Статья опубликована в журнале "Промышленные АСУ и контроллеры". 2006. № 6, с. 43 – 45.

К.В. ВАВИЛОВ (ООО “НИИЭФА-ЭНЕРГО”),
А.А. ШАЛЫТО (СПбГУ ИТМО)

LabVIEW и *SWITCH*-технология

Изложен подход к применению технологии автоматного программирования (SWITCH-технологии), которая позволяет повысить эффективность применения пакета LabVIEW для программирования задач логического управления.

In this article the automata-based approach (SWITCH-technology) is used together with LabVIEW application suite for programming logic control problems.

С 1991 г. в России развивается технология, основанная на применении конечных автоматов в программировании [1]. При этом реализуемые алгоритмы описываются с помощью графов переходов. От этих графов весьма просто и удобно переходить к текстам программ на языке *Си*, используя оператор `switch`.

Пакет графического программирования *LabVIEW* [2] является продуктом, который широко применяется, в частности при автоматизации в промышленности. Однако, если в *LabVIEW* на языке функциональных блоков (язык *G*) предстоит реализовать сложную логику, то программа получается весьма громоздкой и трудно читаемой. Несмотря на то, что в работах [3, 4] были предложены методы реализации графов переходов программами на языке функциональных блоков, указанная проблема остается нерешенной.

Для устранения указанного недостатка в настоящей работе предлагается использовать совместно *LabVIEW* и *SWITCH*-технологию.

Предлагаемый подход

Суть предлагаемого подхода состоит в том, что формирование входной и выходной информации осуществляется с помощью функциональных блоков, а логика программы реализуется на языке *Си* в виде текста, построенного формально и изоморфно по графам переходов. Построение текста программы осуществляется вручную или автоматически, используя конвертор, реализованный на *LabVIEW* или другом программном средстве.

Такой подход позволяет описывать алгоритмы на высоком уровне абстракции, сохранять это описание в тексте программы, использовать графы переходов для тестирования программ, протоколировать работу программы в терминах автоматов, документировать программное обеспечение в виде, удобном для пользователя. Обратим внимание, что при необходимости изменения логики программы они вносятся не в текст программы, а в граф переходов с последующим преобразованием его (вручную или автоматически) в текст программы.

Ввиду того, что в рамках предлагаемого подхода программы строятся так же, как и автоматизируются объекты управления, для описания интерфейса указанных автоматов строится схема связей, содержащая источники информации, автомат, объекты управления и, при необходимости, обратные связи.

Пример реализации графа переходов на основе предлагаемого подхода

1. Пусть задан фрагмент ТЗ на управление питанием приборов измерения давления (адаптированный текст инструкции по эксплуатации).

Питание прибора измерения давления подается вручную или автоматически. После подачи питания прибор прогревается (вводится в рабочий режим) в течение не более 5 мин. Время не-

прерывной работы прибора не более 24 ч. После отключения питания прибору необходимо восстанавливаться в течение 30 мин.

2. Используя технологию, описанную в работе [1], приведенный фрагмент формализуется с помощью схемы связей автомата с источниками информации и объектами управления (рис. 1) и графа переходов (рис. 2), описывающего поведение автомата на схеме связей.

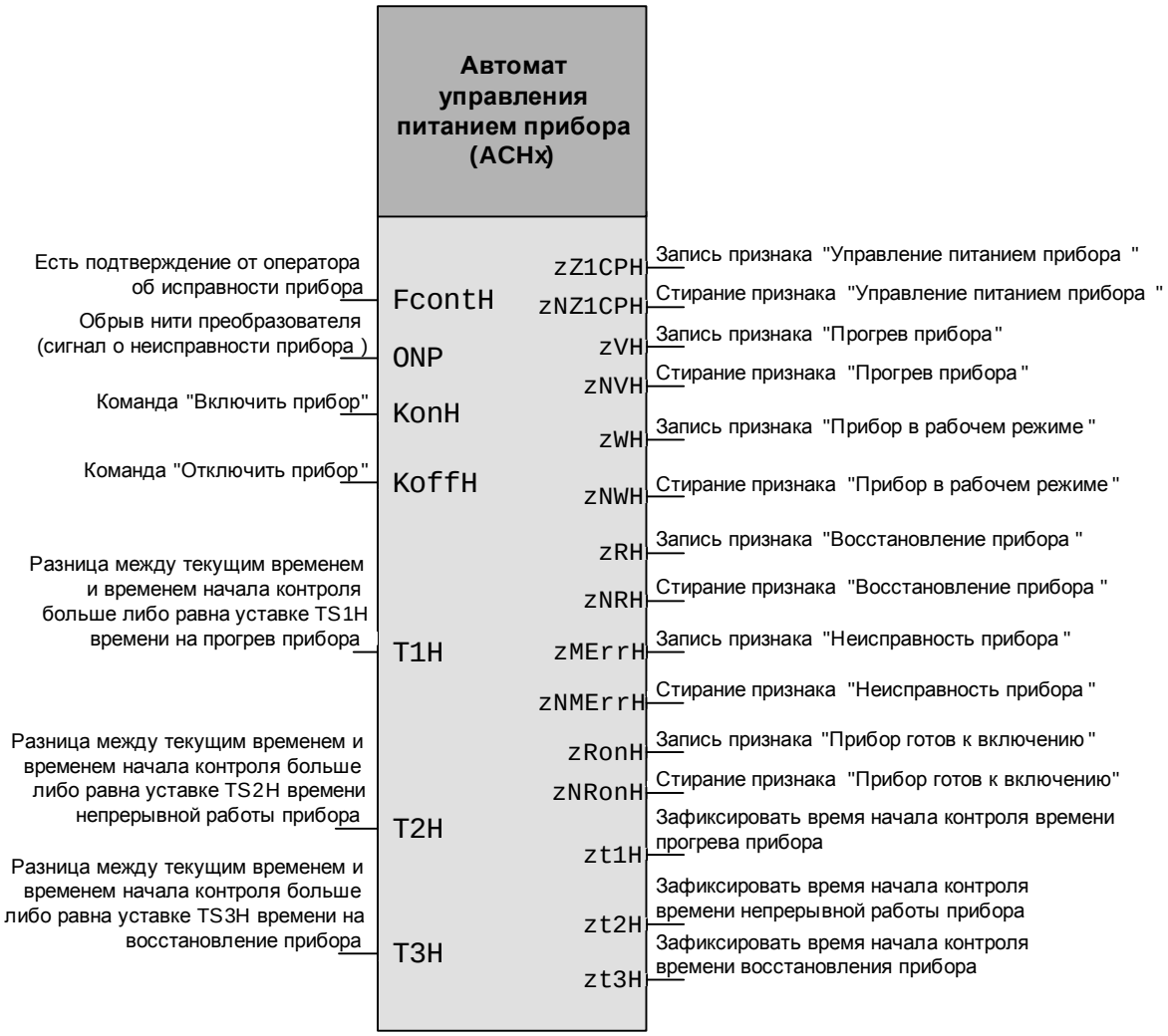

Рис. 1. Схема связей автомата

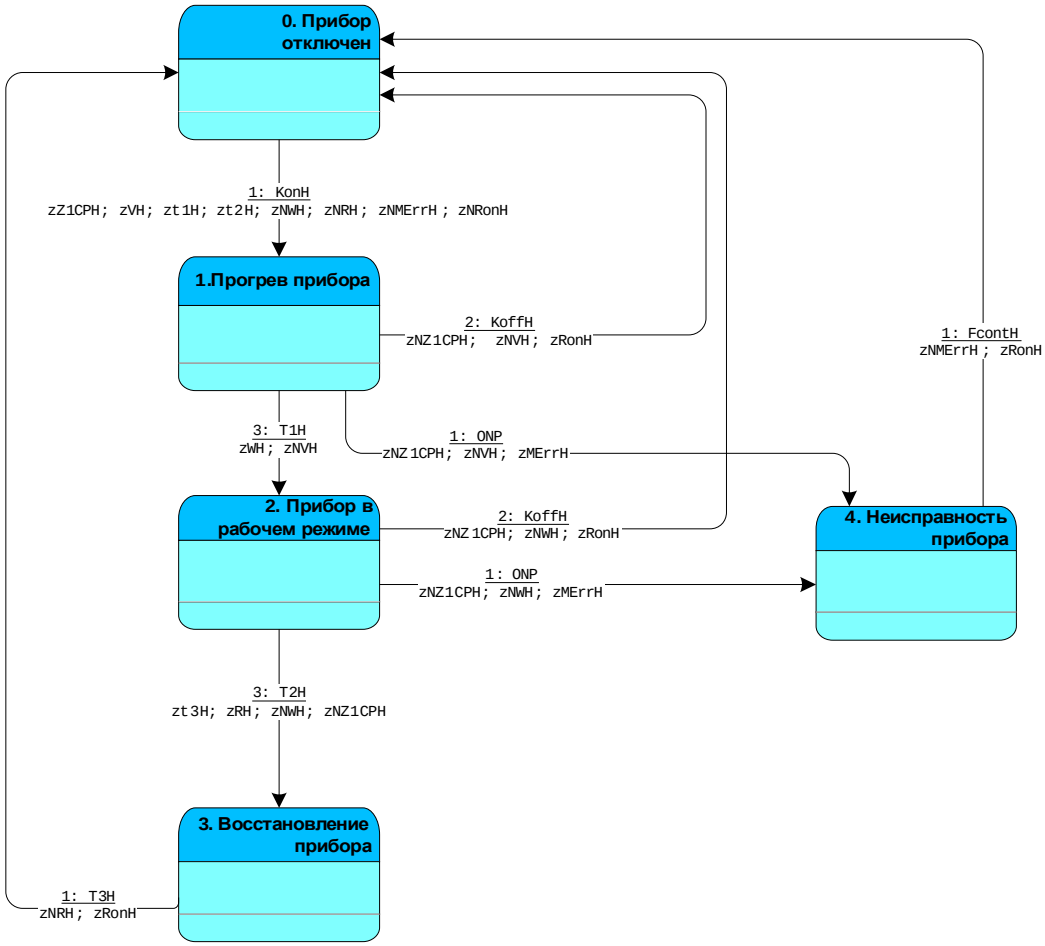

Рис. 2. Граф переходов автомата

3. Используя разработанный авторами конвертор “граф переходов – текст на языке *Си*”, строится следующий текст программы на основе оператора `switch`:

```
// Обнуление (сброс) команд, признаков и т.п., записываемых в автомате,
// которые не имеют обнуления по условию
t1H = 0; t2H = 0; t3H = 0;
// Реализация графа переходов
switch (YACHx) {
case 0: if (KonH) {Z1CPH = 1; VH = 1; t1H = 1; t2H = 1; WH = 0;
        RH = 0; MErrH = 0; RonH = 0; YACHx = 1;} break;

case 1: if (ONP) {Z1CPH = 0; VH = 0; MErrH = 1; YACHx = 4;}
        else if (KoffH) {Z1CPH = 0; VH = 0; RonH = 1; YACHx = 0;}
        else if (T1H) {WH = 1; VH = 0; YACHx = 2;} break;

case 2: if (ONP) {Z1CPH = 0; WH = 0; MErrH = 1; YACHx = 4;}
        else if (KoffH) {Z1CPH = 0; WH = 0; RonH = 1; YACHx = 0;}
        else if (T2H) {t3H = 1; RH = 1; WH = 0; Z1CPH = 0;
        YACHx = 3;} break;

case 3: if (T3H) {RH = 0; RonH = 1; YACHx = 0;} break;
```

```
case 4: if (FcontH) {MErrH = 0; RonH = 1; YACHx = 0;} break;
default: break;};.
```

4. Полученный текст помещается в функцию `Formula Node` (рис. 3).

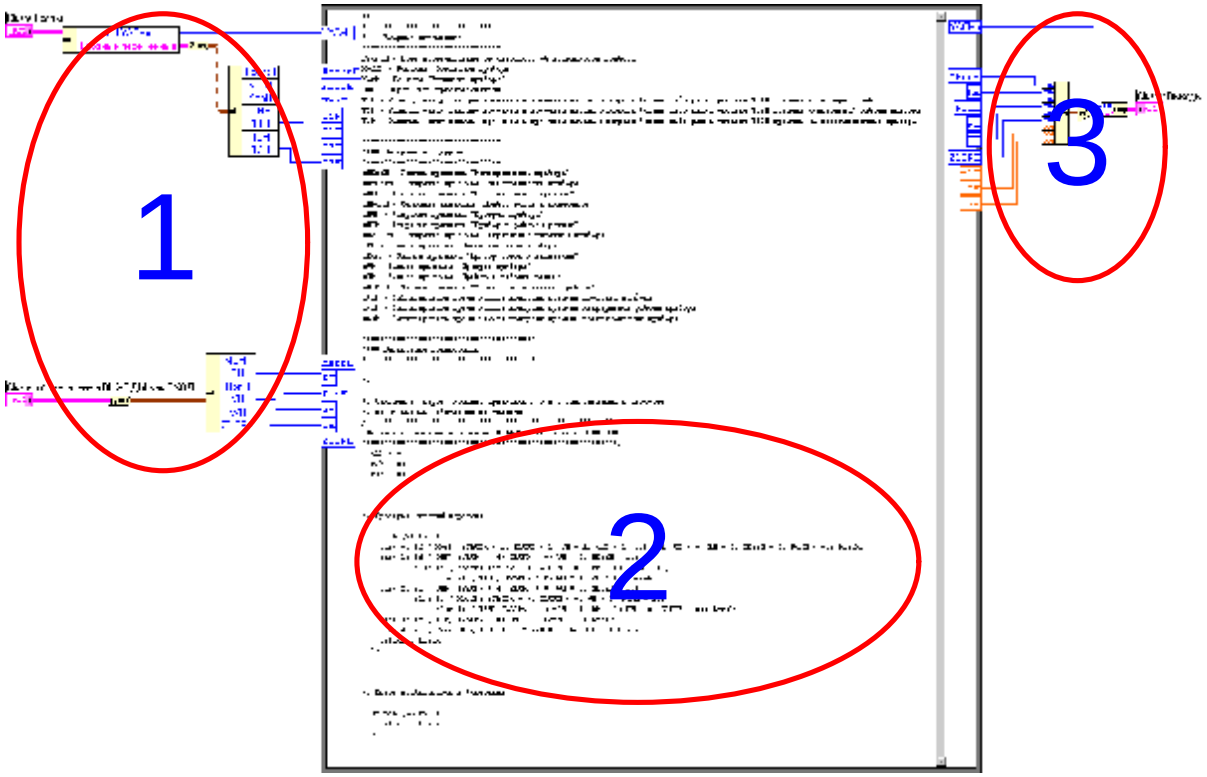

Рис. 3. Подпрограмма автомата - входы (1), обработка в текстовом виде (2) и выходы (3)

5. Фрагмент программы вызова автомата и формирования входной и выходной информации с помощью функциональных блоков языка *G* изображен на рис. 4.

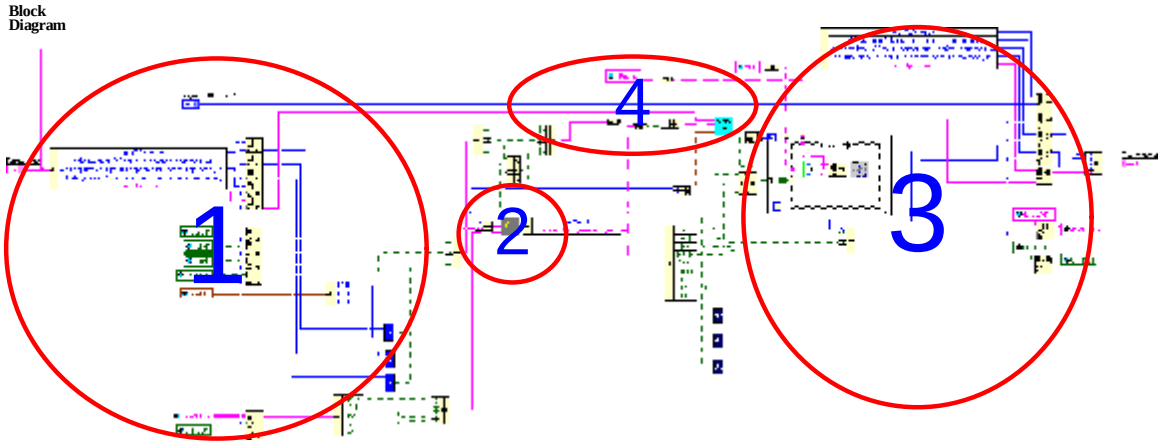

Рис. 4. Вызов автомата и формирование входной и выходной информации

Подпрограмма вызова автомата имеет следующие “блоки” (рис. 4):

– формирование входных параметров для подпрограммы автомата (1);
– вызов подпрограммы автомата (2);
– процедуры формирования значений выходных данных (параметров) автомата (3);
– поддержка протоколирования работы автомата (4);
– если необходимо, поддержка вызова диалогового окна.

6. Программа в таком виде при наличии схемы связей и графа переходов значительно понятней, чем если бы она была целиком построена из функциональных блоков. Это связано с тем, что человеку ближе такие понятия, как “состояние”, “переход”, “входное и выходное воздействия”, по сравнению со схемной реализацией.

7. Выбранный для примера граф переходов для сокращения объема статьи содержит небольшое число состояний. Как будет отмечено ниже, на практике используются графы, содержащие значительно большее число состояний. Эффективность предлагаемого подхода при реализации сложных графов переходов резко возрастает.

Использование предлагаемого подхода при разработке АСУТП

Предложенный подход был использован при разработке АСУ оборудованием криогенно-вакуумной установки.

Количественная характеристика объекта управления:

– нереверсивные приводы (насосы и электромагнитные вентили) – 21;
– реверсивные приводы (затворы и клапаны) – 31;
– управляемые приборы измерения давления – 19.

Необходимо обеспечивать резервирование каждой группы насосов и приборов.

Количественная характеристика логической части проекта:

– типов автоматов – 40 (сложных – 25), общее число автоматов – 107. Каждый тип автомата реализуется подпрограммой на базе оператора `switch`, а автоматы одного типа – передачей в подпрограмму соответствующих параметров;
– число состояний в сложных автоматах – от 6 до 25 (в среднем – 10);
– среднее число переходов в сложных автоматах – 47.

После того, как графы переходов автоматов были построены, их автоматическая конвертация в текст программы на языке *Си* заняла нескольких минут. Формирование входной и выходной информации на языке *G* заняло еще 5 человеко-дней.

Заключение

Опыт разработки АСУТП, рассмотренной выше, и других систем подтвердил эффективность предлагаемого подхода.

Более подробно с изложенным подходом можно будет ознакомиться на сайте http://is.ifmo.ru, раздел “Внедрение”.

Константин Валерьевич Вавилов – начальник группы АСУ, ООО “НИИЭФА-ЭНЕРГО”.
E-mail: wawi@yandex.ru

Анатолий Абрамович Шалыто – д-р техн. наук, профессор, заведующий кафедрой “Технологии программирования” Санкт-Петербургского государственного университета информационных технологий, механики и оптики. E-mail: shalyto@mail.ifmo.ru

Список литературы

1. *Шалыто А.А.* *Switch*-технология. Алгоритмизация и программирование задач логического управления // Промышленные АСУ и контроллеры. 1999. № 9, с.33–37.
2. *Компания Vitec. LabVIEW* за пять минут – первая программа. http://vitec.ru/main.php?action=publ&subaction=articles&param1=article_01.
3. *Шалыто А.А.* Реализация алгоритмов логического управления программами на языке функциональных блоков // Промышленные АСУ и контроллеры. 2000. № 4, с. 45–50. http://is.ifmo.ru/works/logctr/1/
4. *Альтерман И.З., Шалыто А.А.* Формальные методы программирования логических контроллеров //Промышленные АСУ и контроллеры. 2005. № 10, с. 49–52. http://is.ifmo.ru/works/formalcontroller/