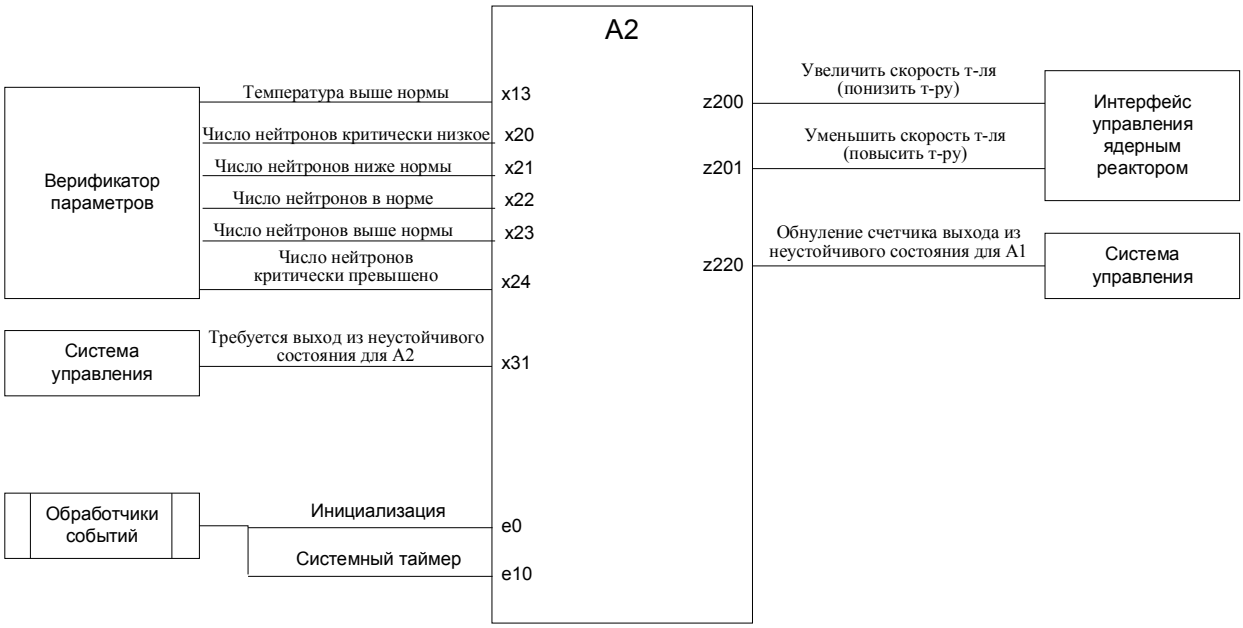

Рис. 8. Схема связей автомата A2

Граф переходов

Граф переходов автомата A2 приведен на рис. 9.

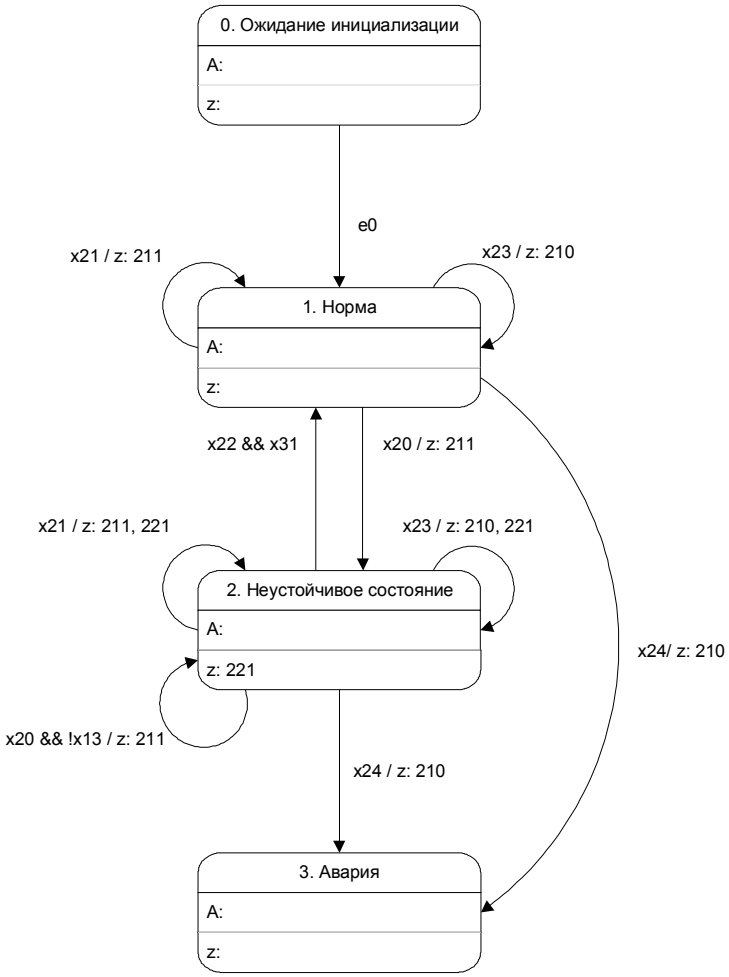

Рис. 9. Граф переходов автомата A2

6.5. Автомат управления запуском (A3)

Словесное описание автомата

Этот автомат вызывается функциями z330, z331 из автомата A0, когда его состояние соответствует запуску реактора. Автомат A3 отвечает за действия, связанные с запуском реактора: он ничего не делает до тех пор, пока не будет произведена предпусковая инициализация третьих (не рассматриваемых в работе) систем (долговременные операции, такие как, разогрев труб). После этого автомат обеспечивает начальный разгон теплоносителя до определенной скорости с тем, чтобы автомат A0 мог перейти в состояние «Работа». В этом состоянии начинается собственно управление, и скорость теплоносителя должна быть заведомо достаточно большой, чтобы не произошла авария. В дальнейшем автоматы управления теплоносителем и стержнями понизят эту скорость до необходимой для работы.

Схема связей

Схема связей автомата A3 приведена на рис. 10.

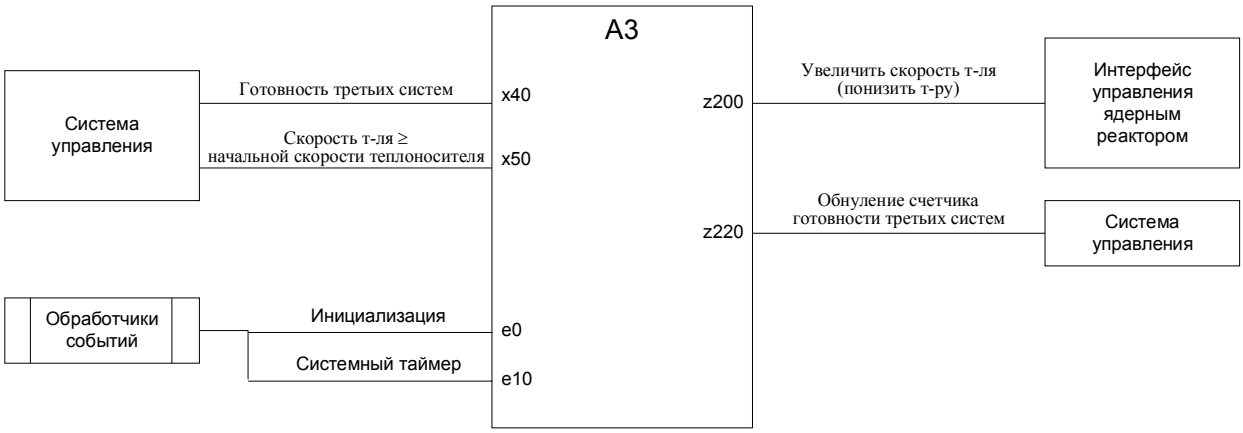

Рис. 10. Схема связей автомата A3

Граф переходов

Граф переходов автомата A3 приведен на рис. 11.

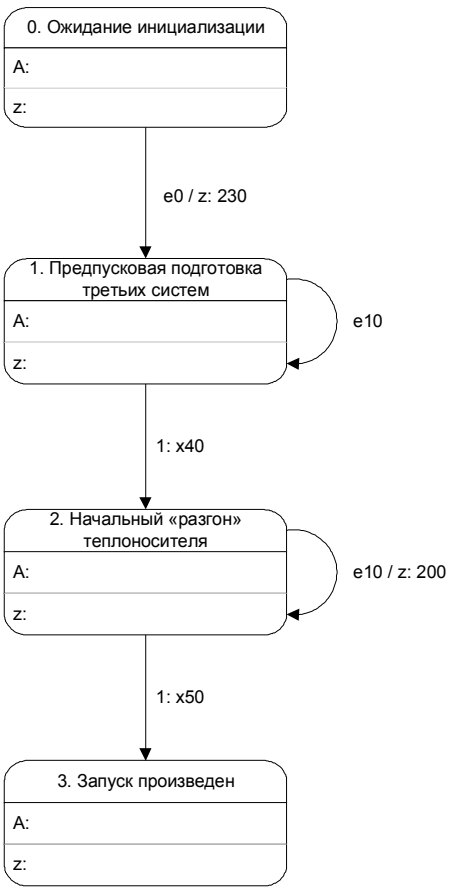

Рис. 11. Граф переходов автомата A3

6.6. Автомат управления остановом (A4)

Словесное описание автомата

Этот автомат получает управление от автомата A0 в состоянии штатного останова реактора, которое происходит в случае нажатия оператором специальной кнопки на панели управления. Логика управления достаточно проста: сначала производится опускание стержней до максимума, затем реактор охлаждается (для этого теплоноситель разгоняется), а потом производится торможение теплоносителя. Автомат не имеет «заключительного» состояния, поскольку подразумевается, что в силу неучтенных факторов температура активной зоны может повыситься, и придется увеличивать скорость теплоносителя, чтобы компенсировать это. Другая причина заключается в том, что ядерный реактор – объект, торможение которого должно производиться в достаточно редких случаях и этот процесс занимает продолжительное время. Поэтому система не рассчитана на переинициализацию своих внутренних переменных и возврат к первоначальному состоянию, поскольку в этом случае система просто может быть перезагружена. Это соответствует перезапуску программы-модели.

Схема связей

Схема связей автомата A4 приведена на рис. 12.

Рис. 12. Схема связей автомата A4

Граф переходов

Граф переходов автомата A4 приведен на рис. 13.

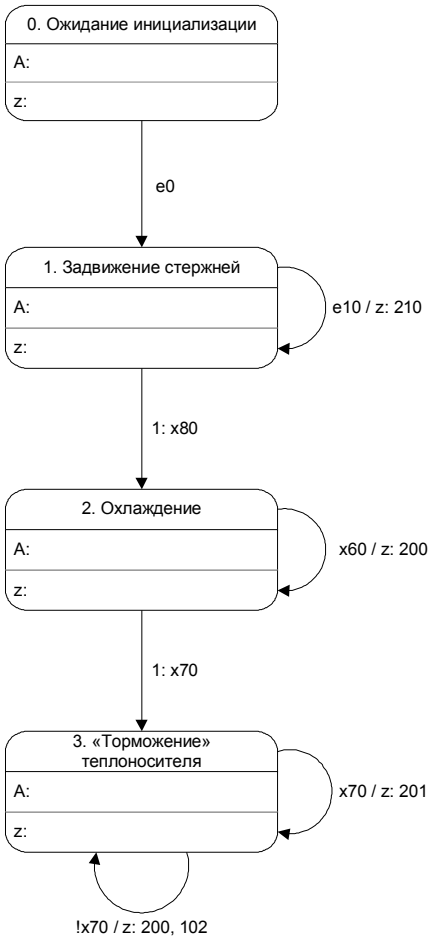

Рис. 13. Граф переходов автомата A4

6.7. Автомат аварийного управления остановом (A5)

Словесное описание автомата

Этот автомат, также как и автомат A4, управляет остановом, однако разница в том, что здесь останов экстренный. При первой передаче управления этому автомату включается аварийный звуковой сигнал. После этого автомат производит экстренные действия, связанные с быстрой нейтрализацией последствий факторов, вызвавших аварийную ситуацию. При этом теплоноситель разгоняется до максимально возможной скорости, поскольку наибольшую опасность представляет перегрев активной зоны реактора.

Схема связей

Схема связей автомата A5 приведена на рис. 14.

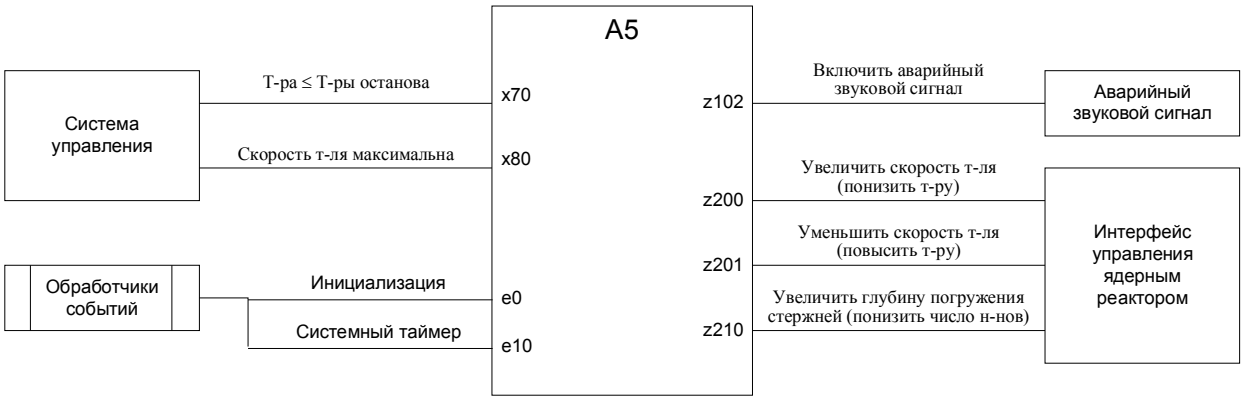

Рис. 14. Граф переходов автомата A5

Граф переходов

Граф переходов автомата A5 приведен на рис. 15.

Рис. 15. Граф переходов автомата A5

7. Протоколирование

В программе реализовано протоколирование всех действий автоматов, что немаловажно при управлении такой системой, как ядерный реактор.

Протоколирование выполнено в двух видах.

Первый из них - это специальным образом размеченный текст, пример которого приведен в Приложении 1. Для получения протокола необходимо запустить приложение, перенаправив вывод в файл. Сделать это можно, например, такой командой:

```
nuke.exe > protocol.txt
```

Второй вид протокола отражает панель, приведенная на рис. 14, на которой отображаются номера и названия состояний всех автоматов в каждый момент времени.

8. Использование Switch-технологии

8.1. Visio2Switch

В процессе разработки авторами использовалась очень удобная программа *Visio2Switch*, которая размещена на сайте http://is.ifmo.ru. Предназначение этой программы – избавить разработчика от утомительного процесса написания автоматной части программы по нарисованному в *Visio* графу переходов автоматов.

При разработке аналогичных проектов эта программа, по всей видимости, используется не очень широко, поэтому хотелось бы рассмотреть некоторые ее достоинства и недостатки.

Простая в использовании, имеющая документацию, она может сэкономить много времени, потому, что при переносе автоматов могут быть ошибки, а так программа на языке C (возможно, в будущем это будет не только C, а, например, Java) достается достаточно дешево – необходимо только перезапустить программу. Правда, при этом неизбежно будут перетираться файлы, и, хотя предусмотрено разделение всех файлов на «пользовательскую» и «непользовательскую» части с тем, чтобы не перезаписывать измененные вручную файлы, – это спасает не всегда.

Также с помощью рассматриваемой программы достаточно просто получить протокол работы автоматов, указав его вид таким, каким нравится (по умолчанию в комментариях предложен вид протокола, который используется авторами). Только для этого надо не забыть для каждого события, входного и выходного воздействий вставить специальный комментарий. В этой связи, кстати, хотелось бы иметь предупреждения (возможно, опциональные) о том, что недостает комментариев для протокола.

Рассмотрим детали, которые кажутся недостатками:

- определенное неудобство представляет то, что протоколирование факта вызова выходного воздействия производится после вызова функции выходного воздействия из «пользовательского» файла, а не до него. Это может ввести в заблуждение того, кто будет читать протокол. Целесообразным представляется поменять местами вызов и протоколирование вызова. Конкретный пример этой ситуации в настоящем проекте – вызываемые автоматы. Факт их вызова выводится в протокол после того, как автомат завершит свое выполнение;
- не очень удобно пересоздавать файлы (это не хочется делать часто, и небольшие изменения проще вставить в программу вручную, чем запускать *Visio2Switch*). Причина – необходимость изменять выходные файлы. Например, подключать свои `.h`-

файлы, или переносить определение структур из `cpp`-файлов в `.h`-файлы с тем, чтобы можно было использовать их из других файлов. Это действительно бывает нужно, например, при нестандартном протоколировании в дополнение к обычному, когда форма при своем обновлении запрашивает информацию из `log.h` и `.cpp`. В процессе слияния дополнений, сделанных вручную и новых файлов очень удобно использовать такие программы, как, например, *CVS* (http://www.cvsgui.org) и *WinMerge* (http://winmerge.sourceforge.net). Эта проблема порождает еще одну. Поскольку хочется избежать слишком частых пересозданий файлов, иногда проще сделать изменения вручную;

- к сожалению, в программе не предусмотрены вызываемые автоматы, поэтому реализовывать их приходится введением двух специальных выходных воздействий – инициализации и собственно вызова автомата из выходного воздействия. В процессе проектирования для этих воздействий использовались такие имена, как `z_Ai_j` (вызов автомата `Ai` с событием `ej`), чтобы не перепутать. Впоследствии эти имена были заменены цифрами;
- выходные воздействия никак не обертываются классами, и получаемые функции и переменные становятся глобальными, что в рамках объектно-ориентированного подхода является не самым лучшим решением;
- не проверяются (или, может быть, проверяются, но не полностью) факт наличия лишних знаков или отсутсвия требуемых знаков, предусмотренных синтаксисом *Visio2Switch* (в файлах *Visio*). Проверить это было бы достаточно просто. Сейчас же пользователи могут совершить ошибку, которую потом нелегко найти. В результате такой ошибки, например, создается новое выходное воздействие, или предусмотренное не создается вообще.

Впрочем, как уже говорилось, эти недостатки не мешают с успехом использовать программу для получения кода автоматных функций.

8.2. Недостатки SWITCH-технологии

Хотелось бы посвятить этот раздел тем недостаткам SWITCH-технологии, которые при активном ее использовании достаточно очевидны, и это хотелось бы осветить здесь.

Во-первых, вложенные и вызываемые автоматы.

Вложенные автоматы представляется удобным использовать только тогда, когда они выполняют отдельную небольшую задачу, которую требуется делать каждый раз при входе в конкретное состояние. Однако если автомат необходимо вызывать только на некоторых петлях вершины, да еще и с различными событиями, то использование таких автоматов осложняется. В этом случае придется в каждом состоянии на каждой петле проверять номер события, с которым вызван автомат, что достаточно трудоемко при проектировании больших автоматов, так как это приходится делать только для того, чтобы обработать «правильные», нужные автомату события, и пропустить остальные.

Проблему хорошо решают вызываемые автоматы. В готовом виде они в нотации, используемой *Visio2Switch*, не предусматриваются, однако это легко разрешимо путем введения специальных выходных воздействий: инициализация автомата и вызов его с некоторым событием. При этом автомат вызывается точно в нужном месте и с нужным событием, поэтому число проверок на его петлях значительно уменьшается.

Однако есть другая проблема, для которой авторы так и не смогли найти хорошего решения. Предположим, у нас имеются два независимых друг от друга параметра системы, которые могут принимать каждый, скажем, семь различных значений. На каждом шаге выполнения автоматов требуется для обоих параметров выполнять специальные выходные воздействия (различные для параметров и их значений). Варианты решения следующие:

- первое – создать автомат, у которого число петель = 7*7 = 49, каждая из которых вызывает два соответствующих выходных воздействия. Но это решение явно избыточно и неинтересно, потому что при увеличении числа состояний каждого параметра, например, до десяти, число петель уже становится равным 100;
- другое решение заключается в том, чтобы управлять параметрами независимо. Но ведь на каждом шаге можно произвести лишь одно действие. Поэтому придется создать два автомата (один на параметр) с одним состоянием и семью петлями каждый, и вызывать из каждой петли какого-то главного автомата оба этих автомата по очереди (либо сделать вложенными). Недостатки такого подхода очевидны: приходится вводить в рассмотрение два (или больше) вырожденных автомата с одним состоянием;
- есть и третье решение, также основанное на независимом управлении параметрами. Достигается это введением отдельного автомата, у которого число состояний будет равно числу параметров (не включая, возможно, инициализирующих и/или деинициализирующих его состояний). При этом в процессе своей работы автомат на каждом шаге будет по очереди «ходить» по этим состояниям, и рассматривать действия для текущего параметра. Это решение представляется лучшим из рассмотренных, но что, если параметры как-то связаны между собой так, что их нужно обязательно поменять на одном шаге? Или же параметров достаточно много, и пропускать шаги не хочется?

Поставленная задача разрешилась сама собой, поскольку в случае ядерного реактора параметры (глубина погружения стержней и скорость теплоносителя) не являются независимыми, кроме того, логика управления ими – более сложная, чем рассмотрена в примере. Тем не менее, этот вопрос остается открытым.

Иногда было удобно получать от автоматов возвращаемые значения. Однако это не предусмотрено, поэтому, когда это необходимо, приходится решать это вызовом специального выходного воздействия (поскольку выходные воздействия не имеют параметров, – для каждого результата различного), сохраняющего результат в специальной переменной, значение которой можно получить, лишь реализуя для проверки значения каждого результата отдельное входное воздействие (x).

9. Литература

1. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. *Белоус В.* Ядерные испытания – без взрывов. // Ядерная безопасность #15-16. http://www.cwpj.org/Publications/nucrep/n15-16/10.htm
3. *Зеленцов Б.П.* Математические модели на основе размножения и гибели объектов // Соросовский образовательный журнал, том 7, №6, 2001. стр. 92-97
4. *Капустин М.А.* Модель изучения технических знаний в приложении к задаче управления сложными системами. ИПМ РАН, Россия.
5. *Бенькович Е., Колесов Ю., Сениченков Ю.* Практическое моделирование динамических систем. СПб.: БХВ-Петербург, 2002.

Приложение 1

Ниже приведен полный протокол работы системы управления ядерным реактором. Повторяющиеся и не представляющие интереса действия были пропущены.

П 1.1. Пример протоколирования

A0 (Главный автомат управления реактором): в состоянии 0 (Ожидание запуска) запущен с событием e10 (Системный таймер)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
A0 (Главный автомат управления реактором): в состоянии 0 (Ожидание запуска) запущен с событием e10 (Системный таймер)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 0 (Ожидание запуска) запущен с событием e100 (Нажатие кнопки "Пуск")
z100. Сделать кнопку "СТАРТ" недоступной
A0 (Главный автомат управления реактором): перешел из состояния 0 (Ожидание запуска) в состояние 1 (Запуск)
A3 (Автомат управления запуском): в состоянии 0 (Ожидание инициализации) запущен с событием e0 (_инициализация_)
z230. Обнуление счетчика готовности третьих систем
A3 (Автомат управления запуском): перешел из состояния 0 (Ожидание инициализации) в состояние 1 (Предпусковая подготовка третьих систем)
A3 (Автомат управления запуском): завершил обработку события e0 (_инициализация_)
z330. Инициализация A3: вызов A3(e0)
A0 (Главный автомат управления реактором): завершил обработку события e100 (Нажатие кнопки "Пуск")

A0 (Главный автомат управления реактором): в состоянии 1 (Запуск) запущен с событием e10 (Системный таймер)
A3 (Автомат управления запуском): в состоянии 1 (Предпусковая подготовка третьих систем) запущен с событием e10 (Системный таймер)
x40 - Готовность третьих систем - вернул 0
A3 (Автомат управления запуском): завершил обработку события e10 (Системный таймер)
z331. Работа A3: вызов A3(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
A0 (Главный автомат управления реактором): в состоянии 1 (Запуск) запущен с событием e10 (Системный таймер)
A3 (Автомат управления запуском): в состоянии 1 (Предпусковая подготовка третьих систем) запущен с событием e10 (Системный таймер)
x40 - Готовность третьих систем - вернул 1
A3 (Автомат управления запуском): перешел из состояния 1 (Предпусковая подготовка третьих систем) в состояние 2 (Начальный <разгон> теплоносителя)
A3 (Автомат управления запуском): завершил обработку события e10 (Системный таймер)
z331. Работа A3: вызов A3(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 1 (Запуск) запущен с событием e10 (Системный таймер)
A3 (Автомат управления запуском): в состоянии 2 (Начальный <разгон> теплоносителя) запущен с событием e10 (Системный таймер)
x50 - Скорость теплоносителя>= начальной скорости теплоносителя - вернул 0
z200. Увеличить скорость теплоносителя (понизить температуру)
A3 (Автомат управления запуском): завершил обработку события e10 (Системный таймер)
z331. Работа A3: вызов A3(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
A0 (Главный автомат управления реактором): в состоянии 1 (Запуск) запущен с событием e10 (Системный таймер)
A3 (Автомат управления запуском): в состоянии 2 (Начальный <разгон> теплоносителя) запущен с событием e10 (Системный таймер)
x50 - Скорость теплоносителя>= начальной скорости теплоносителя - вернул 1
A3 (Автомат управления запуском): перешел из состояния 2 (Начальный <разгон> теплоносителя) в состояние 3 (Запуск произведен)
A3 (Автомат управления запуском): завершил обработку события e10 (Системный таймер)
z331. Работа A3: вызов A3(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 1 (Запуск) запущен с событием e10 (Системный таймер)
A0 (Главный автомат управления реактором): перешел из состояния 1 (Запуск) в состояние 2 (Работа)
A1 (Автомат управления теплоносителем): в состоянии 0 (Ожидание инициализации) запущен с событием e0 (_инициализация_)
A1 (Автомат управления теплоносителем): перешел из состояния 0 (Ожидание инициализации) в состояние 1 (Норма)
A1 (Автомат управления теплоносителем): завершил обработку события e0 (_инициализация_)
z310. Инициализация A1: вызов A1(e0)
A2 (Автомат управления стержнями): в состоянии 0 (Ожидание инициализации) запущен с событием e0 (_инициализация_)
A2 (Автомат управления стержнями): перешел из состояния 0 (Ожидание инициализации) в состояние 1 (Норма)
A2 (Автомат управления стержнями): завершил обработку события e0 (_инициализация_)
z320. Инициализация A2: вызов A2(e0)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 2 (Работа) запущен с событием e10 (Системный таймер)
A1 (Автомат управления теплоносителем): в состоянии 1 (Норма) запущен с событием e10 (Системный таймер)
x13 - Температура выше нормы - вернул 0

x11 - Температура ниже нормы - вернул 0
x10 - Температура критически низкая - вернул 1
z200. Увеличить скорость теплоносителя (понизить температуру)
A1 (Автомат управления теплоносителем): перешел из состояния 1 (Норма) в состояние 2 (Неустойчивое состояние)
z220. Обнуление счетчика выхода из неустойчивого состояния для A1
A1 (Автомат управления теплоносителем): завершил обработку события e10 (Системный таймер)
z311. Работа A1: вызов A1(e10)
A2 (Автомат управления стержнями): в состоянии 1 (Норма) запущен с событием e10 (Системный таймер)
x23 - Число нейтронов выше нормы - вернул 0
x21 - Число нейтронов ниже нормы - вернул 0
x20 - Число нейтронов критически низкое - вернул 1
z211. Уменьшить глубину погружения стержней (повысить число нейтронов)
A2 (Автомат управления стержнями): перешел из состояния 1 (Норма) в состояние 2 (Неустойчивое состояние)
z221. Обнуление счетчика выхода из неустойчивого состояния для A2
A2 (Автомат управления стержнями): завершил обработку события e10 (Системный таймер)
z321. Работа A2: вызов A2(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 2 (Работа) запущен с событием e10 (Системный таймер)
A1 (Автомат управления теплоносителем): в состоянии 2 (Неустойчивое состояние) запущен с событием e10 (Системный таймер)
x13 - Температура выше нормы - вернул 0
x11 - Температура ниже нормы - вернул 0
x12 - Температура в норме - вернул 0
x14 - Температура критически превышена - вернул 0
x10 - Температура критически низкая - вернул 1
x23 - Число нейтронов выше нормы - вернул 0
z201. Уменьшить скорость теплоносителя (повысить температуру)
A1 (Автомат управления теплоносителем): завершил обработку события e10 (Системный таймер)
z311. Работа A1: вызов A1(e10)
A2 (Автомат управления стержнями): в состоянии 2 (Неустойчивое состояние) запущен с событием e10 (Системный таймер)
x23 - Число нейтронов выше нормы - вернул 0
x21 - Число нейтронов ниже нормы - вернул 0
x22 - Число нейтронов в норме - вернул 0
x24 - Число нейтронов критически превышено - вернул 0
x20 - Число нейтронов критически низкое - вернул 1
x13 - Температура выше нормы - вернул 0
z211. Уменьшить глубину погружения стержней (повысить число нейтронов)
A2 (Автомат управления стержнями): завершил обработку события e10 (Системный таймер)
z321. Работа A2: вызов A2(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
...
...
A0 (Главный автомат управления реактором): в состоянии 2 (Работа) запущен с событием e10 (Системный таймер)
A1 (Автомат управления теплоносителем): в состоянии 2 (Неустойчивое состояние) запущен с событием e10 (Системный таймер)
x13 - Температура выше нормы - вернул 0
x11 - Температура ниже нормы - вернул 0
x12 - Температура в норме - вернул 1
x30 - Требуется выход из неустойчивого состояния для A1 - вернул 1
A1 (Автомат управления теплоносителем): перешел из состояния 2 (Неустойчивое состояние) в состояние 1 (Норма)
A1 (Автомат управления теплоносителем): завершил обработку события e10 (Системный таймер)
z311. Работа A1: вызов A1(e10)
A2 (Автомат управления стержнями): в состоянии 2 (Неустойчивое состояние) запущен с событием e10 (Системный таймер)
x23 - Число нейтронов выше нормы - вернул 0
x21 - Число нейтронов ниже нормы - вернул 1
z211. Уменьшить глубину погружения стержней (повысить число нейтронов)
z221. Обнуление счетчика выхода из неустойчивого состояния для A2
A2 (Автомат управления стержнями): завершил обработку события e10 (Системный таймер)
z321. Работа A2: вызов A2(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
...
...
A0 (Главный автомат управления реактором): в состоянии 2 (Работа) запущен с событием e10 (Системный таймер)
A1 (Автомат управления теплоносителем): в состоянии 1 (Норма) запущен с событием e10 (Системный таймер)
x13 - Температура выше нормы - вернул 1
z200. Увеличить скорость теплоносителя (понизить температуру)
A1 (Автомат управления теплоносителем): завершил обработку события e10 (Системный таймер)
z311. Работа A1: вызов A1(e10)
A2 (Автомат управления стержнями): в состоянии 2 (Неустойчивое состояние) запущен с событием e10 (Системный таймер)
x23 - Число нейтронов выше нормы - вернул 0
x21 - Число нейтронов ниже нормы - вернул 0
x22 - Число нейтронов в норме - вернул 1
x31 - Требуется выход из неустойчивого состояния для A2 - вернул 1
A2 (Автомат управления стержнями): перешел из состояния 2 (Неустойчивое состояние) в состояние 1 (Норма)
A2 (Автомат управления стержнями): завершил обработку события e10 (Системный таймер)
z321. Работа A2: вызов A2(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...

...
...
A0 (Главный автомат управления реактором): в состоянии 2 (Работа) запущен с событием e10 (Системный таймер)
A1 (Автомат управления теплоносителем): в состоянии 1 (Норма) запущен с событием e10 (Системный таймер)
x13 - Температура выше нормы - вернул 1
z200. Увеличить скорость теплоносителя (понизить температуру)
A1 (Автомат управления теплоносителем): завершил обработку события e10 (Системный таймер)
z311. Работа A1: вызов A1(e10)
A2 (Автомат управления стержнями): в состоянии 1 (Норма) запущен с событием e10 (Системный таймер)
x23 - Число нейтронов выше нормы - вернул 0
x21 - Число нейтронов ниже нормы - вернул 0
x20 - Число нейтронов критически низкое - вернул 0
x24 - Число нейтронов критически превышено - вернул 0
A2 (Автомат управления стержнями): завершил обработку события e10 (Системный таймер)
z321. Работа A2: вызов A2(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
...
...
A0 (Главный автомат управления реактором): в состоянии 2 (Работа) запущен с событием e101 (Нажатие кнопки "Стоп")
A0 (Главный автомат управления реактором): перешел из состояния 2 (Работа) в состояние 3 (Штатная остановка)
z101. Сделать кнопку "СТОП" недоступной
A4 (Автомат управления остановом): в состоянии 0 (Ожидание инициализации) запущен с событием e0 (_инициализация_)
A4 (Автомат управления остановом): перешел из состояния 0 (Ожидание инициализации) в состояние 1 (Задвижение стержней)
A4 (Автомат управления остановом): завершил обработку события e0 (_инициализация_)
z340. Инициализация A4: вызов A4(e0)
A0 (Главный автомат управления реактором): завершил обработку события e101 (Нажатие кнопки "Стоп")
A0 (Главный автомат управления реактором): в состоянии 3 (Штатная остановка) запущен с событием e10 (Системный таймер)
A4 (Автомат управления остановом): в состоянии 1 (Задвижение стержней) запущен с событием e10 (Системный таймер)
x80 - Скорость теплоносителя максимальна - вернул 0
z210. Увеличить глубину погружения стержней (понизить число нейтронов)
A4 (Автомат управления остановом): завершил обработку события e10 (Системный таймер)
z341. Работа A4: вызов A4(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 3 (Штатная остановка) запущен с событием e10 (Системный таймер)
A4 (Автомат управления остановом): в состоянии 1 (Задвижение стержней) запущен с событием e10 (Системный таймер)
x80 - Скорость теплоносителя максимальна - вернул 0
z210. Увеличить глубину погружения стержней (понизить число нейтронов)
A4 (Автомат управления остановом): завершил обработку события e10 (Системный таймер)
z341. Работа A4: вызов A4(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
A0 (Главный автомат управления реактором): в состоянии 3 (Штатная остановка) запущен с событием e10 (Системный таймер)
A4 (Автомат управления остановом): в состоянии 1 (Задвижение стержней) запущен с событием e10 (Системный таймер)
x80 - Скорость теплоносителя максимальна - вернул 1
A4 (Автомат управления остановом): перешел из состояния 1 (Задвижение стержней) в состояние 2 (Охлаждение)
A4 (Автомат управления остановом): завершил обработку события e10 (Системный таймер)
z341. Работа A4: вызов A4(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 3 (Штатная остановка) запущен с событием e10 (Системный таймер)
A4 (Автомат управления остановом): в состоянии 2 (Охлаждение) запущен с событием e10 (Системный таймер)
x70 - Температура <= температуры останова - вернул 1
A4 (Автомат управления остановом): перешел из состояния 2 (Охлаждение) в состояние 3 (<Торможение> теплоносителя)
A4 (Автомат управления остановом): завершил обработку события e10 (Системный таймер)
z341. Работа A4: вызов A4(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)

A0 (Главный автомат управления реактором): в состоянии 3 (Штатная остановка) запущен с событием e10 (Системный таймер)
A4 (Автомат управления остановом): в состоянии 3 (<Торможение> теплоносителя) запущен с событием e10 (Системный таймер)
x70 - Температура <= температуры останова - вернул 1
z201. Уменьшить скорость теплоносителя (повысить температуру)
A4 (Автомат управления остановом): завершил обработку события e10 (Системный таймер)
z341. Работа A4: вызов A4(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...
A0 (Главный автомат управления реактором): в состоянии 3 (Штатная остановка) запущен с событием e10 (Системный таймер)
A4 (Автомат управления остановом): в состоянии 3 (<Торможение> теплоносителя) запущен с событием e10 (Системный таймер)
x70 - Температура <= температуры останова - вернул 1
z201. Уменьшить скорость теплоносителя (повысить температуру)
A4 (Автомат управления остановом): завершил обработку события e10 (Системный таймер)
z341. Работа A4: вызов A4(e10)
A0 (Главный автомат управления реактором): завершил обработку события e10 (Системный таймер)
...

П 1.2. Контрольная панель с протоколом

Ниже представлен вид главной панели управления, в которой отражается текущее состояние управляющей системы.

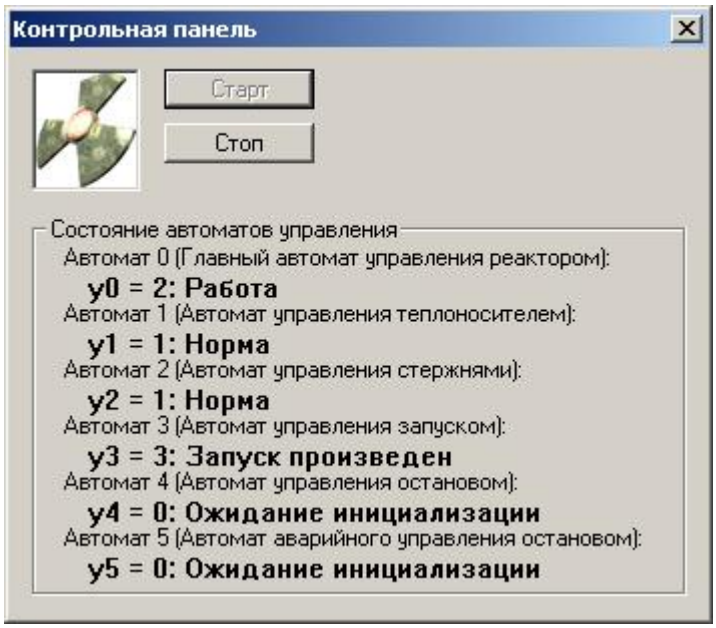

Рис 16. Контрольная панель

Приложение 2.

Ниже приведен текст программы моделирования ядерного реактора.

Текст программы

Файл Nuke.h

```
// Nuke.h : main header file for the NUKE application
//

#if !defined(AFX_NUKE_H__A099FD4C_FD9D_4024_B0EC_E30AEAF6885E__INCLUDED_)
#define AFX_NUKE_H__A099FD4C_FD9D_4024_B0EC_E30AEAF6885E__INCLUDED_

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

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CNukeApp:
// See Nuke.cpp for the implementation of this class
//

class CNukeApp : public CWinApp
{
public:
    CNukeApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNukeApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CNukeApp)
    afx_msg void OnAppAbout();
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_NUKE_H__A099FD4C_FD9D_4024_B0EC_E30AEAF6885E__INCLUDED_)
```

Файл Nuke.cpp

```
// Nuke.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Nuke.h"

#include "MainFrm.h"
#include "NukeDoc.h"
#include "NukeView.h"

#include "nuke_common.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CNukeApp

BEGIN_MESSAGE_MAP(CNukeApp, CWinApp)
   //{{AFX_MSG_MAP(CNukeApp)
   ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
      // NOTE - the ClassWizard will add and remove mapping macros here.
      //    DO NOT EDIT what you see in these blocks of generated code!
   //}}AFX_MSG_MAP
   // Standard file based document commands
   ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
   ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
   // Standard print setup command
   ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CNukeApp construction

CNukeApp::CNukeApp()
{
   // TODO: add construction code here,
   // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CNukeApp object

CNukeApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CNukeApp initialization

BOOL CNukeApp::InitInstance()
{
   AfxEnableControlContainer();

   // Standard initialization
   // If you are not using these features and wish to reduce the size
   //  of your final executable, you should remove from the following
   //  the specific initialization routines you do not need.

#ifdef _AFXDLL
   Enable3dControls();        // Call this when using MFC in a shared DLL
#else
   Enable3dControlsStatic();  // Call this when linking to MFC statically
#endif

   // Change the registry key under which our settings are stored.
   // TODO: You should modify this string to be something appropriate
   // such as the name of your company or organization.
   SetRegistryKey(_T("Local AppWizard-Generated Applications"));

   LoadStdProfileSettings();  // Load standard INI file options (including MRU)

   // Register the application's document templates.  Document templates
   //  serve as the connection between documents, frame windows and views.

   CSingleDocTemplate* pDocTemplate;
   pDocTemplate = new CSingleDocTemplate(
      IDR_MAINFRAME,
      RUNTIME_CLASS(CNukeDoc),
      RUNTIME_CLASS(CMainFrame),      // main SDI frame window
      RUNTIME_CLASS(CNukeView));
   AddDocTemplate(pDocTemplate);

   // Parse command line for standard shell commands, DDE, file open
   CCommandLineInfo cmdInfo;
   ParseCommandLine(cmdInfo);

   // Dispatch commands specified on the command line
   if (!ProcessShellCommand(cmdInfo))
      return FALSE;

   // The one and only window has been initialized, so show and update it.
   m_pMainWnd->ShowWindow(SW_SHOW);
```

```
   m_pMainWnd->UpdateWindow();

   return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
   CAboutDlg();

// Dialog Data
   //{{AFX_DATA(CAboutDlg)
   enum { IDD = IDD_ABOUTBOX };
   //}}AFX_DATA

   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CAboutDlg)
   protected:
   virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
   //}}AFX_VIRTUAL

// Implementation
protected:
   //{{AFX_MSG(CAboutDlg)
      // No message handlers
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() :
   CDialog(CAboutDlg::IDD)

{
   //{{AFX_DATA_INIT(CAboutDlg)
   //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
   CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAboutDlg)
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
   //{{AFX_MSG_MAP(CAboutDlg)
      // No message handlers
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CNukeApp::OnAppAbout()
{
   CAboutDlg aboutDlg;
   aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CNukeApp message handlers
```

Файл NukeDoc.h

```
// NukeDoc.h : interface of the CNukeDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_NUKEDOC_H__3F93971B_7E89_4CF3_8A3A_E05D3DF2EED4__INCLUDED_)
#define AFX_NUKEDOC_H__3F93971B_7E89_4CF3_8A3A_E05D3DF2EED4__INCLUDED_

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

```
class CNukeDoc : public CDocument
{
protected: // create from serialization only
    CNukeDoc();
    DECLARE_DYNCREATE(CNukeDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CNukeDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CNukeDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CNukeDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_NUKEDOC_H__3F93971B_7E89_4CF3_8A3A_E05D3DF2EED4__INCLUDED_)
```

Файл NukeDoc.cpp

```
// NukeDoc.cpp : implementation of the CNukeDoc class
//

#include "stdafx.h"
#include "Nuke.h"

#include "NukeDoc.h"

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

/////////////////////////////////////////////////////////////////////////////
// CNukeDoc

IMPLEMENT_DYNCREATE(CNukeDoc, CDocument)

BEGIN_MESSAGE_MAP(CNukeDoc, CDocument)
  //{{AFX_MSG_MAP(CNukeDoc)
    // NOTE - the ClassWizard will add and remove mapping macros here.
    //    DO NOT EDIT what you see in these blocks of generated code!
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CNukeDoc construction/destruction

CNukeDoc::CNukeDoc()
{
```

```
   // TODO: add one-time construction code here

}

CNukeDoc::~CNukeDoc()
{
}

BOOL CNukeDoc::OnNewDocument()
{
   if (!CDocument::OnNewDocument())
      return FALSE;

   // TODO: add reinitialization code here
   // (SDI documents will reuse this document)

   return TRUE;
}



/////////////////////////////////////////////////////////////////////////////
// CNukeDoc serialization

void CNukeDoc::Serialize(CArchive& ar)
{
   if (ar.IsStoring())
   {
      // TODO: add storing code here
   }
   else
   {
      // TODO: add loading code here
   }
}

/////////////////////////////////////////////////////////////////////////////
// CNukeDoc diagnostics

#ifdef _DEBUG
void CNukeDoc::AssertValid() const
{
   CDocument::AssertValid();
}

void CNukeDoc::Dump(CDumpContext& dc) const
{
   CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CNukeDoc commands
```

Файл StdAfx.h

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__F3D47A32_70EC_40DE_8861_61AD4B50326C__INCLUDED_)
#define AFX_STDAFX_H__F3D47A32_70EC_40DE_8861_61AD4B50326C__INCLUDED_

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

#define VC_EXTRALEAN        // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>       // MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>         // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

```
#include <vector>
#include <iostream.h>


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_STDAFX_H__F3D47A32_70EC_40DE_8861_61AD4B50326C__INCLUDED_)
```

Файл StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//  Nuke.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

Файл Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by Nuke.rc
//
#define IDD_ABOUTBOX                    100
#define IDR_MAINFRAME                   128
#define IDR_NUKETYPE                    129
#define IDB_NR_BASE_BMP                 130
#define IDB_NR_FUEL_PIVOT               132
#define IDD_CONTROLPANEL_MAIN           133
#define IDB_NR_BIGICON                  134
#define IDB_BACKGR                      136
#define IDC_BTN_START                   1001
#define IDC_BTN_STOP                    1002
#define IDC_CHK_STARTING                1003
#define IDC_CHK_STOPPING                1004
#define IDC_CHK_PIVOTS_OUT              1005
#define IDC_BTN_PAUSE                   1006
#define IDC_GRP_STATES                  1007
#define IDC_CHK_PIVOTS_INTO             1008
#define IDC_CHK_HC_SLOW                 1009
#define IDC_CHK_HC_ACCEL                1010
#define ID_CONTROLPANEL_MAIN            32771
#define ID_CONTROLPANEL_TESTTHEMODEL    32772
#define ID_TEST_PIVOTBRAKE              32773
#define ID_TEST_TEMPERATUREGROW         32774
#define ID_TEST_NEUTRONSCOUNTFALL       32775
#define ID_TEST_WORKINGAREATEMPERATUREGROW 32776
#define ID_CONTROLPANEL                 32777
#define ID_MODELLING_HEATCARRIERSTOP    32778

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                     1
#define _APS_NEXT_RESOURCE_VALUE        137
#define _APS_NEXT_COMMAND_VALUE         32779
#define _APS_NEXT_CONTROL_VALUE         1011
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Файл NukeView.h

```
// NukeView.h : interface of the CNukeView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_NUKEVIEW_H__2F220E36_26EA_4598_A587_B081B81F9C2C__INCLUDED_)
#define AFX_NUKEVIEW_H__2F220E36_26EA_4598_A587_B081B81F9C2C__INCLUDED_

#include "nukeGraphs.h" // Added by ClassView
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include <math.h>
```

```
#include <sys/timeb.h>

#include "nuke_common.h"
#include "nuke_data.h"
#include "ControlPanelMain.h"   // Added by ClassView
//#include "nuke_model.h"


// Total graph size(s)
#define GRAPHS_WIDTH    300
#define GRAPHS_HEIGHT   170
// Graph draw area size(s)
#define GRAPHS_DWIDTH   (GRAPHS_WIDTH-70)
#define GRAPHS_DHEIGHT  (GRAPHS_HEIGHT-60)
// Graph bound spacing
#define GRAPHS_BOUND    10
// Left and top of nuclear reactor area
#define NR_LEFT         (GRAPHS_WIDTH + GRAPHS_BOUND * 2)
#define NR_TOP          10
// Fuel Pivots definitions
#define FPIV_AMOUNT     5
#define FPIV_OFFSET     58
#define FPIV_DELTA      50
#define FPIV_MPIV_BOUND 7
// How many milliseconds it is between WM_TIMER events
#define TIMER_ELAPSE    100


class CNukeView : public CScrollView
{
protected: // create from serialization only
   CNukeView();
   DECLARE_DYNCREATE(CNukeView)

// Attributes
public:
   CNukeDoc* GetDocument();

// Operations
public:

// Overrides
   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CNukeView)
   public:
   virtual void OnDraw(CDC* pDC);  // overridden to draw this view
   virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
   protected:
   virtual void OnInitialUpdate(); // called first time after construct
   virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
   virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
   virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
   //}}AFX_VIRTUAL

// Implementation
public:
   void ShowControlPanelMain();
   CControlPanelMain m_dlgControlPanelMain;
   // Heat carrier
   double m_curStrokeState;
   PointVector m_heatCarriesTrsVec[3];
   // Graphics
   PointVector m_pv_h, m_pv_k, m_pv_n, m_pv_v, m_pv_Thc;
   GraphData m_gr_h, m_gr_k, m_gr_n, m_gr_v, m_gr_Thc;
   // Bitmaps
   CDC m_nrBaseDC, m_nrFuelPivotDC, m_scrBufferDC, m_backgrDC;
   CBitmap m_nrBaseBmp, m_nrFuelPivotBmp, m_scrBufferBmp, m_backgrBmp;
   //
   Nuke_data GetModelData(bool recalcData = false);
   void SetModelData(const Nuke_data& data);

   virtual ~CNukeView();
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif

protected:
//    Nuke_model model;
```

```
// Generated message map functions
protected:
   //{{AFX_MSG(CNukeView)
   afx_msg void OnTimer(UINT nIDEvent);
   afx_msg void OnDestroy();
   afx_msg void OnTestTemperaturegrow();
   afx_msg void OnTestPivotbrake();
   afx_msg void OnTestNeutronscountfall();
   afx_msg void OnTestWorkingareatemperaturegrow();
   afx_msg void OnControlpanel();
   afx_msg void OnModellingHeatcarrierstop();
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
private:
   void CheckMenuItem(int menuID, bool checkState);
};

#ifndef _DEBUG  // debug version in NukeView.cpp
inline CNukeDoc* CNukeView::GetDocument()
   { return (CNukeDoc*)m_pDocument; }
#endif

////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_NUKEVIEW_H__2F220E36_26EA_4598_A587_B081B81F9C2C__INCLUDED_)
```

Файл NukeView.cpp

```
// NukeView.cpp : implementation of the CNukeView class
//

#include "stdafx.h"
#include "Nuke.h"

#include "NukeDoc.h"
#include "NukeView.h"

#include "Nuke_model.h"
#include "Nuke_data.h"
#include "Nuke_control_system.h"

#include "Visio\\Automates\\common.h"

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

#define HEAT_CARRIER_TRS    3
#define HEAT_CARRIER_PTS    21

const double THC_MAX = WORKING_AREA_TEMPERATURE_1 + 1000;
const double TWA_MAX = WORKING_AREA_TEMPERATURE_1 + 1000;
const double H_MAX = 100;
const double V_MAX = 100;
const double N_MAX = NEUTRON_COUNT_VALUE_2;

// Hear carrier trajectories
//const int heatCarrierTop = 41, headCarrierLeft = 293;
const int heatCarrierTrs[HEAT_CARRIER_TRS][HEAT_CARRIER_PTS][3] = {
    {{449, 51, -1}, // 3rd not valid
    {336, 51, 50},
    {319, 55, 10},
    {307, 68, 10},
    {303, 84, 10},
    {307, 100, 10},
    {319, 113, 10},
    {336, 117, 10},
    {353, 121, 10},
    {365, 133, 10},
    {369, 150, 10},
    {365, 167, 10},
    {353, 179, 10},
    {336, 183, 10},
    {319, 187, 10},
```

```
    {307, 200, 10},
    {303, 216, 10},
    {307, 233, 10},
    {319, 245, 10},
    {336, 249, 10},
    {449, 249, 50}},

    {{449, 55, -1}, // 3rd not valid
    {336, 55, 50},
    {322, 59, 10},
    {311, 69, 10},
    {307, 84, 10},
    {311, 98, 10},
    {322, 109, 10},
    {336, 113, 10},
    {354, 118, 10},
    {368, 132, 10},
    {373, 150, 10},
    {368, 168, 10},
    {354, 182, 10},
    {336, 187, 10},
    {322, 191, 10},
    {311, 201, 10},
    {307, 216, 10},
    {311, 231, 10},
    {322, 241, 10},
    {336, 245, 10},
    {449, 245, 50}},

    {{449, 47, -1}, // 3rd not valid
    {336, 47, 50},
    {317, 52, 10},
    {304, 65, 10},
    {299, 84, 10},
    {304, 102, 10},
    {317, 116, 10},
    {336, 121, 10},
    {351, 125, 10},
    {361, 135, 10},
    {365, 150, 10},
    {361, 165, 10},
    {351, 175, 10},
    {336, 179, 10},
    {317, 184, 10},
    {304, 198, 10},
    {299, 216, 10},
    {304, 235, 10},
    {317, 248, 10},
    {336, 253, 10},
    {449, 253, 50}}
};

/////////////////////////////////////////////////////////////////////////////
// CNukeView

IMPLEMENT_DYNCREATE(CNukeView, CScrollView)

BEGIN_MESSAGE_MAP(CNukeView, CScrollView)
    //{{AFX_MSG_MAP(CNukeView)
    ON_WM_TIMER()
    ON_WM_DESTROY()
    ON_COMMAND(ID_TEST_TEMPERATUREGROW, OnTestTemperaturegrow)
    ON_COMMAND(ID_TEST_PIVOTBRAKE, OnTestPivotbrake)
    ON_COMMAND(ID_TEST_NEUTRONSCOUNTFALL, OnTestNeutronscountfall)
    ON_COMMAND(ID_TEST_WORKINGAREATEMPERATUREGROW, OnTestWorkingareatemperaturegrow)
    ON_COMMAND(ID_CONTROLPANEL, OnControlpanel)
    ON_COMMAND(ID_MODELLING_HEATCARRIERSTOP, OnModellingHeatcarrierstop)
    //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CNukeView construction/destruction

CNukeView::CNukeView()
{
    // TODO: add construction code here
```

```
    m_curStrokeState = 0.0;

    m_dlgControlPanelMain.Create(IDD_CONTROLPANEL_MAIN, this);
}

CNukeView::~CNukeView()
{
    m_dlgControlPanelMain.DestroyWindow();
}

BOOL CNukeView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return CScrollView::PreCreateWindow(cs);
}

////////////////////////////////////////////////////////////////////////////
// CNukeView drawing

void CNukeView::OnDraw(CDC* pDC)
{
    static bool bFirstDraw = true;
    if( bFirstDraw )
    {
        ShowControlPanelMain();
    }
    bFirstDraw = false;

    CNukeDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here

    // Copy buffer bitmap
    BITMAP bmBufInfo;
    m_scrBufferBmp.GetBitmap( &bmBufInfo );

    pDC->BitBlt( 0, 0, bmBufInfo.bmWidth, bmBufInfo.bmHeight,
                 &m_scrBufferDC, 0, 0, SRCCOPY );
}

void CNukeView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();

    CSize sizeTotal;
    // TODO: calculate the total size of this view
    sizeTotal.cx = 920; //780;
    sizeTotal.cy = 720; //550 480;
    SetScrollSizes(MM_TEXT, sizeTotal);

    // Move window
    RECT areaInfo;
    SystemParametersInfo(SPI_GETWORKAREA, 0, &areaInfo, 0);

    int bounding = 10; // pixels around main window
    areaInfo.top += bounding;
    areaInfo.left += bounding;
    areaInfo.bottom -= bounding;
    areaInfo.right -= bounding;
    AfxGetMainWnd()->MoveWindow( &areaInfo );

    // Initialization / prepare data
    srand( (unsigned)time( NULL ) );

    CClientDC dc(this);

    // Buffer bitmap
    m_scrBufferDC.CreateCompatibleDC( &dc );
    m_scrBufferBmp.CreateCompatibleBitmap( &dc, sizeTotal.cx, sizeTotal.cy );
    m_scrBufferDC.SelectObject( &m_scrBufferBmp );

    // Nuclear reactor base bitmap
    m_nrBaseBmp.LoadBitmap( IDB_NR_BASE_BMP );
    m_nrBaseDC.CreateCompatibleDC( &dc );
    m_nrBaseDC.SelectObject( &m_nrBaseBmp );

    // Nuclear fuel pivot bitmap
    m_nrFuelPivotBmp.LoadBitmap( IDB_NR_FUEL_PIVOT );
```

```
m_nrFuelPivotDC.CreateCompatibleDC( &dc );
m_nrFuelPivotDC.SelectObject( &m_nrFuelPivotBmp );

// Background
m_backgrBmp.LoadBitmap( IDB_BACKGR );
m_backgrDC.CreateCompatibleDC( &dc );
m_backgrDC.SelectObject( &m_backgrBmp );


// Draw nuclear fuel pivots on the reactor base bitmap
int i;
BITMAP bmInfo;
m_nrFuelPivotBmp.GetBitmap( &bmInfo );
for( i = 0; i < FPIV_AMOUNT; ++i )
{
    m_nrBaseDC.BitBlt( i * FPIV_DELTA + FPIV_OFFSET, 25,
                       bmInfo.bmWidth, bmInfo.bmHeight,
                       &m_nrFuelPivotDC, 0, 0, SRCCOPY );
}

// Graphics
m_gr_h.setCanvasSettings( GRAPHS_BOUND, GRAPHS_BOUND,
                          GRAPHS_DWIDTH, GRAPHS_DHEIGHT );
m_gr_h.setCaption( "Глубина погруж. стержней, h(t), 0..1" );
m_gr_h.setData( &m_pv_h );
m_gr_h.setParentDC( &m_scrBufferDC );
m_gr_h.setPlotSettings( 0, 0, GRAPHS_DHEIGHT );

m_gr_k.setCanvasSettings( GRAPHS_BOUND, GRAPHS_BOUND * 2 + GRAPHS_HEIGHT,
                          GRAPHS_DWIDTH, GRAPHS_DHEIGHT );
m_gr_k.setCaption( "К-т размножения нейтронов, k(t)" );
m_gr_k.setData( &m_pv_k );
m_gr_k.setParentDC( &m_scrBufferDC );
m_gr_k.setPlotSettings( 0, 0, GRAPHS_DHEIGHT/2 );

m_gr_v.setCanvasSettings( GRAPHS_BOUND, GRAPHS_BOUND * 3 + GRAPHS_HEIGHT * 2,
                          GRAPHS_DWIDTH, GRAPHS_DHEIGHT );
m_gr_v.setCaption( "Циркуляция теплоносителя, v(t), 0..1" );
m_gr_v.setData( &m_pv_v );
m_gr_v.setParentDC( &m_scrBufferDC );
m_gr_v.setPlotSettings( 0, 0, GRAPHS_DHEIGHT );

m_gr_n.setCanvasSettings( GRAPHS_BOUND * 2 + GRAPHS_WIDTH,
                          GRAPHS_BOUND * 3 + GRAPHS_HEIGHT * 2,
                          GRAPHS_DWIDTH, GRAPHS_DHEIGHT );
m_gr_n.setCaption( "Поток нейтронов, n(t), 0..1" );
m_gr_n.setData( &m_pv_n );
m_gr_n.setParentDC( &m_scrBufferDC );
m_gr_n.setPlotSettings( 0, 0, GRAPHS_DHEIGHT );

m_gr_Thc.setCanvasSettings( GRAPHS_BOUND * 3 + GRAPHS_WIDTH * 2,
                          GRAPHS_BOUND * 3 + GRAPHS_HEIGHT * 2,
                          GRAPHS_DWIDTH, GRAPHS_DHEIGHT );
m_gr_Thc.setCaption( "Темпер. теплоносителя, Thc(t), 0..1" );
m_gr_Thc.setData( &m_pv_Thc );
m_gr_Thc.setParentDC( &m_scrBufferDC );
m_gr_Thc.setPlotSettings( 0, 0, GRAPHS_DHEIGHT );

// Heat carrier trajectories split
int j, k;
for( i = 0; i < HEAT_CARRIER_TRS; ++i )
{
    m_heatCarriesTrsVec[i].clear();
    for( j = 1; j < HEAT_CARRIER_PTS; ++j )
    {
        long x1 = heatCarrierTrs[i][j-1][0], y1 = heatCarrierTrs[i][j-1][1],
             x2 = heatCarrierTrs[i][j][0], y2 = heatCarrierTrs[i][j][1],
             steps = heatCarrierTrs[i][j][2];
        for( k = 0; k < steps; ++k )
        {
            m_heatCarriesTrsVec[i].push_back( myPoint(
                 x1 + (x2-x1) * (double(k)/(steps-1)), // -headCarrierLeft
                 y1 + (y2-y1) * (double(k)/(steps-1)) ) ); //-heatCarrierTop
        }
    }
}

// Timer
SetTimer( 1, TIMER_ELAPSE, NULL );
```

```
    // First paint
    OnTimer( 1 );
}

/////////////////////////////////////////////////////////////////////////////
// CNukeView printing

BOOL CNukeView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}

void CNukeView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CNukeView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CNukeView diagnostics

#ifdef _DEBUG
void CNukeView::AssertValid() const
{
    CScrollView::AssertValid();
}

void CNukeView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}

CNukeDoc* CNukeView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CNukeDoc)));
    return (CNukeDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CNukeView message handlers

void CNukeView::OnTimer(UINT nIDEvent)
{
    // Call main automaton
    A0(10);

    // TODO: Add your message handler code here and/or call default
    // Clear buffer bitmap
    BITMAP bmBufInfo;
    m_scrBufferBmp.GetBitmap( &bmBufInfo );
    m_scrBufferDC.PatBlt( 0, 0, bmBufInfo.bmWidth, bmBufInfo.bmHeight, WHITENESS );

    // Draw nuclear reactor base
    BITMAP bmInfo;
    m_nrBaseBmp.GetBitmap( &bmInfo );
    m_scrBufferDC.BitBlt( NR_LEFT, NR_TOP + 50, bmInfo.bmWidth, bmInfo.bmHeight,
                          &m_nrBaseDC, 0, 0, SRCCOPY );
    //

    // Draw model data
    Nuke_data data = GetModelData(true);    //  кажд. 1/10 с.
    double curTime = data.time();

    // Update data for graphics
    m_pv_h.push_back( myPoint( curTime, data.h() / H_MAX ) );
    m_pv_k.push_back( myPoint( curTime, data.k() ) );
    m_pv_v.push_back( myPoint( curTime, data.v() / V_MAX ) );
    m_pv_n.push_back( myPoint( curTime, data.n() / N_MAX ) );
    m_pv_Thc.push_back( myPoint( curTime, data.Thc() / THC_MAX ) );

    // TODO: this is temporary (erase) !!
    const int bnd = 75;
    if( m_pv_h.size() > bnd ) m_pv_h.erase( m_pv_h.begin() );
    if( m_pv_k.size() > bnd ) m_pv_k.erase( m_pv_k.begin() );
```

```
    if( m_pv_v.size() > bnd ) m_pv_v.erase( m_pv_v.begin() );
    if( m_pv_n.size() > bnd ) m_pv_n.erase( m_pv_n.begin() );
    if( m_pv_Thc.size() > bnd ) m_pv_Thc.erase( m_pv_Thc.begin() );

    int i, j;
    m_nrFuelPivotBmp.GetBitmap( &bmInfo );

    // Fill tmFillRect structure
    RECT tmFillRect;
    tmFillRect.left = NR_LEFT     +25;
    tmFillRect.top = NR_TOP + 50  +37;
    tmFillRect.right = tmFillRect.left + 13;

    double twa_max = data.Twa();
    tmFillRect.bottom = tmFillRect.top + 205 * ( (twa_max < TWA_MAX) ? 1 - (data.Twa() / TWA_MAX)
: 0);

    if (twa_max > TWA_MAX)
    {
        RECT lampRect(tmFillRect);
        lampRect.top -= 20;
        lampRect.bottom = lampRect.top + 13;

        CBrush redBrush;
        redBrush.CreateSolidBrush( RGB( 255, 0, 0 ) );
        CPen redPen;
        redPen.CreatePen( PS_SOLID, 2, RGB( 196, 0, 0 ) );
        m_scrBufferDC.SelectObject( &redBrush );
        m_scrBufferDC.SelectObject( &redPen );
        m_scrBufferDC.Ellipse(&lampRect);
    }

    // Get temperature color
    COLORREF tmColor = m_scrBufferDC.GetPixel(
        (tmFillRect.right + tmFillRect.left) / 2,
        tmFillRect.bottom + 1
    );
    int greenGain = (tmColor & 0x0000ff00) >> 8;
    if( greenGain < 96 ) greenGain = 96;
    tmColor = (tmColor & 0x00ff00ff) | (greenGain << 8);

    // Draw temperature bound around fuel pivots
    CPen penTmFuel;
    penTmFuel.CreatePen( PS_SOLID, 3, tmColor );
    for( i = 0; i < FPIV_AMOUNT; ++i )
    {
        int x1 = i * FPIV_DELTA + FPIV_OFFSET, y1 = 25;
        m_nrBaseDC.SelectStockObject( HOLLOW_BRUSH );
        m_nrBaseDC.SelectObject( &penTmFuel );
        m_nrBaseDC.RoundRect(
            x1, y1,
            x1 + 1 + bmInfo.bmWidth, y1 + bmInfo.bmHeight,
            22, 22 );
    }

    CBrush brMPiv;
    brMPiv.CreateSolidBrush( RGB( 144, 144, 128 ) );
    //brMPiv.CreateSolidBrush(  );

    // Draw neutrons
    RECT rectNeutrons;
    rectNeutrons.top = NR_TOP + 50  +25;
    rectNeutrons.left = NR_LEFT + FPIV_OFFSET;
    rectNeutrons.bottom = rectNeutrons.top + bmInfo.bmHeight;
    rectNeutrons.right = rectNeutrons.left + (FPIV_AMOUNT-1) * FPIV_DELTA + bmInfo.bmWidth;

    double n_cnt = (data.n() < N_MAX) ? data.n() : N_MAX;
    for( i = 0; i < 10*log10(n_cnt+1); ++i )
    {
        int x = rectNeutrons.left + (rectNeutrons.right-rectNeutrons.left) *
double(rand())/RAND_MAX;
        int y = rectNeutrons.top + (rectNeutrons.bottom-rectNeutrons.top) *
double(rand())/RAND_MAX;
        m_scrBufferDC.SelectStockObject( BLACK_PEN );
        m_scrBufferDC.Rectangle( x-1, y-1, x+1, y+1 );
    }

    // Draw managing pivots
    for( i = 0; i < FPIV_AMOUNT-1; ++i )
    {
```

```
        RECT rect;
        rect.top = NR_TOP;
        rect.bottom = NR_TOP + 50 + long(data.h() * 2.5 + 25);
        rect.left = NR_LEFT + i * FPIV_DELTA + FPIV_OFFSET + bmInfo.bmWidth + FPIV_MPIV_BOUND;
        rect.right = rect.left + (FPIV_DELTA - bmInfo.bmWidth - 2 * FPIV_MPIV_BOUND);

        m_scrBufferDC.FillRect( &rect, &brMPiv );
    }

    m_curStrokeState += (data.v() * 3/100);
    if( m_curStrokeState > 4 )
        m_curStrokeState -= 9; // i.e. -4 or more

    // Draw heat carrier trajectories
    // Brush
    CBrush fillBr;
    double color_val = (data.Thc() < THC_MAX) ? data.Thc()/THC_MAX : 1;
    fillBr.CreateSolidBrush( RGB(254 * color_val, 0, 255 - 254 * color_val) );
    //fillBr.CreateHatchBrush( HS_HORIZONTAL, RGB(0,0,255) );
    m_scrBufferDC.SelectObject( &fillBr );
    m_scrBufferDC.ExtFloodFill( NR_LEFT +303, NR_TOP + 50 +84, RGB(255,255,255),
FLOODFILLSURFACE);
    // Pen
    CPen pen;
    pen.CreatePen( PS_SOLID, 1, RGB(255,255,255) );
    //pen.CreatePen( PS_SOLID, 1, RGB(0,0,255) ); //PS_DASH
    m_scrBufferDC.SelectObject( &pen );
    for( i = 0; i < HEAT_CARRIER_TRS; ++i )
    {
        int p1 = m_curStrokeState; if( p1 < 0 ) p1 = 0;
        int p2 = m_curStrokeState + 4;
        while( 1 )
        {
            if( p2 > m_heatCarriesTrsVec[i].size()-1 )
                p2 = m_heatCarriesTrsVec[i].size()-1;

            bool bFirst = true;
            for( j = p1; j < p2; ++j )
            {
                int x = m_heatCarriesTrsVec[i][j].x + NR_LEFT,
                    y = m_heatCarriesTrsVec[i][j].y + NR_TOP + 50;
                if( bFirst )
                    m_scrBufferDC.MoveTo(x, y);
                else
                    m_scrBufferDC.LineTo(x, y);
                bFirst = false;
            }
            if( p2 == m_heatCarriesTrsVec[i].size()-1 )
                break; // while
            p1 = p2 + 4;
            p2 = p1 + 4;
        }
    }

    // Hide some thermometer
    // tmFillRect contains the fill bounds

    CBrush brTmFill;
    brTmFill.CreateSolidBrush( RGB(255,255,255) );
    //brTmFill.CreateSolidBrush( RGB(0,0,0) );
    m_scrBufferDC.FillRect( &tmFillRect, &brTmFill );

    // Correct the time (x) axis
    // NOTE: TODO: move to function
    if( m_gr_h.get_ex() < curTime )
        m_gr_h.setPlotSettings( curTime - (m_gr_h.get_ex() - m_gr_h.get_sx()),
                                m_gr_h.get_sy(), m_gr_h.get_unitsize() );
    if( m_gr_k.get_ex() < curTime )
        m_gr_k.setPlotSettings( curTime - (m_gr_k.get_ex() - m_gr_k.get_sx()),
                                m_gr_k.get_sy(), m_gr_k.get_unitsize() );

    if( m_gr_v.get_ex() < curTime )
        m_gr_v.setPlotSettings( curTime - (m_gr_v.get_ex() - m_gr_v.get_sx()),
                                m_gr_v.get_sy(), m_gr_v.get_unitsize() );

    if( m_gr_n.get_ex() < curTime )
        m_gr_n.setPlotSettings( curTime - (m_gr_n.get_ex() - m_gr_n.get_sx()),
                                m_gr_n.get_sy(), m_gr_n.get_unitsize() );
```

```
    if( m_gr_Thc.get_ex() < curTime )
        m_gr_Thc.setPlotSettings( curTime - (m_gr_Thc.get_ex() - m_gr_Thc.get_sx()),
                                  m_gr_Thc.get_sy(), m_gr_Thc.get_unitsize() );


    // Draw graphs
    m_gr_h.drawMe();
    m_gr_k.drawMe();
    m_gr_v.drawMe(); m_gr_n.drawMe(); m_gr_Thc.drawMe();

    // Update local time
    // TODO: this is temporary !!
//  curTime += TIMER_ELAPSE / 1000.0;

    Invalidate( FALSE );
    //RedrawWindow( NULL, NULL, RDW_INVALIDATE | RDW_UPDATENOW );

    m_dlgControlPanelMain.RefreshAutomatonStates();

    //
    CScrollView::OnTimer(nIDEvent);
}

void CNukeView::OnDestroy()
{
    CScrollView::OnDestroy();

    // TODO: Add your message handler code here
    KillTimer( 1 );
}

void CNukeView::SetModelData(const Nuke_data& data)
{
    Nuke_model::ptr().set_data(data);
}

Nuke_data CNukeView::GetModelData(bool recalcData)
{
    // Get data, then execute
    Nuke_data res = Nuke_model::ptr().get_data();
    if( recalcData ) Nuke_model::ptr().execute_for(CALULATE_TICKS);
    return res;
}

void CNukeView::ShowControlPanelMain()
{
    m_dlgControlPanelMain.ShowWindow(SW_SHOW);
    m_dlgControlPanelMain.SetActiveWindow();
    m_dlgControlPanelMain.BringWindowToTop();
}

void CNukeView::OnTestTemperaturegrow()
{
    // TODO: Add your command handler code here
    static bool checked = true;
    CheckMenuItem( ID_TEST_TEMPERATUREGROW, checked );
    Useriface_data data = Useriface_data(GetModelData());
    data.ud_set_dThc(checked ? 250 : 0);
    SetModelData(data);
    checked = !checked;
}

void CNukeView::OnTestPivotbrake()
{
    // TODO: Add your command handler code here
    static bool checked = true;
    CheckMenuItem( ID_TEST_PIVOTBRAKE, checked );
    Useriface_data data = Useriface_data(GetModelData());
    data.ud_set_dh(checked ? -2 : 0);
    SetModelData(data);
    checked = !checked;
}

void CNukeView::OnTestNeutronscountfall()
{
    // TODO: Add your command handler code here
    static bool checked = true;
    CheckMenuItem( ID_TEST_NEUTRONSCOUNTFALL, checked );
    Useriface_data data = Useriface_data(GetModelData());
    //data.ud_set_dn(checked ? N_MAX/100: 0);
```

```
//  if (checked)
//      data.set_n(data.n()+N_MAX/100);

    SetModelData(data);
    checked = !checked;
}

void CNukeView::OnTestWorkingareatemperaturegrow()
{
    // TODO: Add your command handler code here
    static bool checked = true;
    //CheckMenuItem( ID_TEST_WORKINGAREATEMPERATUREGROW, checked );
    Useriface_data data = Useriface_data(GetModelData());
    //data.ud_set_dTwa(checked ? 900 : 0);
    data.ud_set_Twa(data.Twa()+1000);
    SetModelData(data);
    checked = !checked;
}


void CNukeView::CheckMenuItem(int menuID, bool checkState)
{
    AfxGetMainWnd()->GetMenu()->CheckMenuItem(
        menuID, checkState ? MF_BYCOMMAND | MF_CHECKED : MF_BYCOMMAND | MF_UNCHECKED );
}


void CNukeView::OnControlpanel()
{
    // TODO: Add your command handler code here
    ShowControlPanelMain();
}

void CNukeView::OnModellingHeatcarrierstop()
{
    // TODO: Add your command handler code here
    //static bool checked = true;
    //CheckMenuItem( ID_MODELLING_HEATCARRIERSTOP, checked );
    Useriface_data data = Useriface_data(GetModelData());
    //data.ud_set_dv(checked ? -99 : 0);
    //if (checked)
    data.ud_set_v(1);
    SetModelData(data);
//  checked = !checked;
}
```

Файл NukeGraphs.h

```
#ifndef _NUKEGRAPHS_H_
#define _NUKEGRAPHS_H_

#include "nuke_common.h"
#include "graphBase.h"

class GraphData: public GraphBase
{
protected:
    PointVector* data;

public:
    void setData(PointVector* data);

protected:
    virtual void drawGraphic();
};

#endif    SetModelData(data);
//  checked = !checked;
}
```

Файл NukeGraphs.cpp

```
#include "StdAfx.h"
#include "nukeGraphs.h"

void GraphData::setData(PointVector* data)
{
    this->data = data;
}
```

```
void GraphData::drawGraphic()
{
    bool bFirst = true;
    double step = 1.0 / get_unitsize();

    CPen* oldPen, drawPen(PS_SOLID, 2, RGB(0, 0, 64));
    oldPen = (CPen*)parentDC->SelectObject( &drawPen );

    for( PointVector::const_iterator i = data->begin(); i != data->end(); ++i )
    {
        if( bFirst )
            parentDC->MoveTo( scr_x((*i).x), scr_y((*i).y) );
        else
            parentDC->LineTo( scr_x((*i).x), scr_y((*i).y) );
        bFirst = false;
    }

    parentDC->SelectObject( oldPen );
}
```

Файл graphBase.h

```
#ifndef _GRAPHBASE_H_
#define _GRAPHBASE_H_

#include <math.h>

////////////////////////////////////////////

class GraphBase
{
private:
   long sgn(long v);

protected:
   int screen_start_x, screen_start_y, gr_x_size, gr_y_size, unit_size;
   double sx, sy;
   CDC* parentDC;
   CFont font, capFont;
   char* caption;

   enum{ indent = 12, fontHeight = 16 };

public:
   GraphBase();
   virtual ~GraphBase();

   void setCanvasSettings(int screen_start_x, int screen_start_y, int gr_x_size, int gr_y_size);
   void setParentDC(CDC* parentDC);
   void setPlotSettings(double sx, double sy, int unit_size);
   void setCaption(char* caption);

   // get screen coordinates
   int scr_x(double x);
   int scr_y(double y);

   // get local coordinates (from screen)
   double loc_x(int x);
   double loc_y(int y);

   // get start and end (local) bounds
   double get_sx(); // start x of function
   double get_sy(); // start y of function
   double get_ex(); // end x of function on the graphic
   double get_ey(); // end y of function on the graphic

   // get length of one unit in pixels
   int get_unitsize();

   // get dimensions of the graph window
   int get_gr_x_size();
   int get_gr_y_size();

   // get left up graph window corner coordinates
   int get_graph_x1();
   int get_graph_y1();

   // invalidates graph
```

```
    void drawMe();

protected:
    virtual void drawGraphic() = 0;
};

#endif
```

Файл graphBase.cpp

```
#include "stdafx.h"
#include "graphBase.h"

#define STOP_BOUND  10000

GraphBase::GraphBase()
{
    font.CreateFont(fontHeight, 8, 0, 0, FW_MEDIUM, 0, 0, 0, ANSI_CHARSET,
                    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
                    PROOF_QUALITY, FIXED_PITCH, "Courier New");
    capFont.CreateFont(fontHeight+2, 8, 0, 0, FW_BOLD, 0, 0, 0, ANSI_CHARSET,
                       OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
                       PROOF_QUALITY, VARIABLE_PITCH, "Arial Cyr");
    parentDC = 0;
    caption = 0;
}

GraphBase::~GraphBase()
{
    if( caption )
    {
        delete[] caption;
        caption = 0;
    }
}

void GraphBase::setCanvasSettings(int screen_start_x, int screen_start_y, int gr_x_size, int
gr_y_size)
{
    this->screen_start_x = screen_start_x;
    this->screen_start_y = screen_start_y;
    this->gr_x_size = gr_x_size;
    this->gr_y_size = gr_y_size;
}

void GraphBase::setPlotSettings(double sx, double sy, int unit_size)
{
    this->sx = sx;
    this->sy = sy;
    this->unit_size = unit_size;
}

void GraphBase::setCaption(char* caption)
{
    if( this->caption )
    {
        delete[] this->caption;
        this->caption = 0;
    }
    this->caption = strdup(caption);
}

void GraphBase::setParentDC(CDC* parentDC)
{
    this->parentDC = parentDC;
}

void GraphBase::drawMe()
{
    CRgn total_rgn, graphic_rgn;
    CFont* oldFont;
    //CBrush* oldBrush;
    char buffer[10];
    TEXTMETRIC tm;

    CBrush bgBrush1, bgBrush2;
    bgBrush1.CreateSolidBrush(RGB(0, 255, 255));
    bgBrush2.CreateSolidBrush(RGB(196, 196, 196));
```

```
    oldFont = (CFont*)parentDC->SelectObject(&font);
    parentDC->GetTextMetrics(&tm);

    CPen* oldPen, drawPen(PS_SOLID, 1, RGB(0, 0, 0));
    oldPen = (CPen*)parentDC->SelectObject( &drawPen );

    int total_x1 = screen_start_x,
        total_y1 = screen_start_y,
        total_x2 = get_graph_x1() + gr_x_size + indent,
        total_y2 = get_graph_y1() + gr_y_size + indent + tm.tmHeight +
                   (caption ? (int)(indent*0.5 + tm.tmHeight) : 0);

    total_rgn.CreateRectRgn( total_x1, total_y1, total_x2, total_y2 );
    graphic_rgn.CreateRectRgn( get_graph_x1(), get_graph_y1(),
                               get_graph_x1() + gr_x_size, get_graph_y1() + gr_y_size);

    parentDC->SelectClipRgn(&total_rgn);
    parentDC->SetBkMode(TRANSPARENT);
    /////////////// begin drawing /////////////

    // bounding box and background 1
    parentDC->Rectangle( total_x1, total_y1, total_x2, total_y2 );
    parentDC->FillRect( CRect(total_x1 + 1, total_y1 + 1, total_x2 - 1, total_y2 - 1), &bgBrush1
);

    // bounding box and background 2
    parentDC->Rectangle( get_graph_x1(), get_graph_y1(),
                         get_graph_x1() + gr_x_size + 1, get_graph_y1() + gr_y_size + 1 );
    parentDC->FillRect( CRect(get_graph_x1() + 1,
                              get_graph_y1() + 1,
                              get_graph_x1() + gr_x_size,
                              get_graph_y1() + gr_y_size),
                        &bgBrush2 );

    int i;
    double ex = get_ex(), ey = get_ey();

    // vertical lines of grid
    for( i = (int)ceil(sx); i <= (int)ex; ++i )
    {
        parentDC->MoveTo( scr_x(i), scr_y(sy) );
        parentDC->LineTo( scr_x(i), scr_y(ey) );
        if( i <= -10 || i >= 100 )
            sprintf(buffer, "%4d", i);
        else
            sprintf(buffer, "%4.1f", (double)i);
        //parentDC->
        parentDC->TextOut( scr_x(i) - parentDC->GetTextExtent(buffer).cx/2,
                           get_graph_y1() + gr_y_size + indent/2, buffer );
    }

    // horizontal lines of grid
    for( i = (int)ceil(sy); i <= (int)ey; ++i )
    {
        parentDC->MoveTo( scr_x(sx), scr_y(i) );
        parentDC->LineTo( scr_x(ex), scr_y(i) );
        if( i <= -10 || i >= 100 )
            sprintf(buffer, "%4d", i);
        else
            sprintf(buffer, "%4.1f", (double)i);
        parentDC->TextOut( screen_start_x + indent/2, scr_y(i) - tm.tmHeight/2, buffer );
    }

    if( caption )
    {
        oldFont = (CFont*)parentDC->SelectObject(&capFont);
        COLORREF oldColor = parentDC->SetTextColor(RGB(0,0,255));
        parentDC->TextOut( (total_x2 + total_x1)/2 - parentDC->GetTextExtent(caption).cx/2,
                           get_graph_y1() + gr_y_size + indent*0.5 + tm.tmHeight, caption);
        parentDC->SetTextColor( oldColor );
        parentDC->SelectObject(oldFont);
    }

    parentDC->SelectObject( oldPen );

    // draw graphic
    parentDC->SelectClipRgn(&graphic_rgn);
    drawGraphic();
    parentDC->SelectClipRgn(&total_rgn);
```

```
    ////////////// end drawing //////////////
    parentDC->SelectObject(oldFont);
    parentDC->SelectClipRgn(0);
}

double GraphBase::get_sx()
{
    return sx;
}

double GraphBase::get_sy()
{
    return sy;
}

double GraphBase::get_ex()
{
    return sx + ((double)gr_x_size) / (unit_size);
}

double GraphBase::get_ey()
{
    return sy + ((double)gr_y_size) / (unit_size);
}

int GraphBase::get_unitsize()
{
    return unit_size;
}

int GraphBase::get_gr_x_size()
{
    return gr_x_size;
}

int GraphBase::get_gr_y_size()
{
    return gr_y_size;
}

int GraphBase::get_graph_x1()
{
    TEXTMETRIC tm;
    CFont* oldFont = parentDC->SelectObject(&font);
    parentDC->GetTextMetrics(&tm);
    parentDC->SelectObject(oldFont);
    return screen_start_x + indent + tm.tmAveCharWidth * 4;
}

int GraphBase::get_graph_y1()
{
    return screen_start_y + indent;
}

int GraphBase::scr_x(double x)
{
    int r = (int)(get_graph_x1() + (x - sx) * unit_size + 0.5);
    return (labs(r) <= STOP_BOUND ) ? r : STOP_BOUND * sgn(r);
}

int GraphBase::scr_y(double y)
{
    int r = (int)(get_graph_y1() + gr_y_size - (y - sy) * unit_size + 0.5);
    return (labs(r) <= STOP_BOUND ) ? r : STOP_BOUND * sgn(r);
}

double GraphBase::loc_x(int x)
{
    return (x-get_graph_x1())/((double)unit_size) + sx;
}

double GraphBase::loc_y(int y)
{
    return (gr_y_size-y+get_graph_y1())/((double)unit_size) + sy;
}

long GraphBase::sgn(long v)
{
    return v > 0 ? 1 : v < 0 ? -1 : 0;
}
```

```
/////////////////////////////////////////////
```

Файл MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__C61CC757_01FD_4796_BA6D_A5ECFE18BF9E__INCLUDED_)
#define AFX_MAINFRM_H__C61CC757_01FD_4796_BA6D_A5ECFE18BF9E__INCLUDED_

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

class CMainFrame : public CFrameWnd
{

protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
    CStatusBar  m_wndStatusBar;
    CToolBar    m_wndToolBar;

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_MAINFRM_H__C61CC757_01FD_4796_BA6D_A5ECFE18BF9E__INCLUDED_)
```

Файл MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "Nuke.h"

#include "MainFrm.h"

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

/////////////////////////////////////////////////////////////////////////////
```

```
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
    // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
        | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }

    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;      // fail to create
    }

    // TODO: Delete these three lines if you don't want the toolbar to
    //  be dockable
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
```

```
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

Файл ControlPanelMain.h

```
#if !defined(AFX_CONTROLPANELMAIN_H__A120C3C2_F1E0_4270_9CD6_843344995B63__INCLUDED_)
#define AFX_CONTROLPANELMAIN_H__A120C3C2_F1E0_4270_9CD6_843344995B63__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ControlPanelMain.h : header file
//

#define     FRAME_TOP_OFFSET    14
#define     FRAME_LEFT_OFFSET   16
#define     FRAME_Y_DELTA       14
#define     FRAME_LEFT_OFFSET2  12
#define     FRAME_FILL_OFFSET   6
#define     FRAME_FILL_TOP_ADD  10

/////////////////////////////////////////////////////////////////////////////
// CControlPanelMain dialog

class CControlPanelMain : public CDialog
{
// Construction
public:
    void RefreshAutomatonStates();
    CControlPanelMain(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CControlPanelMain)
    enum { IDD = IDD_CONTROLPANEL_MAIN };
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CControlPanelMain)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CControlPanelMain)
    afx_msg void OnBtnStart();
    afx_msg void OnBtnStop();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_CONTROLPANELMAIN_H__A120C3C2_F1E0_4270_9CD6_843344995B63__INCLUDED_)
```

Файл ControlPanelMain.cpp

```
// ControlPanelMain.cpp : implementation file
//
#include "stdafx.h"

#include "Nuke.h"
#include "ControlPanelMain.h"
#include "Visio\\Automates\\common.h"
#include "Visio\\Automates\\log.h"

#include "Visio\\Automates\\common.h"
```

```
#include "Visio\\Automates\\log.h"

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

/////////////////////////////////////////////////////////////////////////////
// CControlPanelMain dialog


CControlPanelMain::CControlPanelMain(CWnd* pParent /*=NULL*/)
    : CDialog(CControlPanelMain::IDD, pParent)
{
    //{{AFX_DATA_INIT(CControlPanelMain)
    //}}AFX_DATA_INIT
}


void CControlPanelMain::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CControlPanelMain)
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CControlPanelMain, CDialog)
    //{{AFX_MSG_MAP(CControlPanelMain)
    ON_BN_CLICKED(IDC_BTN_START, OnBtnStart)
    ON_BN_CLICKED(IDC_BTN_STOP, OnBtnStop)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CControlPanelMain message handlers

void CControlPanelMain::OnBtnStart()
{
    A0(100);
}

void CControlPanelMain::OnBtnStop()
{
    A0(101);
}

void CControlPanelMain::RefreshAutomatonStates()
{
    CWnd* pFrameWnd = GetDlgItem( IDC_GRP_STATES );
    RECT rectFramePos;
    pFrameWnd->GetWindowRect( &rectFramePos );
    this->ScreenToClient( &rectFramePos );
    int left = rectFramePos.left, top = rectFramePos.top;

    CClientDC dc(this);
    dc.SetBkMode( TRANSPARENT );
    //dc.SetBkMode( OPAQUE );

    CBrush br;
    br.CreateSolidBrush( RGB(212, 208, 200) );
    rectFramePos.left += FRAME_FILL_OFFSET;
    rectFramePos.top  += FRAME_FILL_OFFSET + FRAME_FILL_TOP_ADD;
    rectFramePos.right  -= FRAME_FILL_OFFSET;
    rectFramePos.bottom -= FRAME_FILL_OFFSET;
    dc.FillRect( &rectFramePos, &br );

    CRgn total_rgn;
    total_rgn.CreateRectRgn( rectFramePos.left, rectFramePos.top, rectFramePos.right,
rectFramePos.bottom );
    dc.SelectClipRgn(&total_rgn);

    unsigned char m_states[] = { cm.y0, cm.y1, cm.y2, cm.y3, cm.y4, cm.y5 };
    str2_t* m_stateDescs[] = { a0_str2, a1_str2, a2_str2, a3_str2, a4_str2, a5_str2 };

    int i;
    char buf[256];
    CFont *formFont = this->GetFont(), *boldFont = dc.GetCurrentFont();
    //dc.SelectObject( this->GetFont() );
```

```
    for( i = 0; i <= 5; ++i )
    {
        int state = m_states[i];
        const char* automatonName = (A_str3 + i)->n_name;
        sprintf( buf, "└тЄюьрЄ %d (%s):", i, automatonName );
        dc.SelectObject( formFont );
        dc.TextOut(
            left + FRAME_LEFT_OFFSET,
            top + FRAME_TOP_OFFSET + i * 2 * FRAME_Y_DELTA,
            buf);

        const char* stateName = (m_stateDescs[i] + state)->n_name;
        sprintf( buf, "y%d = %d: %s", i, state, stateName );
        dc.SelectObject( boldFont );
        dc.TextOut(
            left + FRAME_LEFT_OFFSET + FRAME_LEFT_OFFSET2,
            top + FRAME_TOP_OFFSET + (i + 0.5) * 2 * FRAME_Y_DELTA,
            buf);
    }
    dc.SelectClipRgn(0);
}
```

Файл NukeCommon.h

```
// common.h

//

#ifndef NUKE_COMMON_H_
#define NUKE_COMMON_H_

typedef double timetype;
//typedef double valuetype;

#define PI 3.1415926536
#define PI2 (PI/2.0)

struct myPoint
{
   double x;
   double y;
   myPoint(double x, double y) { this->x = x; this->y = y; }
};
typedef std::vector<myPoint> PointVector;

const double EPS = 1e-5;


#endif //NUKE_COMMON_H_
```

Файл NukeControlSystem.h

```
// NUKE_CONTROL_SYSTEM.h

//

#ifndef NUKE_CONTROL_SYSTEM_H_
#define NUKE_CONTROL_SYSTEM_H_

//#include "common.h"
#include "nuke_data.h"

// параметры для управления
//const double DEF_NEUTRON_COUNT_VALUE = 1e27;
const double NEUTRON_COUNT_VALUE_M2 = 1e10;     // level -2: data.n() < this
const double NEUTRON_COUNT_VALUE_M1 = 1e19;     // level -1: data.n() < this
//const double NEUTRON_COUNT_VALUE_M0 = 1e27;       // level 0: data.n() < this
const double NEUTRON_COUNT_VALUE_1 = 2*1e19;        // level 1: data.n() < this
const double NEUTRON_COUNT_VALUE_2 = 5e19;      // level 2: data.n() < this

//const double DEF_WORKING_AREA_TEMPERATURE = 1000;
const double WORKING_AREA_TEMPERATURE_M2 = 400;     //
const double WORKING_AREA_TEMPERATURE_M1 = 800;     //
//const double WORKING_AREA_TEMPERATURE_M0 = 1000;  //
const double WORKING_AREA_TEMPERATURE_1 = 900;      //
const double WORKING_AREA_TEMPERATURE_2 = 3500;     //
const double NO_HEAT_TEMPERATURE = 350;
```

```
const timetype UNSTABLE_T_TIME = 3;//1;
const timetype UNSTABLE_N_TIME = 3;//1;
const timetype SIDE_SYS_INIT_TIME = 5;//5;

const double HEAT_CARRIER_INITIAL_VELOCITY = 50;
const double HEAT_CARRIER_BOUND_VELOCITY = 50;


#endif //NUKE_CONTROL_SYSTEM_H_
```

Файл NukeData.h

```
// NUKE_DATA.h

//

#ifndef NUKE_DATA_H_
#define NUKE_DATA_H_

#include "nuke_common.h"

const double SECONDS_IN_TICK = 1e-6;

// шаг извлечения/погружения стержней [проценты]
const double PIVOT_H_STEP = 0.2;

// начальная глубина погружения стержней [проценты]
const double INITIAL_PIVOT_H = 100;

// начальная скорость циркуляции теплоносителя [проценты]
const double INITIAL_HEAT_CARRIER_V = 1;

const double HEAT_CARRIER_V_STEP = 5.0;

const double CALULATE_TICKS = 0.1;      // на столько увеличивается время при расчете

//const double SMALLEST_TIME_TICK = 1;

class Nuke_model;

//-[ структура данных для обмена между модулями
struct Nuke_data
{
    friend class Nuke_model;   // модели должны быть доступны любые изменения параметров
private:
    timetype _time; // текущее время реактора (модели)
    // Сделаны графики
    double _h;      // глубина погружения стержней, в процентах (0-100)
    double _k;      // коэффициент размножения (примерно равен 1)
    double _v;      // скорость обращения теплоносителя, в процентах (0-100)
    double _n;      // число вылетающих нейтронов, в процентах (0-100)
    double _Thc;    // температура теплоносителя, в процентах (0-100)
    // Не сделаны графики
    double _Twa;    // температура рабочей зоны
    double _N;      // тепловая мощность
    double _P;      // полезная мощность (электрическая)
    // смещения для основных параметров
    double _dtime;  // смещение по времени (вперед/назад)
    double _dh;     // смещение глубины погружения (напр. стержень сломался)
    double _dk;     // смещение к-та размножения (напр. дырка в реакторе)
    double _dv;     // смещение скорости теплонос (затор в трубах)
    double _dn;     // изменение числа нейтронов (доп. источник)
    double _dTwa;   // изменение темп. акт. зоны (нарушен теплооток)
    double _dThc;   // изменение темп. теплоносителя (нарушен теплооток)
public:
    // constructor
    inline Nuke_data();
    // getters
    inline timetype time() const;
    inline double h() const;
    inline double k() const;
    inline double v() const;
    inline double n() const;
    inline double Thc() const;
    inline double Twa() const;
    inline double N()   const;
    inline double P()   const;
    // setters:     будут реализованы лишь примитивные сеттеры, не треубющие сложных
    //              расчетов. Все нетривиальные, а также зависимые от выбранной модели
```

```
    //              расчеты будут производиться в Nuke_model
protected:
    // main parameters
    inline void inc_time(timetype dt);   // инкрементировать время на dt

    inline void inc_h();                 // инкрементировать h - "погрузить стержни" на величину
    inline void dec_h();                 // декрементировать h - "выдвинуть стержни" на величину

    inline void set_v(double v);         // установить скорость циркуляции теплоносителя
    inline void inc_v();                 // инкрементировать v
    inline void dec_v();                 // декрементировать v

    inline void set_Twa(double Twa);
    // additional parameters

    inline void set_dtime(double dtime);     // установить смещение параметра
    inline void set_dh(double dh);   // установить смещение параметра
    inline void set_dk(double dk);   // установить смещение параметра
    inline void set_dv(double dv);   // установить смещение параметра
    inline void set_dn(double dn);   // установить смещение параметра
    inline void set_dTwa(double dTwa);   // установить смещение параметра
    inline void set_dThc(double dThc);   // установить смещение параметра

//    inline void set_P(double P);         // задать уровень полезной мощности

//  inline void set_k(double);
//  inline void set_n(double);
//  inline void set_Thc(double);
//  inline void set_Twa(double);
//  inline void set_N(double);
};

struct Csystem_data;
struct Useriface_data;

//-[ структура только для чтения - возврат
//struct Readonly_data : public Nuke_data
//{
//};

//-[ структура данных для блока управления ЯР
// доступны изменения:
//   инкрементировать h - "погрузить стержни" на величину
//   декрементировать h - "выдвинуть стержни" на величину
//   установить скорость циркуляции теплоносителя
struct Csystem_data : public Nuke_data
{
public:
    Csystem_data(const Nuke_data& data) :   Nuke_data(data) {}

    inline void cd_inc_h()                      { inc_h(); }
    inline void cd_dec_h()                      { dec_h(); }
    inline void cd_set_v(double v)              { set_v(v); }
    inline void cd_inc_v()                      { inc_v(); }
    inline void cd_dec_v()                      { dec_v(); }
};

//-[ структура данных для интерфейса пользователя
// доступны изменения:
//   инкрементировать h - "погрузить стержни" на величину
//   декрементировать h - "выдвинуть стержни" на величину
//   установить скорость циркуляции теплоносителя
struct Useriface_data : public Nuke_data
{
public:
    Useriface_data(const Nuke_data& data) :   Nuke_data(data) {}

    inline void ud_inc_h()                      { inc_h(); }
    inline void ud_dec_h()                      { dec_h(); }
    inline void ud_set_v(double v)              { set_v(v); }

    inline void ud_set_dtime(double dtime)      { set_dtime(dtime); }
    inline void ud_set_dh(double dh)            { set_dh(dh); }
    inline void ud_set_dk(double dk)            { set_dk(dk); }
    inline void ud_set_dv(double dv)            { set_dv(dv); }
    inline void ud_set_dn(double dn)            { set_dn(dn); }
    inline void ud_set_dTwa(double dTwa)        { set_dTwa(dTwa); }
    inline void ud_set_dThc(double dThc)        { set_dThc(dThc); }
    inline void ud_set_Twa(double Twa)          { set_Twa(Twa); }
```

```
    //    inline void ud_set_P(double P)              { set_P(P); }
};

// ------------------ INLINES --------------------

// ------------------------ Nuke_data ----------------

inline Nuke_data::Nuke_data() :
    _time(0),       // начальный момент времени - 0
    _h(INITIAL_PIVOT_H),        // стержни погружены на 100%
    _v(INITIAL_HEAT_CARRIER_V), // теплоноситель циркулирует со скоростью 50% максимальной
    _n(0),          // поток нейтронов          (?)
    _Thc(300),        // температура носителя     (?)
    _Twa(300),      // температура рабочей зоны (?)
    _N(0),          // тепловая мощность        (?)
    _P(0),          // полезная мощность        (?)
    _dtime(0),  // смещение по времени (вперед/назад)
    _dh(0),     // смещение глубины погружения (напр. стержень сломался)
    _dk(0),     // смещение к-та размножения (напр. дырка в реакторе)
    _dv(0),     // смещение скорости теплонос (затор в трубах)
    _dn(0),     // изменение числа нейтронов (доп. источник)
    _dTwa(0),   // изменение темп. акт. зоны (нарушен теплооток)
    _dThc(0)    // изменение темп. теплоносителя (нарушен теплооток)
{}

// getters
inline timetype Nuke_data::time() const { return _time + _dtime; }
inline double Nuke_data::h() const      { return _h + _dh; }
inline double Nuke_data::k() const      { return _k + _dk; }
inline double Nuke_data::v() const      { return _v + _dv; }
inline double Nuke_data::n() const      { return _n + _dn; }
inline double Nuke_data::Thc() const    { return _Thc + _dThc; }
inline double Nuke_data::Twa() const    { return _Twa + _dTwa; }
inline double Nuke_data::N() const      { return _N; }
inline double Nuke_data::P() const      { return _P; }

//some setters
inline void Nuke_data::inc_time(timetype dt)   { _time += dt; }

inline void Nuke_data::inc_h()                 { _h = ( (_h < 100) ? _h + PIVOT_H_STEP : 100); }
inline void Nuke_data::dec_h()                 { _h = ( (_h > 0) ? _h - PIVOT_H_STEP : 0); }

inline void Nuke_data::set_v(double v)         { _v = v; }
inline void Nuke_data::inc_v()                 { _v = ( (_v < 100) ? _v + HEAT_CARRIER_V_STEP :
100); }
inline void Nuke_data::dec_v()                 { _v = ( (_v > 1) ? _v - HEAT_CARRIER_V_STEP : 1);
}

// additional setters
inline void Nuke_data::set_dtime(double dtime)  { _dtime = dtime;}
inline void Nuke_data::set_dh(double dh)        { _dh = dh; }
inline void Nuke_data::set_dk(double dk)        { _dk = dk; }
inline void Nuke_data::set_dv(double dv)        { _dv = dv; }
inline void Nuke_data::set_dn(double dn)        { _dn = dn; }
inline void Nuke_data::set_dTwa(double dTwa)    { _dTwa = dTwa; }
inline void Nuke_data::set_dThc(double dThc)    { _dThc = dThc; }

inline void Nuke_data::set_Twa(double Twa)      { _Twa = Twa; }

//inline void Nuke_data::set_P(double P)         { _P = P; }

#endif //NUKE_DATA_H_
```

Файл NukeModel.h

```
// NUKE_MODEL.h

//

#ifndef NUKE_MODEL_H_
#define NUKE_MODEL_H_

#include "nuke_common.h"
#include "nuke_data.h"

class Nuke_model
{
public:
    static Nuke_model& ptr() { static Nuke_model model; return model; }
```

```
    void execute_until(timetype newtime);
    inline void execute_for(timetype timestep);

    inline void set_data(const Nuke_data& data);
    inline Nuke_data get_data() const;

    inline timetype get_cur_time() { return data.time(); }

private:  // calculations
    void calc_step(timetype tick);

private:    // data
    Nuke_data data;

    Nuke_model();
    Nuke_model(const Nuke_model&);
    Nuke_model& operator=(Nuke_model&);
};

// ------------- INLINES ----------------

inline void Nuke_model::set_data(const Nuke_data& data)
{
    this->data = data;
}

inline Nuke_data Nuke_model::get_data() const
{
    return data;
}

inline void Nuke_model::execute_for(timetype timestep)
{
    execute_until(data.time() + timestep);
}



#endif //NUKE_MODEL_H_
```

Файл NukeModel.cpp

```
#include "stdafx.h"
#include "nuke_common.h"
#include "nuke_model.h"

#include <math.h>

const timetype CALC_TICK = 1e-2;        // величина шага по времени при расчете модели

// физические постоянные
const double N_AVOGADRO = 6.02e+23;     // число Авогадро

// параметры модели
const double MODEL_PIVOT_EXP = 1;       // степень зависимости k от глубины погружения стержня

const double FUEL_MASS = 3000;          // масса топлива кг.

const long FUEL_ORD_NUMBER = 92;       // ???  порядковый номер U в таблице Менделеева

const double FUEL_MOLAR_MASS = 238;     // !!!  рассчитать, пока U 238

const double ATHOMS_COUNT = FUEL_MASS/FUEL_MOLAR_MASS *  N_AVOGADRO;
const long ATHOM_NEUTRONS_COUNT = FUEL_MOLAR_MASS - FUEL_ORD_NUMBER;

// постоянное число нейтронов в активной зоне х ATHOMS_COUNT
const double ZONE_NEUTRONS_COUNT_COEF = 1;

const double COEF_USEFUL_WORK = 30. / 100;      //  КПД реактора

const double HEAT_CARRIER_DENSITY = 1000;       // ??? ( не вода)

const double HEAT_CARRIER_CANAL_SQUARE = 1e5;//0.5;   // ???  площадь канала теплонос

const double HEAT_CARRIER_TERMAL_CAPACITY = 4200 + 5e09;   // ??? ( не вода) + плавление

const double INITIAL_HEAT_CARRIER_TEMPERATURE = 300;        //  комнатная + 27
```

```
//const double STABLE_WORKING_AREA_TEMPERATURE = 1000; //  как регулируемый параметр ???

const double REACTION_SIGMA = 1e5 * 1e19 * 1e-28;          //  сечение реакции 1 барн ???

const double ATHOM_DESTROY_ENERGY = 200 * 1e6 * 1.6e-19;  //  средняя энергия расщепления 1 атома

const double WORKING_AREA_TERMAL_CAPACITY = 500;    //  теплоемкость оболочки

const double WORKING_AREA_MASS = 2*10e8;             //  масса оболочки

const double SURROUNDING_AREA_TERMAL_CAPACITY = WORKING_AREA_TERMAL_CAPACITY; //  общая
теплоемкость

const double SURROUNDING_AREA_MASS = WORKING_AREA_MASS + FUEL_MASS;  //  общая масса

//const double INITIAL_SURROUNDING_AREA_TEMPERATURE = STABLE_WORKING_AREA_TEMPERATURE; // ???

//const double NEUTRON_GENERATION_LIFE_TIME = 1e-12;     //  не запаздывающие

const double NEUTRON_GENERATION_LIFE_TIME = 1e-10;     //

const double LAG_NEUTRON_PART = 0.0065;                //  доля запаздывающих

//const double SECONDS_IN_TICK = ;

Nuke_model::Nuke_model()
{}

// рассчитать коэф-т размножения на основе глубины погружения стержней
// 0..k..2-4, k - во сколько раз нейтронов в след. поколении больше, чем в пред.
// 0..h..100, h=100 - погружены послностью, h=0 - полностью выдвинуты
inline double calc_k(double h)
{
    return 1 + (0.7 - h/100.);
    //return  1 + (0.6 - h/100.) * ((MODEL_PIVOT_EXP - 1) > 0 ? pow( fabsl(0.6 - h/100.),
(MODEL_PIVOT_EXP - 1)) : 1);
}

// рассчитать примерное число нейтронов в активной зоне к концу периода времени
// FIXME: добавить запаздывающие нейтроны!!!!
// порядок величины ~ N_AVOGADRO
inline double calc_neutrons_count(double k, double peroid_count)
{
   return ZONE_NEUTRONS_COUNT_COEF * ATHOMS_COUNT * ATHOM_NEUTRONS_COUNT
            * pow(k, peroid_count);
}

// расчитать температуру теплоносителя, причем требуемая полезная мощность
// принимается как константа
// T в градусах Кельвина
// V в метрах за секунду
// P в ваттах
inline double calc_heat_carrier_temperature(double V_heat_carrier, double P_useful,
                                            double dt,
                                            double& P_surplus, double T_working_area = 0)
{
    // мощность оттока энергии при помощи теплоносителя
    double P_heat_carrier = P_useful / COEF_USEFUL_WORK;

    // масса греющегося теплоносителя
    double mass_heat_carrier = HEAT_CARRIER_DENSITY * dt * HEAT_CARRIER_CANAL_SQUARE
        * V_heat_carrier;

    // результирующая температура теплоносителя
    double Thc_final = P_heat_carrier / (HEAT_CARRIER_TERMAL_CAPACITY * mass_heat_carrier)
        - INITIAL_HEAT_CARRIER_TEMPERATURE;

    // излишек тепловой мощности, который не смог унести теплоноситель
    P_surplus = 0;
    if (Thc_final > T_working_area)
    {
        P_surplus = HEAT_CARRIER_TERMAL_CAPACITY * mass_heat_carrier * (Thc_final -
T_working_area);
        Thc_final = T_working_area;
    }

    return Thc_final;
}

/*  double _h;      // глубина погружения стержней, в процентах (0-100)
```

```
    double _k;      // коэффи+циент размножения (примерно равен 1)
    double _v;      // скорость обращения теплоносителя, в процентах (0-100)
    double _n;      // число вылетающих нейтронов, в процентах (0-100)
    double _Thc;    // температура теплоносителя, в процентах (0-100)
    // Не сделаны графики
    double _Twa;    // температура рабочей зоны
    double _N;      // тепловая мощность
    double _P;      // полезная мощность
*/

//
double calc_working_area_temperature(double T_working_area_old, double P_surplus, double dt)
{
    return 0;
}

// энергия в ед. времени
double calc_reaction_energy(double total_neutron_count, double k)
{
    return (k > 0 ? k : 0) * ATHOM_DESTROY_ENERGY * REACTION_SIGMA *
        (total_neutron_count > 0 ? total_neutron_count : 0);
}

// dt in seconds
double calc_work_and_heater_temperature(double prev_temperature, double total_reaction_energy,
                                        double V_heat_carrier, double dt, double
heat_carrier_grow)
{
/*
    // for heat carrier
    double k1 = HEAT_CARRIER_TERMAL_CAPACITY * HEAT_CARRIER_DENSITY * HEAT_CARRIER_CANAL_SQUARE
        * V_heat_carrier * dt;

    // for working area
    double k2 = WORKING_AREA_TERMAL_CAPACITY * WORKING_AREA_MASS;

    // for surrounding radiators
    double k3 = SURROUNDING_AREA_TERMAL_CAPACITY * SURROUNDING_AREA_MASS;

    return (total_reaction_energy + k1 * INITIAL_HEAT_CARRIER_TEMPERATURE +
              k2 * prev_temperature + k3 * INITIAL_SURROUNDING_AREA_TEMPERATURE)
              / (k1 + k2 + k3);
*/
    const double val_hc = HEAT_CARRIER_TERMAL_CAPACITY * HEAT_CARRIER_DENSITY *
HEAT_CARRIER_CANAL_SQUARE
        * V_heat_carrier * dt;

    const double val_sa = SURROUNDING_AREA_TERMAL_CAPACITY * SURROUNDING_AREA_MASS;

    return (total_reaction_energy + val_hc * (INITIAL_HEAT_CARRIER_TEMPERATURE+heat_carrier_grow)
+
        val_sa * prev_temperature) / (val_hc + val_sa);

}

// рассчитать параметры модели на основе текущей Nuke_data через время tick
// tick - время в тиках
void Nuke_model::calc_step(timetype tick)
{
    // input: data on data.time
    // data.h       // глубина погружения стержней, в процентах (0-100)
    // data.v       // скорость обращения теплоносителя, в процентах (0-100)

    data._k = calc_k(data._h + data._dh);

    // кол-во теплоты цепной ядерной реакции

    double generations_count = tick * SECONDS_IN_TICK / NEUTRON_GENERATION_LIFE_TIME;
    // количество нейтронов в наличии к концу расчетного периода
    double N = calc_neutrons_count(data._k + data._dk, generations_count);

    // тут рассчитаем среднее число вылетающих нейтронов через поверхность
    data._n = pow(N, 2/3.);     // !!! FIXME: по-другому

    // вся энергия в ед. времени, выделяемая ядерной реакцией
    data._N = calc_reaction_energy(N + data._dn, data._k + data._dk);

    // тут надо написать дифур, выражающий зависимость
    // _P, _Thc, _Twc от остальных параметров и от собственных предыдущих
    // значений
```

```
    //....
    // _P было задано как рекомендованная полезная мощность. Но может меняться в
    // случае недостаточности, например скорости обращения теплоносителя
    // _Thc изменяет свою температуру от начальной до некоторой, не выше _Twa
    // _Twa зависит от количества теплоты, забранной теплоносителем, количества
    // рассеиваемого реактором тепла

    data._Thc = data._Twa = calc_work_and_heater_temperature(data._Twa + data._dTwa, data._N,
        data._v + data._dv, SECONDS_IN_TICK * tick, data._dThc);

    data._P = COEF_USEFUL_WORK * data._N;
}

void Nuke_model::execute_until(timetype newtime)
{
    while (data.time() < newtime)
    {
        calc_step(CALC_TICK);
        data.inc_time(CALC_TICK);
    }
}
```

Файл Visio\Automates\common.h

```
//--- this file is machine generated ---

#ifndef CommonH
#define CommonH

#include "types.h"
#include "..\\..\\Nuke_model.h"

typedef struct{
  ubyte y0; // Главный автомат управления реактором
  ubyte y1; // Автомат управления теплоносителем
  ubyte y2; // Автомат управления стержнями
  ubyte y3; // Автомат управления запуском
  ubyte y4; // Автомат управления остановом
  ubyte y5; // Автомат аварийного управления остановом
} common_t;

extern common_t cm;

// Автоматы A
void A0( ubyte e );
void A1( ubyte e );
void A2( ubyte e );
void A3( ubyte e );
void A4( ubyte e );
void A5( ubyte e );

// Переменные X
ubyte x10(void);
ubyte x11(void);
ubyte x12(void);
ubyte x13(void);
ubyte x14(void);
ubyte x20(void);
ubyte x21(void);
ubyte x22(void);
ubyte x23(void);
ubyte x24(void);
ubyte x30(void);
ubyte x31(void);
ubyte x40(void);
ubyte x50(void);
ubyte x60(void);
ubyte x70(void);
ubyte x80(void);

// Действия Z
void z100(void);
void z101(void);
void z102(void);
void z200(void);
void z201(void);
void z210(void);
void z211(void);
void z220(void);
void z221(void);
```

```
void z230(void);
void z310(void);
void z311(void);
void z320(void);
void z321(void);
void z330(void);
void z331(void);
void z340(void);
void z341(void);
void z350(void);
void z351(void);

// События E
// e0 - _инициализация_
// e10
// e100
// e101

// TODO:

static timetype T_time;
static timetype n_time;
static timetype side_sys_init_time;

#endif
```

Файл Visio\Automates\common.cpp

```
//--- this file is machine generated ---

#include "stdafx.h"
#include "common.h"
#include "log.h"

common_t cm;

//-----------------------------------------------------------
// A0 - Главный автомат управления реактором
//-----------------------------------------------------------
void A0( ubyte e )
{
  ubyte y_old = cm.y0;

  #ifdef A0_BEGIN_LOGGING
    log_a_begin(0, y_old, e);
  #endif

  switch( cm.y0 )
  {
    case 0:// Ожидание запуска
      if((e == 100))
      {
        z100();
                                        cm.y0 = 1;
      }
    break;

    case 1:// Запуск
      if(cm.y3 == 3)
      {
                                        cm.y0 = 2;
      }
      else
      if((e == 101))
      {
                                        cm.y0 = 3;
      }
      else
      if((e == 10))
      {
        z331();
      }
    break;

    case 2:// Работа
      if(cm.y1 == 3||cm.y2 == 3)
      {
                                        cm.y0 = 4;
      }
      else
```

```
      if((e == 101))
      {
                                        cm.y0 = 3;
      }
      else
      if((e == 10))
      {
        z311(); z321();
      }
    break;

    case 3:// Штатная остановка
      if((e == 10))
      {
        z341();
      }
    break;

    case 4:// Аварийная отстановка
      if((e == 10))
      {
        z351();
      }
    break;

    default:
      #ifdef A0_ERRORS_LOGGING
        log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A0: неизвестный номер состояния!");
      #else
        ;
      #endif
  }

  if( y_old == cm.y0 ) goto A0_end;

  #ifdef A0_TRANS_LOGGING
    log_a_trans(0, y_old, cm.y0);
  #endif

  switch( cm.y0 )
  {
    case 0:// Ожидание запуска
    break;

    case 1:// Запуск
      z330();
    break;

    case 2:// Работа
      z310(); z320();
    break;

    case 3:// Штатная остановка
      z101(); z340();
    break;

    case 4:// Аварийная отстановка
      z101(); z350();
    break;

  }

  A0_end: ;
  #ifdef A0_END_LOGGING
    log_a_end(0, cm.y0, e);
  #endif
}
//-----------------------------------------------------------
// A1 - Автомат управления теплоносителем
//-----------------------------------------------------------
void A1( ubyte e )
{
  ubyte y_old = cm.y1;

  #ifdef A1_BEGIN_LOGGING
    log_a_begin(1, y_old, e);
  #endif

  switch( cm.y1 )
  {
```

```
  case 0:// Ожидание инициализации
    if((e == 0))
    {
                                      cm.y1 = 1;
    }
  break;

  case 1:// Норма
    if(x13())
    {
      z200();
    }
    else
    if(x11())
    {
      z201();
    }
    else
    if(x10())
    {
      z200();
                                      cm.y1 = 2;
    }
    else
    if(x14())
    {
      z200();
                                      cm.y1 = 3;
    }
  break;

  case 2:// Неустойчивое состояние
    if(x13())
    {
      z200(); z220();
    }
    else
    if(x11())
    {
      z201(); z220();
    }
    else
    if(x12()&&x30())
    {
                                      cm.y1 = 1;
    }
    else
    if(x14())
    {
      z200();
                                      cm.y1 = 3;
    }
    else
    if(x10()&&!x23())
    {
      z201();
    }
  break;

  case 3:// Авария
  break;

  default:
    #ifdef A1_ERRORS_LOGGING
      log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A1: неизвестный номер состояния!");
    #else
      ;
    #endif
}

if( y_old == cm.y1 ) goto A1_end;

#ifdef A1_TRANS_LOGGING
  log_a_trans(1, y_old, cm.y1);
#endif

switch( cm.y1 )
{
  case 0:// Ожидание инициализации
  break;
```

```
    case 1:// Норма
    break;

    case 2:// Неустойчивое состояние
      z220();
    break;

    case 3:// Авария
    break;

  }

  A1_end: ;
  #ifdef A1_END_LOGGING
    log_a_end(1, cm.y1, e);
  #endif
}
//-----------------------------------------------------------
// A2 - Автомат управления стержнями
//-----------------------------------------------------------
void A2( ubyte e )
{
  ubyte y_old = cm.y2;

  #ifdef A2_BEGIN_LOGGING
    log_a_begin(2, y_old, e);
  #endif

  switch( cm.y2 )
  {
    case 0:// Ожидание инициализации
      if((e == 0))
      {
                                        cm.y2 = 1;
      }
    break;

    case 1:// Норма
      if(x23())
      {
        z210();
      }
      else
      if(x21())
      {
        z211();
      }
      else
      if(x20())
      {
        z211();
                                        cm.y2 = 2;
      }
      else
      if(x24())
      {
        z210();
                                        cm.y2 = 3;
      }
    break;

    case 2:// Неустойчивое состояние
      if(x23())
      {
        z210(); z221();
      }
      else
      if(x21())
      {
        z211(); z221();
      }
      else
      if(x22()&&x31())
      {
                                        cm.y2 = 1;
      }
      else
      if(x24())
      {
```

```
        z210();
                                        cm.y2 = 3;
      }
      else
      if(x20()&&!x13())
      {
        z211();
      }
    break;

    case 3:// Авария
    break;

    default:
      #ifdef A2_ERRORS_LOGGING
        log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A2: неизвестный номер состояния!");
      #else
        ;
      #endif
  }

  if( y_old == cm.y2 ) goto A2_end;

  #ifdef A2_TRANS_LOGGING
    log_a_trans(2, y_old, cm.y2);
  #endif

  switch( cm.y2 )
  {
    case 0:// Ожидание инициализации
    break;

    case 1:// Норма
    break;

    case 2:// Неустойчивое состояние
      z221();
    break;

    case 3:// Авария
    break;

  }

  A2_end: ;
  #ifdef A2_END_LOGGING
    log_a_end(2, cm.y2, e);
  #endif
}
//-----------------------------------------------------------
// A3 - Автомат управления запуском
//-----------------------------------------------------------
void A3( ubyte e )
{
  ubyte y_old = cm.y3;

  #ifdef A3_BEGIN_LOGGING
    log_a_begin(3, y_old, e);
  #endif

  switch( cm.y3 )
  {
    case 0:// Ожидание инициализации
      if((e == 0))
      {
        z230();
                                        cm.y3 = 1;
      }
    break;

    case 1:// Предпусковая подготовка третьих систем
      if(x40())
      {
                                        cm.y3 = 2;
      }
      else
      if((e == 10))
      {
      }
    break;
```

```
    case 2:// Начальный <разгон> теплоносителя
      if(x50())
      {
                                        cm.y3 = 3;
      }
      else
      if((e == 10))
      {
        z200();
      }
    break;

    case 3:// Запуск произведен
    break;

    default:
      #ifdef A3_ERRORS_LOGGING
        log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A3: неизвестный номер состояния!");
      #else
        ;
      #endif
  }

  if( y_old == cm.y3 ) goto A3_end;

  #ifdef A3_TRANS_LOGGING
    log_a_trans(3, y_old, cm.y3);
  #endif

  switch( cm.y3 )
  {
    case 0:// Ожидание инициализации
    break;

    case 1:// Предпусковая подготовка третьих систем
    break;

    case 2:// Начальный <разгон> теплоносителя
    break;

    case 3:// Запуск произведен
    break;

  }

  A3_end: ;
  #ifdef A3_END_LOGGING
    log_a_end(3, cm.y3, e);
  #endif
}
//-----------------------------------------------------------
// A4 - Автомат управления остановом
//-----------------------------------------------------------
void A4( ubyte e )
{
  ubyte y_old = cm.y4;

  #ifdef A4_BEGIN_LOGGING
    log_a_begin(4, y_old, e);
  #endif

  switch( cm.y4 )
  {
    case 0:// Ожидание инициализации
      if((e == 0))
      {
                                        cm.y4 = 1;
      }
    break;

    case 1:// Задвижение стержней
      if(x80())
      {
                                        cm.y4 = 2;
      }
      else
      if((e == 10))
      {
        z210();
```

```
      }
    break;

    case 2:// Охлаждение
      if(x70())
      {
                                        cm.y4 = 3;
      }
      else
      if(x60())
      {
        z200();
      }
    break;

    case 3:// <Торможение> теплоносителя
      if(x70())
      {
        z201();
      }
      else
      if(!x70())
      {
        z200(); z102();
      }
    break;

    default:
      #ifdef A4_ERRORS_LOGGING
        log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A4: неизвестный номер состояния!");
      #else
        ;
      #endif
  }

  if( y_old == cm.y4 ) goto A4_end;

  #ifdef A4_TRANS_LOGGING
    log_a_trans(4, y_old, cm.y4);
  #endif

  switch( cm.y4 )
  {
    case 0:// Ожидание инициализации
    break;

    case 1:// Задвижение стержней
    break;

    case 2:// Охлаждение
    break;

    case 3:// <Торможение> теплоносителя
    break;

  }

  A4_end: ;
  #ifdef A4_END_LOGGING
    log_a_end(4, cm.y4, e);
  #endif
}
//-----------------------------------------------------------
// A5 - Автомат аварийного управления остановом
//-----------------------------------------------------------
void A5( ubyte e )
{
  ubyte y_old = cm.y5;

  #ifdef A5_BEGIN_LOGGING
    log_a_begin(5, y_old, e);
  #endif

  switch( cm.y5 )
  {
    case 0:// Ожидание инициализации
      if((e == 0))
      {
        z102();
                                        cm.y5 = 1;
```

```
      }
    break;

    case 1:// Принятие экстренных мер
      if(x80())
      {
                                        cm.y5 = 2;
      }
      else
      if((e == 10))
      {
        z200(); z210();
      }
    break;

    case 2:// Охлаждение
      if(!x70())
      {
        z200();
      }
      else
      if(x70())
      {
                                        cm.y5 = 3;
      }
    break;

    case 3:// <Торможение> теплоносителя
      if(x70())
      {
        z201();
      }
      else
      if(!x70())
      {
        z200();
      }
    break;

    default:
      #ifdef A5_ERRORS_LOGGING
        log_write(LOG_GRAPH_ERROR, "Ошибка в автомате A5: неизвестный номер состояния!");
      #else
        ;
      #endif
  }

  if( y_old == cm.y5 ) goto A5_end;

  #ifdef A5_TRANS_LOGGING
    log_a_trans(5, y_old, cm.y5);
  #endif

  switch( cm.y5 )
  {
    case 0:// Ожидание инициализации
    break;

    case 1:// Принятие экстренных мер
    break;

    case 2:// Охлаждение
    break;

    case 3:// <Торможение> теплоносителя
    break;

  }

  A5_end: ;
  #ifdef A5_END_LOGGING
    log_a_end(5, cm.y5, e);
  #endif
}
```

Файл Visio\Automates\log.h

```
//--- this file is machine generated ---

#ifndef LogH
#define LogH

#include "types.h"

typedef struct{
  ubyte       dig;
  const char* n;
  const char* n_name;
} int_str2_t;

typedef struct{
  const char* n;
  const char* n_name;
} str2_t;

typedef struct{
  const char* n;
  const char* n_name;
  str2_t* str2;
} str3_t;

extern str2_t a0_str2[5];
extern str2_t a1_str2[4];
extern str2_t a2_str2[4];
extern str2_t a3_str2[4];
extern str2_t a4_str2[4];
extern str2_t a5_str2[4];
extern str3_t A_str3[6];

#define SWITCH_LOGGING

#ifdef SWITCH_LOGGING
  #define Z_LOGGING
  #define X_LOGGING
  #define A_BEGINS_LOGGING
  #define A_TRANS_LOGGING
  #define A_ENDS_LOGGING
  #define A_ERRORS_LOGGING

  enum{
    LOG_Z = '*',
    LOG_X = '>',
    LOG_GRAPH_BEGIN = '{',
    LOG_GRAPH_TRANS = 'T',
    LOG_GRAPH_END = '}',
    LOG_GRAPH_ERROR = 'E'
  };

  void log_a_begin(ubyte a, ubyte y, ubyte e);
  void log_a_trans(ubyte a, ubyte yo, ubyte yn);
  void log_a_end(ubyte a, ubyte y, ubyte e);
  void log_x(ubyte x, ubyte res);
  void log_z(ubyte z);
  void log_write(char, const char* str);
#endif

#ifdef A_BEGINS_LOGGING
  #define A0_BEGIN_LOGGING
  #define A1_BEGIN_LOGGING
  #define A2_BEGIN_LOGGING
  #define A3_BEGIN_LOGGING
  #define A4_BEGIN_LOGGING
  #define A5_BEGIN_LOGGING
#endif

#ifdef A_TRANS_LOGGING
  #define A0_TRANS_LOGGING
  #define A1_TRANS_LOGGING
  #define A2_TRANS_LOGGING
  #define A3_TRANS_LOGGING
  #define A4_TRANS_LOGGING
  #define A5_TRANS_LOGGING
#endif

#ifdef A_ENDS_LOGGING
  #define A0_END_LOGGING
  #define A1_END_LOGGING
```

```
  #define A2_END_LOGGING
  #define A3_END_LOGGING
  #define A4_END_LOGGING
  #define A5_END_LOGGING
#endif

#ifdef A_ERRORS_LOGGING
  #define A0_ERRORS_LOGGING
  #define A1_ERRORS_LOGGING
  #define A2_ERRORS_LOGGING
  #define A3_ERRORS_LOGGING
  #define A4_ERRORS_LOGGING
  #define A5_ERRORS_LOGGING
#endif

#endif
```

Файл Visio\Automates\x.cpp

```
//--- this file is machine generated ---

#include "stdafx.h"
#include "common.h"
#include "log.h"

//--------------------------------------------------------------------------
ubyte x10_user(void);
ubyte x10(void)
{
  ubyte b = x10_user();

  #ifdef X_LOGGING
    log_x(0, b);
  #endif

  return b;
}
//--------------------------------------------------------------------------
ubyte x11_user(void);
ubyte x11(void)
{
  ubyte b = x11_user();

  #ifdef X_LOGGING
    log_x(1, b);
  #endif

  return b;
}
//--------------------------------------------------------------------------
ubyte x12_user(void);
ubyte x12(void)
{
  ubyte b = x12_user();

  #ifdef X_LOGGING
    log_x(2, b);
  #endif

  return b;
}
//--------------------------------------------------------------------------
ubyte x13_user(void);
ubyte x13(void)
{
  ubyte b = x13_user();

  #ifdef X_LOGGING
    log_x(3, b);
  #endif

  return b;
}
//--------------------------------------------------------------------------
ubyte x14_user(void);
ubyte x14(void)
{
  ubyte b = x14_user();

  #ifdef X_LOGGING
```

```
    log_x(4, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x20_user(void);
ubyte x20(void)
{
  ubyte b = x20_user();

  #ifdef X_LOGGING
    log_x(5, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x21_user(void);
ubyte x21(void)
{
  ubyte b = x21_user();

  #ifdef X_LOGGING
    log_x(6, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x22_user(void);
ubyte x22(void)
{
  ubyte b = x22_user();

  #ifdef X_LOGGING
    log_x(7, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x23_user(void);
ubyte x23(void)
{
  ubyte b = x23_user();

  #ifdef X_LOGGING
    log_x(8, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x24_user(void);
ubyte x24(void)
{
  ubyte b = x24_user();

  #ifdef X_LOGGING
    log_x(9, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x30_user(void);
ubyte x30(void)
{
  ubyte b = x30_user();

  #ifdef X_LOGGING
    log_x(10, b);
  #endif

  return b;
}
//-----------------------------------------------------------------------------
ubyte x31_user(void);
ubyte x31(void)
```

```
{
  ubyte b = x31_user();

  #ifdef X_LOGGING
    log_x(11, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
ubyte x40_user(void);
ubyte x40(void)
{
  ubyte b = x40_user();

  #ifdef X_LOGGING
    log_x(12, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
ubyte x50_user(void);
ubyte x50(void)
{
  ubyte b = x50_user();

  #ifdef X_LOGGING
    log_x(13, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
ubyte x60_user(void);
ubyte x60(void)
{
  ubyte b = x60_user();

  #ifdef X_LOGGING
    log_x(14, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
ubyte x70_user(void);
ubyte x70(void)
{
  ubyte b = x70_user();

  #ifdef X_LOGGING
    log_x(15, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
ubyte x80_user(void);
ubyte x80(void)
{
  ubyte b = x80_user();

  #ifdef X_LOGGING
    log_x(16, b);
  #endif

  return b;
}
//---------------------------------------------------------------------------
```

Файл Visio\Automates\x_user.cpp

```
#include "stdafx.h"
#include "common.h"

//#include "..\\..\\resource.h"
//#include "..\\..\\NukeDoc.h"
//#include "..\\..\\NukeView.h"
```

```
//#include "..\\..\\Nuke_Model.h"
#include "..\\..\\Nuke_control_system.h"
#include "..\\..\\Nuke_model.h"

//---------------------------------------------------------------------------
ubyte x10_user(void)
{
    // T-ра критически низкая
    double val = Nuke_model::ptr().get_data().Twa();
    return val <= WORKING_AREA_TEMPERATURE_M2;
}
//---------------------------------------------------------------------------
ubyte x11_user(void)
{
    // T-ра ниже нормы
    double val = Nuke_model::ptr().get_data().Twa();
    return (val > WORKING_AREA_TEMPERATURE_M2) && (val <= WORKING_AREA_TEMPERATURE_M1);
}
//---------------------------------------------------------------------------
ubyte x12_user(void)
{
    // T-ра в норме
    double val = Nuke_model::ptr().get_data().Twa();
    return (val > WORKING_AREA_TEMPERATURE_M1) && (val < WORKING_AREA_TEMPERATURE_1);
}
//---------------------------------------------------------------------------
ubyte x13_user(void)
{
    // T-ра выше нормы
    double val = Nuke_model::ptr().get_data().Twa();
    return (val >= WORKING_AREA_TEMPERATURE_1) && (val < WORKING_AREA_TEMPERATURE_2);
}
//---------------------------------------------------------------------------
ubyte x14_user(void)
{
    // T-ра критически превышена
    double val = Nuke_model::ptr().get_data().Twa();
    return (val >= WORKING_AREA_TEMPERATURE_2);
}

ubyte x20_user(void)
{
    // Число н-нов критически низкое
    double val = Nuke_model::ptr().get_data().n();
    return val <= NEUTRON_COUNT_VALUE_M2;
}
//---------------------------------------------------------------------------
ubyte x21_user(void)
{
    // Число н-нов ниже нормы
    double val = Nuke_model::ptr().get_data().n();
    return (val > NEUTRON_COUNT_VALUE_M2) && (val <= NEUTRON_COUNT_VALUE_M1);
}
//---------------------------------------------------------------------------
ubyte x22_user(void)
{
    // Число н-нов в норме
    double val = Nuke_model::ptr().get_data().n();
    return (val > NEUTRON_COUNT_VALUE_M1) && (val < NEUTRON_COUNT_VALUE_1);
}
//---------------------------------------------------------------------------
ubyte x23_user(void)
{
    // Число н-нов выше нормы
    double val = Nuke_model::ptr().get_data().n();
    return (val >= NEUTRON_COUNT_VALUE_1) && (val < NEUTRON_COUNT_VALUE_2);
}
//---------------------------------------------------------------------------
ubyte x24_user(void)
{
    // Число н-нов критически превышено
    double val = Nuke_model::ptr().get_data().n();
    return (val >= NEUTRON_COUNT_VALUE_2);
}

//---------------------------------------------------------------------------
ubyte x30_user(void)
{
    // Требуется выход из неусточивого состояния для A1
    return Nuke_model::ptr().get_cur_time() - T_time > UNSTABLE_T_TIME;
```

```
}
//---------------------------------------------------------------------------
ubyte x31_user(void)
{
    // Требуется выход из неусточивого состояния для A2
    return Nuke_model::ptr().get_cur_time() - n_time > UNSTABLE_N_TIME;
}
//---------------------------------------------------------------------------
ubyte x40_user(void)
{
    // Готовность третьих систем
    return Nuke_model::ptr().get_cur_time() - side_sys_init_time > SIDE_SYS_INIT_TIME;
}
//---------------------------------------------------------------------------
ubyte x50_user(void)
{
    // Скорость т-ля >= начальной скорости теплоносителя
    double val = Nuke_model::ptr().get_data().v();
    return val >= HEAT_CARRIER_INITIAL_VELOCITY;
}
//---------------------------------------------------------------------------
ubyte x60_user(void)
{
    // Скорость т-ля < требуемой скорости т-я для останова
    double val = Nuke_model::ptr().get_data().v();
    return val < HEAT_CARRIER_BOUND_VELOCITY;
}
//---------------------------------------------------------------------------
ubyte x70_user(void)
{
    // T-ра <= T-ры останова
    double val = Nuke_model::ptr().get_data().Twa();
    return val < NO_HEAT_TEMPERATURE;
}
//---------------------------------------------------------------------------
ubyte x80_user(void)
{
    // Стержни задвинуты полностью
    double val = Nuke_model::ptr().get_data().h();
    return (val - 100) > -EPS;
}
//---------------------------------------------------------------------------
```

Файл Visio\Automates\z.cpp

```
//--- this file is machine generated ---

#include "stdafx.h"
#include "common.h"
#include "log.h"

//---------------------------------------------------------------------------
void z100_user(void);
void z100(void)
{
  z100_user();

  #ifdef Z_LOGGING
    log_z(0);
  #endif
}
//---------------------------------------------------------------------------
void z101_user(void);
void z101(void)
{
  z101_user();

  #ifdef Z_LOGGING
    log_z(1);
  #endif
}
//---------------------------------------------------------------------------
void z102_user(void);
void z102(void)
{
  z102_user();

  #ifdef Z_LOGGING
    log_z(2);
  #endif
```

```
}
//------------------------------------------------------------------------------
void z200_user(void);
void z200(void)
{
  z200_user();

  #ifdef Z_LOGGING
    log_z(3);
  #endif
}
//------------------------------------------------------------------------------
void z201_user(void);
void z201(void)
{
  z201_user();

  #ifdef Z_LOGGING
    log_z(4);
  #endif
}
//------------------------------------------------------------------------------
void z210_user(void);
void z210(void)
{
  z210_user();

  #ifdef Z_LOGGING
    log_z(5);
  #endif
}
//------------------------------------------------------------------------------
void z211_user(void);
void z211(void)
{
  z211_user();

  #ifdef Z_LOGGING
    log_z(6);
  #endif
}
//------------------------------------------------------------------------------
void z220_user(void);
void z220(void)
{
  z220_user();

  #ifdef Z_LOGGING
    log_z(7);
  #endif
}
//------------------------------------------------------------------------------
void z221_user(void);
void z221(void)
{
  z221_user();

  #ifdef Z_LOGGING
    log_z(8);
  #endif
}
//------------------------------------------------------------------------------
void z230_user(void);
void z230(void)
{
  z230_user();

  #ifdef Z_LOGGING
    log_z(9);
  #endif
}
//------------------------------------------------------------------------------
void z310_user(void);
void z310(void)
{
  z310_user();

  #ifdef Z_LOGGING
    log_z(10);
  #endif
```

```
}
//---------------------------------------------------------------------------
void z311_user(void);
void z311(void)
{
  z311_user();

  #ifdef Z_LOGGING
    log_z(11);
  #endif
}
//---------------------------------------------------------------------------
void z320_user(void);
void z320(void)
{
  z320_user();

  #ifdef Z_LOGGING
    log_z(12);
  #endif
}
//---------------------------------------------------------------------------
void z321_user(void);
void z321(void)
{
  z321_user();

  #ifdef Z_LOGGING
    log_z(13);
  #endif
}
//---------------------------------------------------------------------------
void z330_user(void);
void z330(void)
{
  z330_user();

  #ifdef Z_LOGGING
    log_z(14);
  #endif
}
//---------------------------------------------------------------------------
void z331_user(void);
void z331(void)
{
  z331_user();

  #ifdef Z_LOGGING
    log_z(15);
  #endif
}
//---------------------------------------------------------------------------
void z340_user(void);
void z340(void)
{
  z340_user();

  #ifdef Z_LOGGING
    log_z(16);
  #endif
}
//---------------------------------------------------------------------------
void z341_user(void);
void z341(void)
{
  z341_user();

  #ifdef Z_LOGGING
    log_z(17);
  #endif
}
//---------------------------------------------------------------------------
void z350_user(void);
void z350(void)
{
  z350_user();

  #ifdef Z_LOGGING
    log_z(18);
  #endif
```

```
}
//---------------------------------------------------------------------------
void z351_user(void);
void z351(void)
{
  z351_user();

  #ifdef Z_LOGGING
    log_z(19);
  #endif
}
//---------------------------------------------------------------------------
```

Файл Visio\Automates\z_user.cpp

```
#include "stdafx.h"
#include "common.h"

#include "..\\..\\Nuke_control_system.h"

#include "..\\..\\resource.h"
#include "..\\..\\MainFrm.h"
#include "..\\..\\NukeDoc.h"
#include "..\\..\\NukeView.h"

//---------------------------------------------------------------------------
void z100_user(void)
{
    // Сделать кнопку "СТАРТ" недоступной
    try
    {
        CMainFrame *pMainFrm = (CMainFrame *)AfxGetMainWnd();
        CNukeView *pNukeView = (CNukeView *)pMainFrm->GetActiveView();
        pNukeView->m_dlgControlPanelMain.GetDlgItem( IDC_BTN_START )->EnableWindow( FALSE );
    }
    catch(...)
    {
    }
}
//---------------------------------------------------------------------------
void z101_user(void)
{
    // Сделать кнопку "СТОП" недоступной
    try
    {
        CMainFrame *pMainFrm = (CMainFrame *)AfxGetMainWnd();
        CNukeView *pNukeView = (CNukeView *)pMainFrm->GetActiveView();
        pNukeView->m_dlgControlPanelMain.GetDlgItem( IDC_BTN_STOP )->EnableWindow( FALSE );
    }
    catch(...)
    {
    }
}
//---------------------------------------------------------------------------
void z102_user(void)
{
    // Включить аварийный звуковой сигнал (долговременно)
}
//---------------------------------------------------------------------------
void z200_user(void)
{
    // Увеличить скорость т-ля (понизить т-ру)
    Csystem_data data(Nuke_model::ptr().get_data());
    data.cd_inc_v();
    Nuke_model::ptr().set_data( data );
}
//---------------------------------------------------------------------------
void z201_user(void)
{
    // Уменьшить скорость т-ля (повысить т-ру)
    Csystem_data data(Nuke_model::ptr().get_data());
    data.cd_dec_v();
    Nuke_model::ptr().set_data( data );
}
//---------------------------------------------------------------------------
void z210_user(void)
{
    // Увеличить глубину погружения стержней (понизить число н-нов)
    Csystem_data data(Nuke_model::ptr().get_data());
    data.cd_inc_h();
```

```
    Nuke_model::ptr().set_data( data );
}
//---------------------------------------------------------------------------
void z211_user(void)
{
    // Уменьшить глубину погружения стержней (повысить число н-нов)
    Csystem_data data(Nuke_model::ptr().get_data());
    data.cd_dec_h();
    Nuke_model::ptr().set_data( data );
}
//---------------------------------------------------------------------------
void z220_user(void)
{
    // Обнуление счетчика выхода из неустойчивого состояния для A1
    T_time = Nuke_model::ptr().get_cur_time();
}
//---------------------------------------------------------------------------
void z221_user(void)
{
    // Обнуление счетчика выхода из неустойчивого состояния для A2
    n_time = Nuke_model::ptr().get_cur_time();
}
//---------------------------------------------------------------------------
void z230_user(void)
{
    // Обнуление счетчика готовности третьих систем
    side_sys_init_time = Nuke_model::ptr().get_cur_time();
}
//---------------------------------------------------------------------------
void z310_user(void)
{
    // Инициализация A1
    A1(0);
}
//---------------------------------------------------------------------------
void z311_user(void)
{
    // Работа A1
    A1(10);
}
//---------------------------------------------------------------------------
void z320_user(void)
{
    // Инициализация A2
    A2(0);
}
//---------------------------------------------------------------------------
void z321_user(void)
{
    // Работа A2
    A2(10);
}
//---------------------------------------------------------------------------
void z330_user(void)
{
    // Инициализация A3
    A3(0);
}
//---------------------------------------------------------------------------
void z331_user(void)
{
    // Работа A3
    A3(10);
}
//---------------------------------------------------------------------------
void z340_user(void)
{
    // Инициализация A4
    A4(0);
}
//---------------------------------------------------------------------------
void z341_user(void)
{
    // Работа A4
    A4(10);
}
//---------------------------------------------------------------------------
void z350_user(void)
{
    // Инициализация A5
```

```
    A5(0);
}
//---------------------------------------------------------------------------
void z351_user(void)
{
    // Работа A5
    A5(10);
}
//---------------------------------------------------------------------------
```

Файл Visio\Automates\log.cpp

```
//--- this file is machine generated ---

#include "stdafx.h"
#include "log.h"

#ifdef SWITCH_LOGGING

int_str2_t e_str2[4] =
{
  {  0, "e0", "_инициализация_" },
  {  10, "e10", "Системный таймер" },
  {  100, "e100", "Нажатие кнопки \"Пуск\"" },
  {  101, "e101", "Нажатие кнопки \"Стоп\"" }
};

str2_t x_str2[17] =
{
  { "x10", "Температура критически низкая" },
  { "x11", "Температура ниже нормы" },
  { "x12", "Температура в норме" },
  { "x13", "Температура выше нормы" },
  { "x14", "Температура критически превышена" },
  { "x20", "Число нейтронов критически низкое" },
  { "x21", "Число нейтронов ниже нормы" },
  { "x22", "Число нейтронов в норме" },
  { "x23", "Число нейтронов выше нормы" },
  { "x24", "Число нейтронов критически превышено" },
  { "x30", "Требуется выход из неустойчивого состояния для A1" },
  { "x31", "Требуется выход из неустойчивого состояния для A2" },
  { "x40", "Готовность третьих систем" },
  { "x50", "Скорость теплоносителя>= начальной скорости теплоносителя" },
  { "x60", "Скорость теплоносителя < требуемой скорости теплоносителя для останова" },
  { "x70", "Температура <= температуры останова" },
  { "x80", "Скорость теплоносителя максимальна" }
};

str2_t z_str2[20] =
{
  { "z100", "Сделать кнопку \"СТАРТ\" недоступной" },
  { "z101", "Сделать кнопку \"СТОП\" недоступной" },
  { "z102", "Включить аварийный звуковой сигнал" },
  { "z200", "Увеличить скорость теплоносителя (понизить температуру)" },
  { "z201", "Уменьшить скорость теплоносителя (повысить температуру)" },
  { "z210", "Увеличить глубину погружения стержней (понизить число нейтронов)" },
  { "z211", "Уменьшить глубину погружения стержней (повысить число нейтронов)" },
  { "z220", "Обнуление счетчика выхода из неустойчивого состояния для A1" },
  { "z221", "Обнуление счетчика выхода из неустойчивого состояния для A2" },
  { "z230", "Обнуление счетчика готовности третьих систем" },
  { "z310", "Инициализация A1: вызов A1(e0)" },
  { "z311", "Работа A1: вызов A1(e10)" },
  { "z320", "Инициализация A2: вызов A2(e0)" },
  { "z321", "Работа A2: вызов A2(e10)" },
  { "z330", "Инициализация A3: вызов A3(e0)" },
  { "z331", "Работа A3: вызов A3(e10)" },
  { "z340", "Инициализация A4: вызов A4(e0)" },
  { "z341", "Работа A4: вызов A4(e10)" },
  { "z350", "Инициализация A5: вызов A5(e0)" },
  { "z351", "Работа A5: вызов A5(e10)" }
};

str2_t a0_str2[5] =
{
  { "0", "Ожидание запуска" },
  { "1", "Запуск" },
  { "2", "Работа" },
  { "3", "Штатная остановка" },
  { "4", "Аварийная отстановка" }
```

```
};

str2_t a1_str2[4] =
{
  { "0", "Ожидание инициализации" },
  { "1", "Норма" },
  { "2", "Неустойчивое состояние" },
  { "3", "Авария" }
};

str2_t a2_str2[4] =
{
  { "0", "Ожидание инициализации" },
  { "1", "Норма" },
  { "2", "Неустойчивое состояние" },
  { "3", "Авария" }
};

str2_t a3_str2[4] =
{
  { "0", "Ожидание инициализации" },
  { "1", "Предпусковая подготовка третьих систем" },
  { "2", "Начальный <разгон> теплоносителя" },
  { "3", "Запуск произведен" }
};

str2_t a4_str2[4] =
{
  { "0", "Ожидание инициализации" },
  { "1", "Задвижение стержней" },
  { "2", "Охлаждение" },
  { "3", "<Торможение> теплоносителя" }
};

str2_t a5_str2[4] =
{
  { "0", "Ожидание инициализации" },
  { "1", "Принятие экстренных мер" },
  { "2", "Охлаждение" },
  { "3", "<Торможение> теплоносителя" }
};

str3_t A_str3[6] =
{
  { "A0", "Главный автомат управления реактором", a0_str2 },
  { "A1", "Автомат управления теплоносителем", a1_str2 },
  { "A2", "Автомат управления стержнями", a2_str2 },
  { "A3", "Автомат управления запуском", a3_str2 },
  { "A4", "Автомат управления остановом", a4_str2 },
  { "A5", "Автомат аварийного управления остановом", a5_str2 }
};

//---------------------------------------------------------------------------
void e_find(ubyte e, const char** n, const char** n_name)
{
  static const char* nothing = "нет такого!";
  *n = nothing;
  *n_name = nothing;
  for(uint i = 0; i < 4; i++)
    if(e_str2[i].dig == e){
      *n = e_str2[i].n; *n_name = e_str2[i].n_name; return;
    }
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
void log_a_begin_user(const char* a, const char* a_name, const char* y, const char* y_name, const
char* e, const char* e_name);
void log_a_begin(ubyte a, ubyte y, ubyte e)
{
  const char *e_n, *e_n_name;
  e_find(e, &e_n, &e_n_name);
  log_a_begin_user(A_str3[a].n, A_str3[a].n_name, A_str3[a].str2[y].n, A_str3[a].str2[y].n_name,
e_n, e_n_name);
}
//---------------------------------------------------------------------------
void log_a_trans_user(const char* a, const char* a_name, const char* yo, const char* yo_name,
const char* yn, const char* yn_name);
void log_a_trans(ubyte a, ubyte yo, ubyte yn)
{
```

```
  log_a_trans_user(A_str3[a].n, A_str3[a].n_name, A_str3[a].str2[yo].n,
A_str3[a].str2[yo].n_name, A_str3[a].str2[yn].n, A_str3[a].str2[yn].n_name);
}
//---------------------------------------------------------------------------
void log_a_end_user(const char* a, const char* a_name, const char* y, const char* y_name, const
char* e, const char* e_name);
void log_a_end(ubyte a, ubyte y, ubyte e)
{
  const char *e_n, *e_n_name;
  e_find(e, &e_n, &e_n_name);
  log_a_end_user(A_str3[a].n, A_str3[a].n_name, A_str3[a].str2[y].n, A_str3[a].str2[y].n_name,
e_n, e_n_name);
}
//---------------------------------------------------------------------------
void log_x_user(const char* x, const char* x_name, ubyte res);
void log_x(ubyte x, ubyte res)
{
  log_x_user(x_str2[x].n, x_str2[x].n_name, res);
}
//---------------------------------------------------------------------------
void log_z_user(const char* z, const char* z_name);
void log_z(ubyte z)
{
  log_z_user(z_str2[z].n, z_str2[z].n_name);
}
//---------------------------------------------------------------------------
void log_write_user(char ch, const char* str);
void log_write(char ch, const char* str)
{
  log_write_user(ch, str);
}
//---------------------------------------------------------------------------

#endif
```

Файл Visio\Automates\log_user.cpp

```
#include "stdafx.h"
#include "log.h"

#ifdef SWITCH_LOGGING

//---------------------------------------------------------------------------
void log_a_begin_user(const char* a, const char* a_name, const char* y, const char* y_name, const
char* e, const char* e_name)
{
  // LOG_GRAPH_BEGIN 'a'('a_name'): в состоянии 'y'() запущен с событием 'e'('e_name')
    cout << a << " (" << a_name << "): в состоянии "
        << y << " (" << y_name << ") запущен с событием "
        << e << " (" << e_name << ")" << endl;
}
//---------------------------------------------------------------------------
void log_a_trans_user(const char* a, const char* a_name, const char* yo, const char* yo_name,
const char* yn, const char* yn_name)
{
  // LOG_GRAPH_TRANS 'a'('a_name'): перешел из состояния 'yo'() в состояние 'yn'()
    cout << a << " (" << a_name << "): перешел из состояния "
        << yo << " (" << yo_name << ") в состояние "
        << yn << " (" << yn_name << ")" << endl;
}
//---------------------------------------------------------------------------
void log_a_end_user(const char* a, const char* a_name, const char* y, const char* y_name, const
char* e, const char* e_name)
{
  // LOG_GRAPH_END 'a'('a_name'): завершил обработку события 'e'('e_name') в состоянии 'y'()
    cout << a << " (" << a_name << "): завершил обработку события  "
        << e << " (" << e_name << ")" << endl;
}
//---------------------------------------------------------------------------
void log_x_user(const char* x, const char* x_name, ubyte res)
{
  // LOG_X 'x' - 'x_name' - вернул 'res'
    cout << x << " - " << x_name << " - вернул " << (int)res << endl;
}
//---------------------------------------------------------------------------
void log_z_user(const char* z, const char* z_name)
{
  // LOG_Z 'z'. 'z_name'
    cout << z << ". " << z_name << endl;
}
```

```
//---------------------------------------------------------------------------
void log_write_user(char ch, const char* str)
{
  // ch, str
    cout << ch << ", " << str << endl;
}
//---------------------------------------------------------------------------

#endif
```

Файл Visio\Automates\types.h

```
//--- this file is machine generated ---

#ifndef TypesH
#define TypesH

typedef unsigned char ubyte;
typedef signed char sbyte;
typedef unsigned short int usint;
typedef signed short int ssint;
typedef unsigned int uint;
typedef signed int sint;
typedef unsigned long ulong;
typedef signed long slong;

#endif
```