электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Применение графов в разработке программ для ПЛК

 # 02, февраль 2015

Новожилов Б. М.[1,*]

УДК: 681.3.06

[1]Россия, МГТУ им. Н.Э. Баумана
[*]boris.m.novozhilov@yandex.ru

Технической базой современных систем промышленной автоматизации являются программируемые логические контроллеры (ПЛК). Диапазон их применения простирается от устройств локального управления рабочими органами машин и технологических агрегатов до многоуровневых систем управления сложными технологическими объектами [5]. ПЛК выпускаются в широкой номенклатуре такими крупными зарубежными компаниями, как Siemens, Schneider Electric, Mitsubishi Electric, Rockwell Automation и др. В России ПЛК выпускаются отечественными высокотехнологичными промышленными компаниями, например, компанией Fastwel.

Программируемый логический контроллер представляет собой универсальную микропроцессорную систему, аппаратные и программные средства которой специально адаптированы для решения задач управления техническими объектами в условиях промышленной среды. Основой ПЛК является микроконтроллер или универсальный процессор, осуществляющий обработку информации, представленной двоичными числами.

В системах логического управления ПЛК выполняет функции управляющего автомата (УА), который реализует алгоритм функционирования объекта управления (ОУ) [1]. Он осуществляет сбор сигналов от датчиков и обрабатывает их по программе с последующей выдачей управляющих сигналов на исполнительные устройства ОУ. Специализация ПЛК на выполнение заданного алгоритма управления техническим объектом осуществляется прикладной программой пользователя. Выполняемые в ней преобразования описываются формулами, отражающими с помощью формальных языков функциональные взаимосвязи между логическими переменными процесса.

Известно, что основным принципом применения ПЛК является их доступность для пользователя-технолога, который обычно и является заказчиком автоматизации технологического процесса, но не обладает знаниями профессионального программиста. Поэтому при разработке прикладных программ для ПЛК используются специализированные, проблемно-ориентированные языки программирования, которые понятны пользователю, знакомому с основами аналоговой и цифровой автоматики, а также имеющему опыт работы в области информатики.

Для создания прикладных программ производители контроллеров выпускают специальное программное обеспечение (ПО), устанавливаемое на компьютер разработки. Как правило, оно представляет собой интегрированную среду, содержащую редакторы для написания программ, конфигурирования аппаратных средств ПЛК и устройств человеко-машинного интерфейса (HMI), создания коммуникационных связей, а также средства для компилирования и тестирования готовых программ. Для ПО разработки обязательной является поддержка языков программирования стандарта IEC 61131-3. В их состав входят [2]:

- язык лестничных диаграмм (*Ladder Diagram* – LD);
- язык функциональных блоковых диаграмм (*Function Block Diagram* - FBD);
- список инструкций (*Instruction* List - IL*);*
- структурированный текст *(Structured* Text – ST).

Первые два из перечисленных языков являются графическими языками, остальные – текстовыми. Графические языки LD и FBD просты в применении и сравнительно легко осваиваются пользователями. Программы, написанные на этих языках, напоминают релейно-контактные схемы и логические схемы, соответственно. Текстовые языки IL и ST требуют от пользователя определенных навыков программирования с использованием низкоуровневых символьных языков типа *Assembler* (для языка IL) или высокоуровневых языков типа *Pascal* (для языка ST).

Однако знания того или иного языка программирования ПЛК недостаточно для успешной разработки прикладной программы. Необходимо владеть формальными методами перехода от содержательных понятий словесного описания алгоритмов управления, контроля или диагностики к абстрактным понятиям управляющей логики. Без знания этих методов программирование ПЛК будет носить эвристический характер, а разработанные прикладные программы будут иметь неудовлетворительные характеристики по объему, быстродействию, функциональности и другим показателям.

Формальное описание работы УА в составе системы управления связано прежде всего с разработкой его математической модели на основании описания функционирования ОУ.

Основным свойством объектов логического управления является свойство пребывать в течение определенных промежутков времени в относительно неизменном состоянии (свойство дискретности). При этом в некоторые моменты времени происходит переход ОУ из одного состояния в другое. Обеспечение заданных переходов между различными состояниями ОУ и является целью логического управления объектом.

Управляющий автомат обычно представляют в виде дискретного устройства с памятью. В качестве формальной модели УА используют чаще других модель конечного детерминированного автомата (КДА), который имеет конечное число внутренних состояний, работает с дискретными (двоичными) входными и выходными сигналами. Функционирует автомат в дискретном времени и в каждый момент времени может находиться только в одном состоянии. В управляемом процессе каждое состояние автомата связыва-

ют с одним из физических состояний ОУ, и переход автомата в новое состояние происходит при наступлении некоторого события в системе управления. В свою очередь, это приводит к изменению состояния ОУ в соответствии с ходом управляемого процесса.

Смена состояний автомата, задающая дискретность (тактность) времени, происходит при изменении входных сигналов. Моделью ПЛК является синхронный конечный автомат, в котором тактность времени задается сигналами системной синхронизации, подаваемыми на отдельный вход автомата.

Функционирование КДА заключается в генерировании последовательности выходных сигналов при возбуждении автомата в определенные моменты времени некоторой последовательностью входных сигналов. Среди многообразия способов описания поведения автомата наибольшее распространение получили табличный и графический методы. Первый из них использует так называемые автоматные таблицы (таблицы переходов и таблицы выходов, а также совмещенные таблицы переходов и выходов), второй – представление работы автомата средствами условных графических изображений с указанием внутренних состояний автомата, его состояний входа и выхода (временные диаграммы и циклограммы, схемы алгоритмов и графы переходов). Одним из преимуществ графических методов описания автомата является возможность получения булевых формул управляющей логики непосредственно из графического представления работы автомата.

Для автоматов с памятью при их программной реализации в качестве алгоритмической модели предпочтительнее использовать графы переходов (диаграммы состояний) [6]. Граф переходов автомата – это ориентированный граф, вершины которого соответствуют состояниям автомата, а дуги – переходам между ними (рис. 1).

Дуга, направленная из вершины c_i в вершину c_j , задает переход в автомате между этими состояниями. Дуга, представляющая собой петлю, отражает сохранение состояния, в котором автомат находится до тех пор, пока выполняется условие, реализующее петлю. Состояния входа и выхода автомата указываются на графе соответствующими наборами внешних переменных. Обычно дуги обозначаются булевыми функциями от входных переменных и переменных, отражающих временные соотношения в управляющем процессе. Эти функции отражают состояние управляемого процесса и их единичные значения определяют возможность выполнения переходов. Состояния выхода проставляются на графе в зависимости от выбранной модели автомата. Предпочтительной для описания поведения УА является модель автомата Мура, у которого значения выходных переменных зависят от текущего состояния автомата и не зависят от значений входных переменных [4]. Это позволяет указывать в явном виде состояния выхода автомата у вершин его графа по направлению дуг. Выбор модели автомата Мура облегчает понимание графа, делает более наглядными и ясными причинно-следственные связи в системе управления.

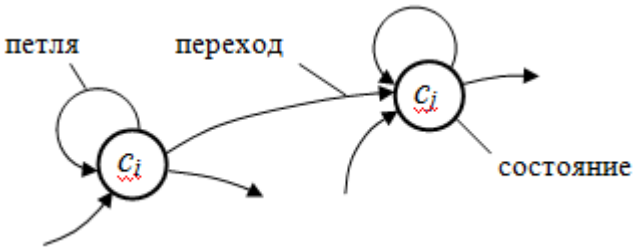

Рис. 1. Элементы ориентированного графа

Смысловое содержание внешних переменных, используемых в графе переходов должно определяться путем разработки схемы связей "управляющий автомат – объект управления".

Для формального описания внутренних состояний автомата вводят конечное число абстрактных внутренних сигналов (вторичных переменных), что позволяет поставить в соответствие каждому состоянию автомата их различные комбинации.

Разработку графа переходов для конкретного ОУ начинают с построения технологического графа, описывающего алгоритм управления данным объектом [3]. Технологический граф строят по словесному описанию процесса, определив внешние входные и выходные сигналы, а также сигналы таймеров, задающих временные интервалы в управляемом процессе. Число вершин в технологическом графе должно соответствовать числу состояний ОУ (включая состояния, соответствующие его неисправностям или неправильным действиям оператора).

В качестве начального состояния, с которого начинается последовательность событий в ОУ, выбирают обычно состояние ожидания, в которое УА должен быть переведен по сигналу начальной установки (инициализации). Далее, используя причинно-следственные связи между сигналами и состояниями в системе управления и модель автомата Мура, определяют переходы из одного состояния в другое, начиная с первого входного сигнала. При этом в явной форме проставляют условия выполнения переходов на дугах автомата и выходные сигналы у его вершин. Например, входные сигналы: $T \leq +5$ $^{\circ}$C, Δ = 10 с, $p \geq 3$ бар, выходные сигналы: клапан 6, вентилятор 3, таймер 1 и т. д. Построение переходов технологического графа из одного состояния в другое продолжают до тех пор, пока не образуется замкнутый граф (или подграф), что свидетельствует об образовании цикла переходов. Появление на графе тупикового состояния говорит, как правило, об ошибке в построении графа или о незавершенности описания событий в ОУ.

Проблему управления параллельными процессами решают декомпозицией автомата: алгоритм представляют в виде нескольких графов, решая при этом задачу их согласования.

Процесс разработки продолжают назначением переменных с присвоением им символьных имен на языке проекта с использованием аббревиатур или полных слов (обычно на английском языке), отражающих семантику сигналов. На рис. 2 в общем виде показаны три вершины графа переходов, соответствующие некоторым внутренним состояниям ав-

томата и обозначенные буквами английского алфавита *ST* (от слова *State* – состояние). Дуги графа обозначены буквами *TR* (от слова *Transition* – переход). Состояния выходов автомата обозначены символами *Q* (от слова *Quit* – покидать).

Следует отметить, что при формальном описании синхронного автомата тактовые сигналы на его графе переходов обычно не выделяют.

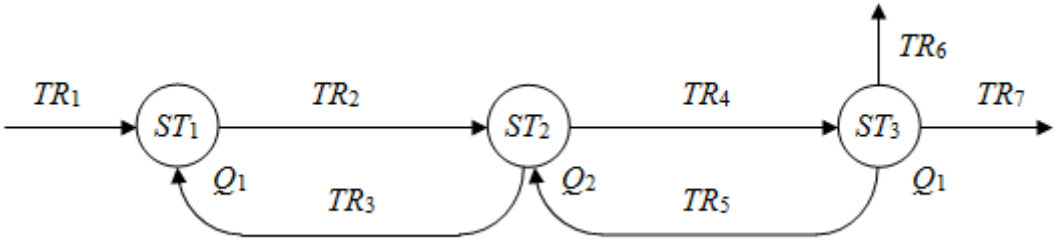

Рис. 2. Фрагмент графа переходов

Присвоение вершинам графа вторичных переменных называют кодированием графа. Среди известных способов кодирования для пользователя-технолога наиболее предпочтителен способ кодирования, при котором каждой вершине графа переходов присваивают одну двоичную переменную (переменную состояния), принимающую единичное значение только в данной вершине и нулевое во всех остальных вершинах графа.

Для удобства описания графа его дугам также присваивают двоичные переменные (переменные переходов), которые принимают значение логической единицы (становятся активными) при выполнении данного перехода.

Переменная состояния должна устанавливаться в единицу единичными сигналами дуг, входящих в данную вершину, и сбрасываться в ноль единичными сигналами дуг, выходящих из вершины. Кроме того, переменная состояния должна сохранять свое значение до тех пор, пока оно не будет изменено сигналами входящих или выходящих дуг соответствующей вершины. Этими свойствами обладает элемент памяти, функционирующий как *RS*-триггер с приоритетом по входу *R*. Т. е. переменная состояния принимает нулевое значение при единичном значении переменной выходящего перехода независимо от значения переменной входящего перехода. Логическая функция такого элемента памяти в обозначениях графа переходов описывается следующей формулой (знаки действий соответствуют логическим операциям):

$$ST = (ST + A) \cdot \overline{B}\,, \qquad (1)$$

где A – сигнал установки переменной состояния в единицу, B – сигнал сброса переменной состояния в ноль.

Логические условия установки и сброса переменной состояния определяют с учетом того обстоятельства, что поведение автоматов с памятью в отличие от автоматов без памяти зависит от предыстории. Поэтому каждый переход автомата зависит от одного, непосредственно связанного с ним состояния. Другими словами, переход реализуем, если

автомат находится в состоянии, из которого он выходит, и состояние управляемого процесса соответствует выполнению данного перехода. Т. е. любая переменная перехода *TR* на графе может быть определена как логическое произведение переменной состояния и условия выхода из него, выраженного булевой переменной:

$$TR = ST \cdot B, \qquad (2)$$

где *ST* – переменная состояния, *B* – логическое условие выхода из данного состояния.

Тогда для *k*-ой вершины графа автомата формулу (1) можно записать в следующем виде:

$$ST_k = (ST_k + \sum_i^I TR_i) \cdot \overline{\sum_j^J TR_j} \text{ или } ST_k = (ST_k + \sum_i^I TR_i) \cdot \prod_j^J \overline{TR_j}, \qquad (3)$$

где индекс *i* соответствует дугам, входящим в данную вершину графа, а индекс *j* – дугам, выходящим из данной вершины; *I* и *J* – число дуг, входящих в данную вершину графа и выходящих из нее, соответственно.

Например, для вершины ST_2 графа, показанного на рис. 2, формула состояния автомата имеет вид:

$$ST_2 = (ST_2 + TR_2 + TR_5) \cdot \overline{TR_3} \cdot \overline{TR_4}$$

Таким образом, программа для ПЛК должна содержать две системы булевых функций – функций переходов *TR* и функций состояний *ST*. Выбор языка программирования при этом большого значения не имеет. На рис. 3 показана программная реализация состояния автомата на языках LAD (клон языка LD) и FBD в среде ПО STEP7 V.12 компании Siemens [7]. Здесь переменными *Set_condition* и *Reset_condition* обозначены логические условия установки и сброса переменной состояния *ST*_1, соответственно, и им назначены адреса в памяти контроллера.

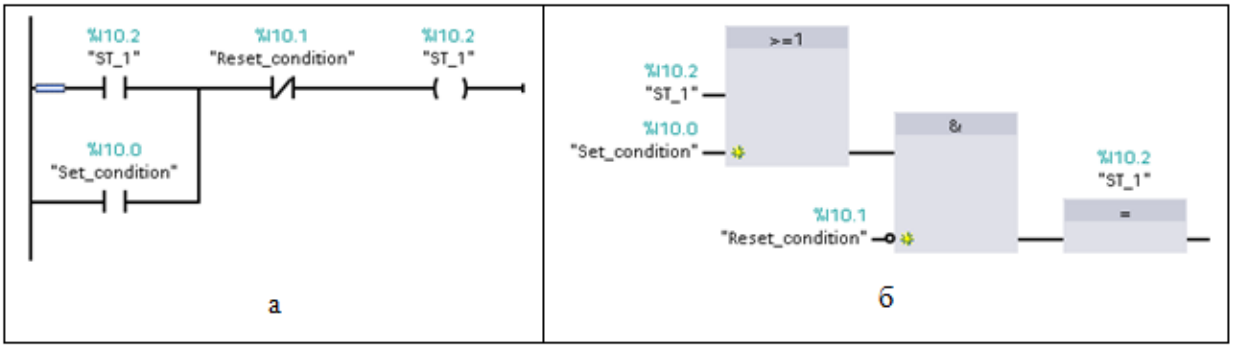

Рис. 3. Исполнение элемента памяти: а – на языке LAD; б – на языке FBD

Следует отметить, что в составе команд битовой логики современных контроллеров имеются команды *RS*-триггеров с приоритетами. В частности, команда *SR*: *Set/reset flip-flop* из STEP7 V.12 реализует функцию элемента памяти на *RS*-триггере с приоритетом входа сброса. Это позволяет упростить и сделать более наглядным текст программы. На рис. 4 показан вид вычислительной цепочки на языках LAD и FBD с использованием указанной команды.

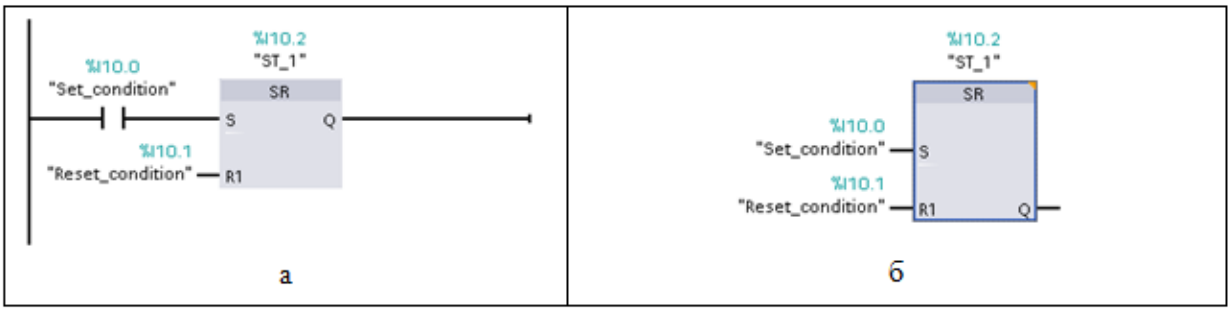

Рис. 4. Программа с использованием команды *SR*-триггера: а – на языке LAD; б – на языке FBD

Для завершения программы систему булевых функций переходов и состояний дополняют булевыми функциями выходов, связывая их с определенными состояниями графа переходов. Например, для графа по рис. 2 формулы для вычисления выходных переменных Q_1 и Q_2 имеют следующий вид:

$$Q_1 = ST_1 + ST_3 \text{ и } Q_2 = ST_2.$$

Рассмотрим применение изложенной методики разработки программы для ПЛК на примере контура управления двумя вентиляторами системы поддержания температурного режима компрессорной станции.

Алгоритм управления. В исходном состоянии, когда компрессор выключен, оба вентилятора выключены. Если компрессор включен и температура воздуха в помещении станции достигает +25 °С, включается основной вентилятор. Одновременно таймером задается интервал времени 3 минуты, по истечении которого проверяется температура воздуха. Если она опустилась ниже +20 °С, вентилятор выключается, при последующем повышении температуры до +25 °С вентилятор снова включается и т. д. Если при включении основного вентилятора температура воздуха продолжает повышаться, то по достижении значения +30 °С включается дополнительный вентилятор и снова задается временной интервал 3 минуты. Если по истечении этого времени температура опустилась ниже +25 °С, дополнительный вентилятор выключается, а основной вентилятор продолжает работать. Если при работе обоих вентиляторов температура воздуха в помещении станции повышается до +35 °С, то выдается сигнал предупреждения. Этот сигнал снимается при снижении температуры до +30 °С. Выключение компрессора возвращает управляющий автомат в исходное состояние с выключением обоих вентиляторов.

Задачу нахождения формул управляющей логики для ПЛК решают в следующем порядке.

Шаг 1. Определение входных и выходных сигналов управляющего автомата с присвоением им символьных имен

Входные сигналы:

- сигнал включенного состояния компрессора – *K* (= 1 при работающем компрессоре);

- сигнал от датчика температуры воздуха – T (сигнал аналоговый, поэтому требуется наличие в контроллере аналогового входа и масштабирование в управляющей программе).

Выходные сигналы:

- сигнал управления основным вентилятором – *fan*1 (= 1 для включения);
- сигнал управления дополнительным вентилятором – *fan*2 (= 1 для включения);
- сигнал предупреждения – *alarm* (= 1 при превышении предельной температуры воздуха в помещении).

Внутренние сигналы:

- сигнал запуска первого таймера – *timer*1 (= 1 для запуска таймера);
- сигнал запуска второго таймера – *timer*2 (= 1 для запуска таймера);
- сигнал окончания счета первого таймера – Δt_1 (= 1 при окончании счета);
- сигнал окончания счета второго таймера – Δt_2 (= 1 при окончании счета);

сигнал начальной установки – *InitBit* (= 1 в первом рабочем цикле);

Шаг 2. Построение графа переходов управляющего автомата

На рис. 5 показан граф переходов, построенный по словесному описанию алгоритма управления (черный цвет графических фигур).

Граф содержит 7 вершин, соответствующих семи состояниям объекта управления. Дуги графа помечены условиями переходов управляющего автомата из одного состояния в другое. Вершинам графа и дугам присвоены индексированные символьные имена состояний и переходов, отмеченные на графе синим и красным цветом, соответственно.

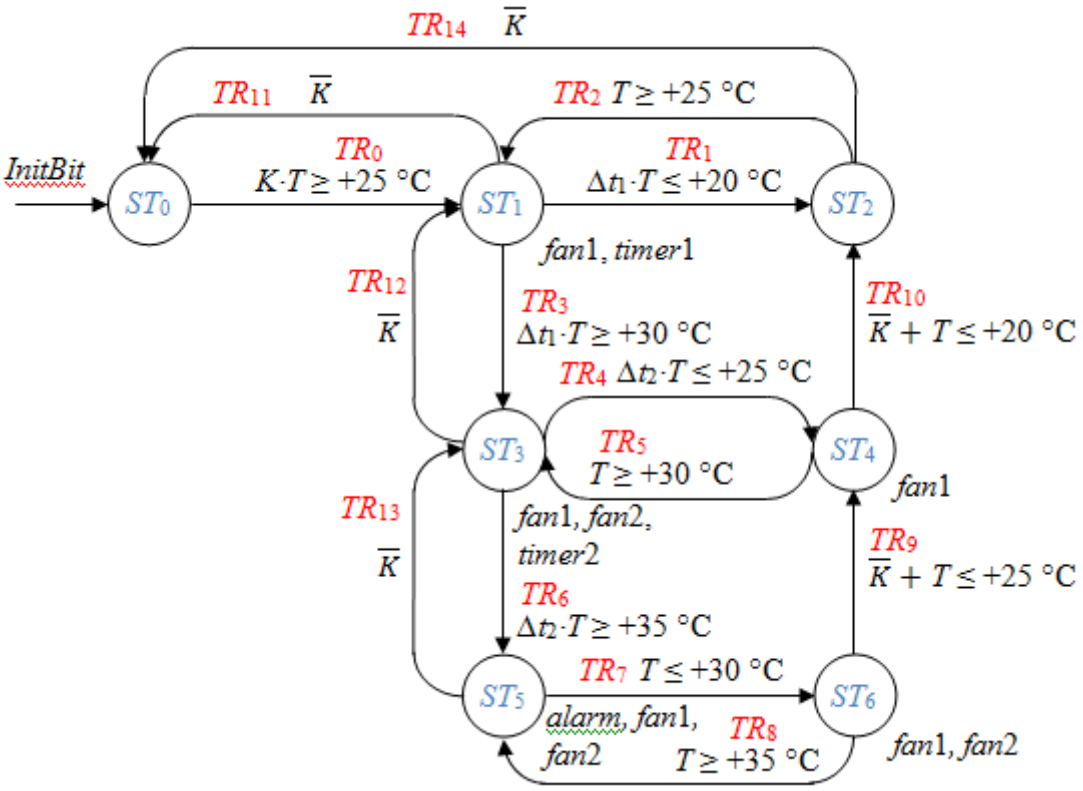

Рис. 5. Граф переходов управляющего автомата

Начальное состояние автомата (ST_0) устанавливается внутренней переменной (*InitBit*) на первом рабочем цикле (скане). Это может быть определенный бит системной памяти, имеющий значение логической единицы на первом скане, или выход функционального блока инициализации, выполняющего на первом скане начальные установки программы. Сюда же, в состояние ST_0, переходит автомат из любого состояния при выключении компрессора.

Выходные состояния автомата проставлены у вершин графа снизу. Как следует из графа, выходы контроллера, к которым подключаются вентиляторы *fan*1 и *fan*2, а также выход управления сигналом *alarm*, должны настраиваться соответствующим командами в режим потенциального управления.

Шаг 3. Формирование систем логических функций управляющей логики

Функции переходов (см. формулу (2)):

$TR_0 = ST_0 \cdot K \cdot T \geq +25\ ^\circ C$	$TR_8 = ST_6\ T \geq +35\ ^\circ C$
$TR_1 = ST_1 \cdot \Delta t_1 \cdot T \leq +20\ ^\circ C$	$TR_9 = ST_6 \cdot (\overline{K} + T \leq +25\ ^\circ C)$
$TR_2 = ST_2 \cdot T \geq +25\ ^\circ C$	$TR_{10} = ST_4 \cdot (\overline{K} + T \leq +20\ ^\circ C)$
$TR_3 = ST_1 \cdot \Delta t_1 \cdot T \geq +30\ ^\circ C$	$TR_{11} = ST_1 \cdot \overline{K}$
$TR_4 = ST_3 \cdot \Delta t_2 \cdot T \leq +25\ ^\circ C$	$TR_{12} = ST_3 \cdot \overline{K}$
$TR_5 = ST_4 \cdot T \geq +30\ ^\circ C$	$TR_{13} = ST_5 \cdot \overline{K}$
$TR_6 = ST3 \cdot \Delta t_2 \cdot T \geq +35\ ^\circ C$	$TR_{14} = ST_2 \cdot \overline{K}$
$TR_7 = ST_5 \cdot T \leq +30\ ^\circ C$	

Функции состояний (см. формулу (3)):

$ST_0 = (ST_0 + InitBit + TR_{11} + TR_{14})\ \overline{TR_0}$

$ST_1 = (ST_1 + TR_0 + TR_2 + TR_{12})\ \overline{TR_1} \cdot \overline{TR_3} \cdot \overline{TR_9}$

$ST_2 = (ST_2 + TR_1 + TR_{10})\ \overline{TR_2} \cdot \overline{TR_{14}}$

$ST_3 = (ST_3 + TR_3 + TR_5 + TR_{13})\ \overline{TR_4} \cdot \overline{TR_6} \cdot \overline{TR_{12}}$

$ST_4 = (ST_4 + TR_4 + TR_9)\ \overline{TR_5} \cdot \overline{TR_{10}}$

$ST_5 = (ST_5 + TR_6 + TR_8)\ \overline{TR_7} \cdot \overline{TR_{13}}$

$ST_6 = (ST_5 + TR_7)\ \overline{TR_8} \cdot \overline{TR_9}$

Выходные сигналы:

$fan1 = ST_1 + ST_3 + ST_4 + ST_5 + ST_6$

$fan2 = ST_3 + ST_5 + ST_6$

$Alarm = ST_5$

Внутренние сигналы:

*timer*1 = ST1

*timer*2 = ST3

Далее пишут программу, предварительно присвоив всем переменным входов/выходов, а также внутренним переменным адреса в памяти контроллера.

Рассмотренный метод разработки программ для ПЛК прошел проверку на ряде проектов различной сложности. Он понятен как профессиональному программисту, так и специалисту в области промышленной автоматики, который может принимать непосредственное участие в разработке управляющей программы и ее последующем сопровождении. К недостаткам метода можно отнести сравнительно большой объем исполняемого кода. Однако объем кода можно сократить применением многозначного кодирования вершин графа переходов с последующим использованием в программе команд *RS*- триггеров.

Список литературы

1. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов / В. Лазарев, У. Пийль; - 3-е изд. – М.: Энергоатомиздат, 1989. 328 с.
2. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного программирования / И. Петров; под ред. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2004. 256 с.
3. Хоуп Г. Проектирование цифровых вычислительных устройств на интегральных схемах / Г. Хоуп; пер. с англ. М.: Мир, 1984. 400.
4. Шалыто А.А. Алгоритмизация и программирование задач логического управления / А. Шалыто; Изд–во СПбГУ ИТМО, 1998. 56 с.
5. Бармин И.В., Королев Н.С., Чугунков В.В. Особенности построения заправочной системы сжиженного природного газа на стартовом комплексе. // Инженерный вестник. Электронный научно-технический журнал # 12, декабрь 2014. http://engbul.bmstu.ru/doc/744047.html
6. The PLC Book – URL: http://www.theautomationstore.com/the-plc-book/
7. SIMATIC S7. S7-1200 Programmable controller: System Manual. – SIEMENS, 04/2012. 864 p.